The Design of Hybrid Stepping Motors Aided by Three Dimensional Finite Element Analysis

CLIFFORD MARK JOLLIFFE

Thesis submitted for the examination of the degree of Doctor of Philosophy

Department of Engineering
University of Warwick
Coventry, CV4 7AL

January 1999

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 7

GLOSSARY OF TERMS

A is Potential

A is the Cross-sectional area

A_{cond} is the cross-sectional area of a conductor

B is the flux density

E_L is the linearised mesh energy per unit length

f is the frequency of the supply.

H is the field strength

H_c is the field strength coercivity

I is the current

i is the instantaneous current

i_j is the current in phase j, (j=1,2)

i_{10} is the initial current in phase 1

i_{20} is the initial current in phase 2

J is the current density,

L is the inductance

l is the instantaneous inductance

m is the number of phases

N is the number of turns in a conductor

n_r is the number of teeth on the rotor stack.

R is the resistance

S is the steps per revolution,

T is the Torque

t is time

$t_{constant}$ is the electrical time constant

V is the voltage

v is the instantaneous voltage

W_e is the electrical input power

$W_{mech.}$ is the mechanical output power,

W_{field} is the energy stored in the field.

ω_{rpm} is the speed in revolutions per minute

ℓ is the mean flux path length

λ is the flux-linkage

$\lambda_{(\theta,i)}$ flux-linkage in terms of position and current

$\lambda_{(0)}$ is the flux-linkage at time t=0

$\lambda_{m(\theta, i1=0, i2=0)}$ is the flux linkage associated with the stator phase due to the permanent magnet at any position θ,

MMF_{mag} is the MMF produced by the magnet

mmf is the instantaneous MMF

Φ_{mag} is the peak flux generated by the magnet

ϕ is the magnetic flux

$\mathcal{R}_L$ is the mesh reluctance per unit length

θ is the position of the rotor

μ_0 is the permeability of free space

μ_r is the relative permeability of the magnetic circuit

EMF_{mag} is the rotational EMF produced by the magnet

emf_1 is the instantaneous motional EMF induced in phase 1 by the permanent magnet rotor.

Pf is the packing factor

rps are the revolutions per second

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to Dr. Charles Pollock, and Dr. Alex Michaelides, for their patience, encouragement, and supervision.

This project could not have been undertaken without the financial support of Stebon. Ltd., in particular Mr Peter Buonaquisti, managing director. Industrial supervision has been given in the highest calibre by Mr Dave Baker, and I thank him for this. Thank you to all my new colleagues at Stebon, especially to Brian Nutter and Jon Harding.

Gratitude should be made to both John Reeve, and Matts Persson (Hoganas), who individually supported part of this project.

I would like to thank the DTI/ SERC programme and the Department of Engineering for funding and the use of facilities. Also to Vector Fields Ltd. for their excellent support with their Finite Element software.

To my companions in the Power Electronics Lab. Cheers to Mike Barnes (hope all goes well at UMIST), Kevin Richardson (Up the VILLA!), Helen Pollock (great kids), John Wale (bell ringer extraordinare), Andrew Chu (Don't you dare run that Saber simulation now!), T'Shen (It's only a plastic snake), Alex Michaelides (Up with the Turks), and Andy Leeson (gladiator).

Thank you to the Wade family, for their love and support. Special thanks to Kevin Brant, for some excellent intercontinental adventures.

Lastly praise to my wife, who has had to put up with an untidy office, and no fitted wardrobes, but at least I have finished the pond! I still wonder if you know what a hybrid stepper motor is, but I devote this work to you.

SUMMARY

Though the hybrid stepping motor has a long and proven history, in terms of toughness, accuracy of position and the ability to operate in open loop, motor performance improvements can still be made in terms of the physical structure of the motor's components. It is impossible to build a complete solution of the hybrid stepping motor using simple analytical functions or equivalent circuit representations. This is due to the difficulties introduced by the motor's highly non-linear three dimensional magnetic structure, of which the doubly salient tooth structure, axial magnet, and back iron all complicate the situation. However, with the recent advances in three dimensional finite element software a comprehensive study of the motor has been achieved in this thesis. This has allowed improvements to simpler two dimensional based mathematical models, which allow faster computation of the motor's electromagnetic performance. To aid modelling, novel equations which accurately model today's high permeability steels have been developed. These are shown to be more accurate than the established Jiles-Atherton method. Inductance calculations of the steel's flux paths have been comprehensively improved by the use of elliptical functions. The thesis concludes with the design of two quite individual new machines. The first dramatically improves a motor's power output, smoothness, noise levels, and resonance by modifying the tooth structure. The second uses soft magnetic composite materials to provide an isotropic path for cross lamination flux which flows in a stator's back iron. Both new designs are shown to offer a significant improvements to the high speed torque capability of the hybrid stepping motor.

CHAPTER 1

INTRODUCTION

1.1 Introduction

The Hybrid stepping motor is an electromechanical device usually used in positioning applications, but also finds its uses in applications where a high torque and controllable speed are required. Its basic operation is to translate a received electrical pulse into a mechanical movement. The hybrid stepping motor is a member of a large family of stepping motors of which there are many design types available. These range from extremely simple AC synchronous two pole magnet types to the more elaborate in physical design, like the hybrid stepping motor. In this thesis the emphasis is focused on the latter. However it is beneficial that two other members from the family of stepping motors are briefly described so that the evolution of the hybrid stepping motor can be better comprehended.

1.2 Basic Variations of the Stepping Motor

Three basic rotary stepping motors are found in industrial applications. These are the variable reluctance, permanent magnet, and hybrid stepping motor. All are essentially positioning devices, and exhibit the benefits of low cost, ruggedness, simple construction, low maintenance, and open loop control.

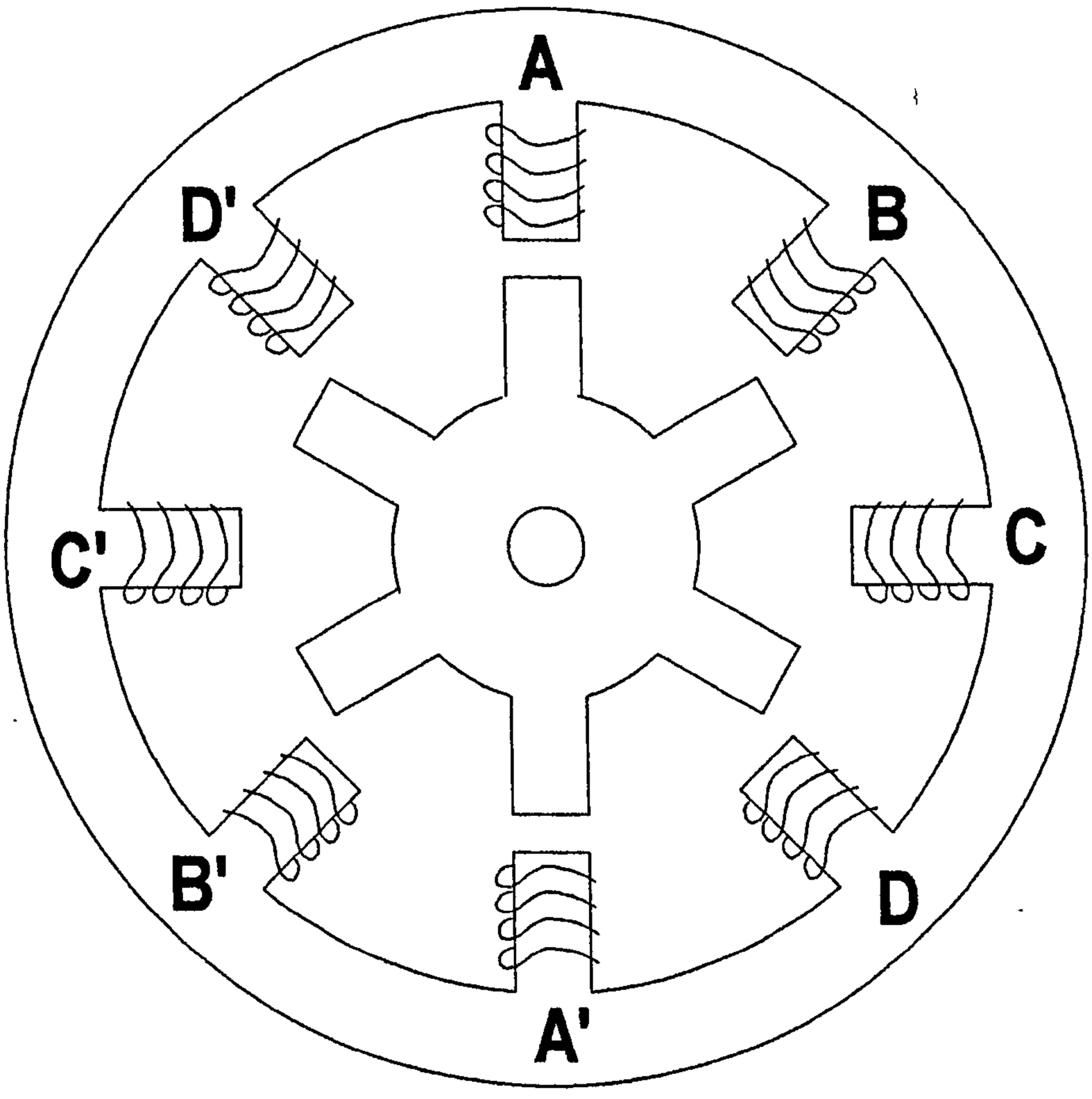


FIGURE 1.1 - Simple Variable Reluctance Motor

1.2.1 Variable Reluctance Motor

The variable reluctance motor has a construction that is rugged and relatively simple [1]. As the name suggests the motor works on the variable reluctance principle, where the change in the rotor position relative to the stator poles relates to a corresponding change in reluctance. This change in reluctance directly affects the flux/ MMF ratio, and hence the torque. In its simplest form each pole may be a single tooth (figure 1.1), but a more practical design of pole is shown in figure 1.2. Here the rotor has numerous teeth and each stator pole has a set of teeth castellations. It should be noted that the simpler tooth structure in figure 1.1, is found more commonly in the type of motor called the switched

reluctance motor, which is presently seeing growing interest due to its simple mechanical structure and improving drive electronics [2].

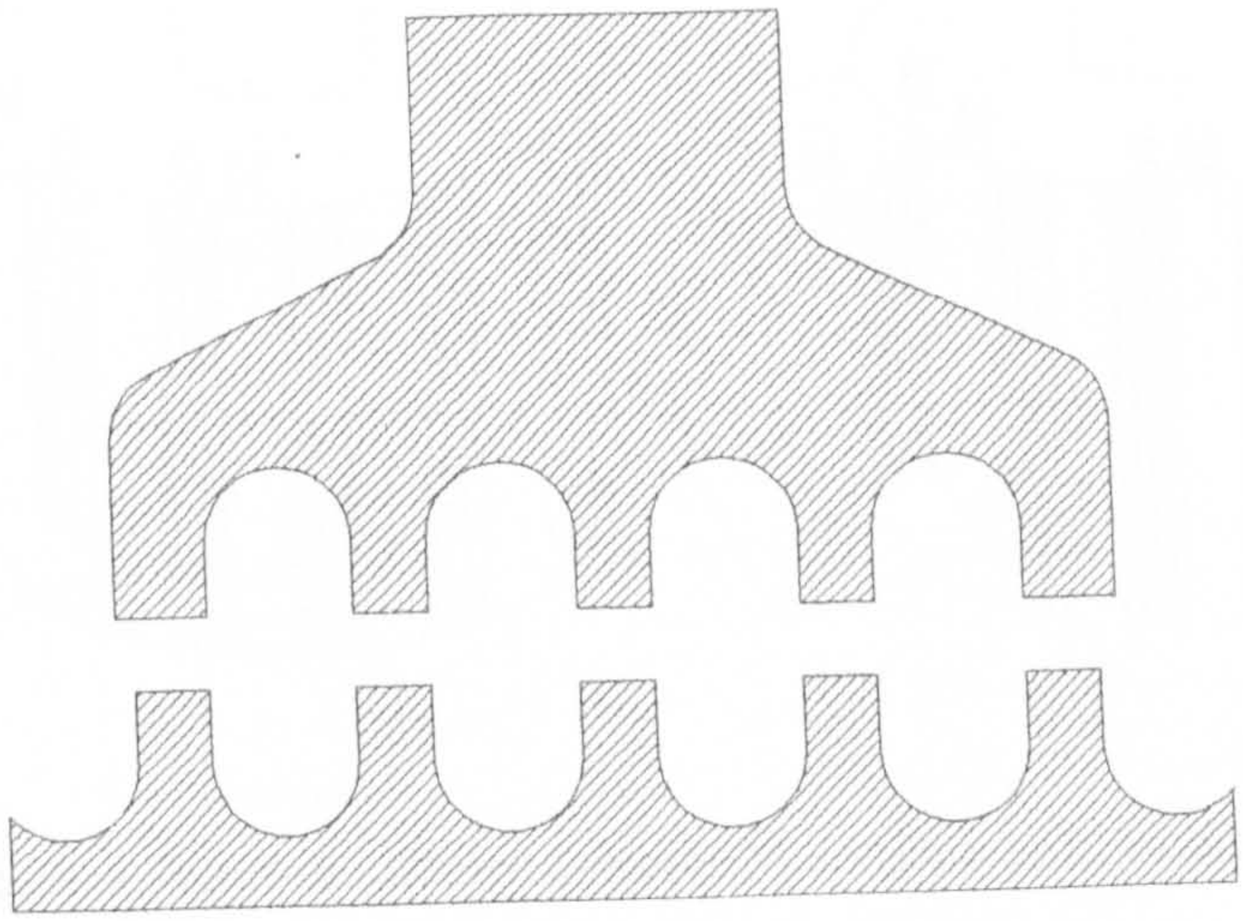


FIGURE 1.2 - Pole Castellations and Rotor Teeth

The simple structure in figure 1.1 will be used to briefly describe the motor's construction. It can be seen in the figure that the stator has eight salient poles, whereas the rotor has six. The rotor and stator are made up of thin pieces of sheet metal called laminations. Each stator pole has a coil wound around it and each opposite pole is connected to produce a reverse magnetic polarity. The example in figure 1.1 has four phases A, B, C, and D, made up of coils 1-1', 2-2', 3-3', and 4-4' respectively. Rotation is caused by applying current into the phases in sequence that causes two rotor teeth to line up with the excited stator poles. The energisation of the stator is then switched and the rotor is pulled to its next position.

1.2.2 Permanent Magnet Stepping Motor

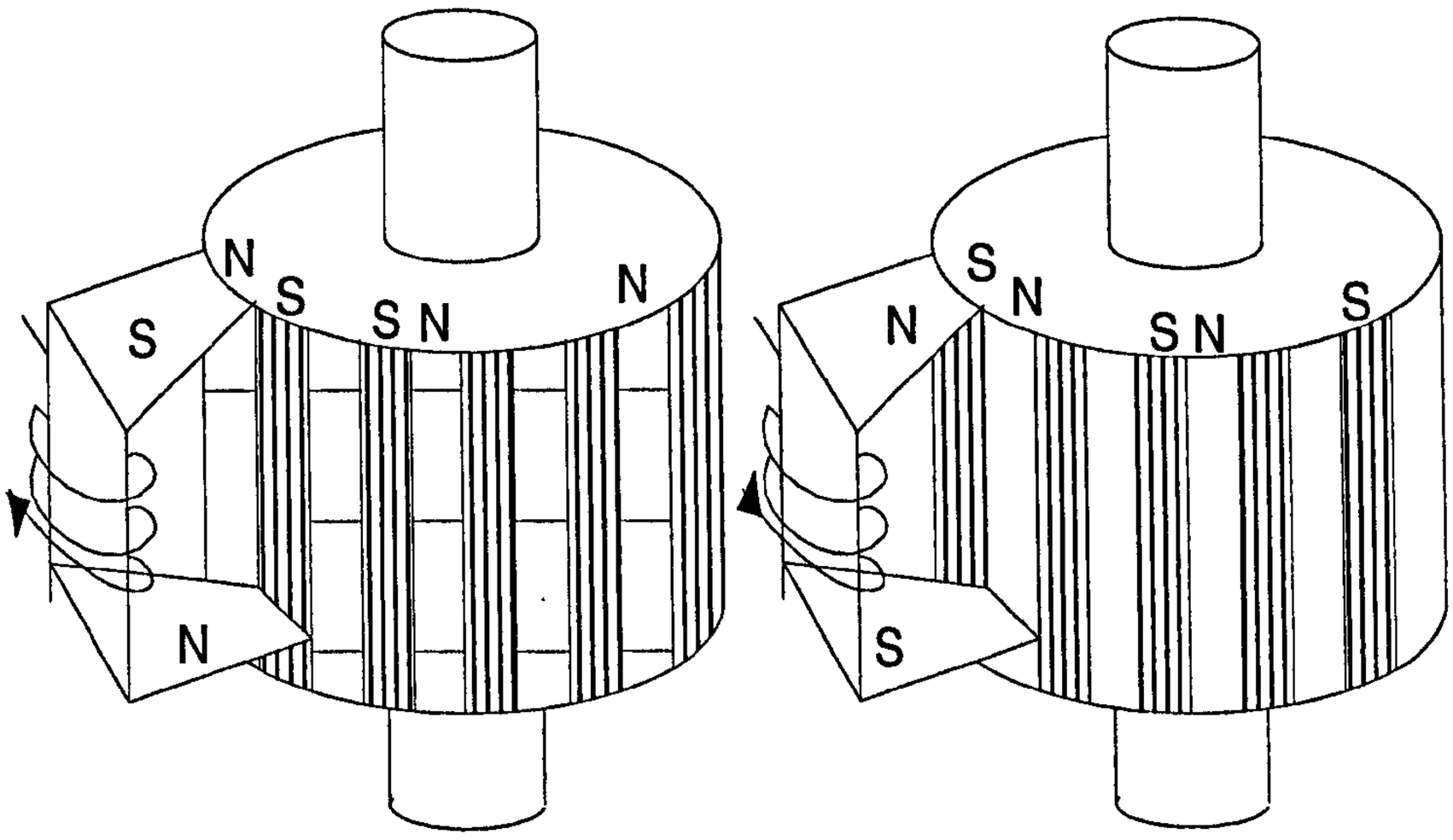


FIGURE 1.3 (A,B) - Permanent Magnet Stepping Motor, Change of Excitation in the Stator

Sometimes referred to as the tin can or can-stack motor, the construction of the permanent magnet stepping motor is shown in figure 1.3 (A,B). This motor is a low cost, low torque, low speed device, and is found commonly in computer peripherals. The motor steps in relatively large angles, but its simplicity allows mass production of a low cost device. The rotor is made of permanent magnet material and has individual magnetic poles magnetised radially on the rotor. The stator has a phase winding which can induce a two pole magnetic field. A rotor north pole aligns with the stator's south pole and a rotor south pole aligns with the north pole of the stator. The stator poles are made of two triangular sections that allow the top and bottom to line up with poles of opposite polarity on the rotor. Changing the coil current polarity in the stator coil causes a rotation of one rotor pole pitch as shown in figures 1.3(A) and (B).

1.2.3 Hybrid Stepping Motors

A hybrid stepping motor shown in figure 1.4, is by far the most widely used stepping motor in industrial applications.

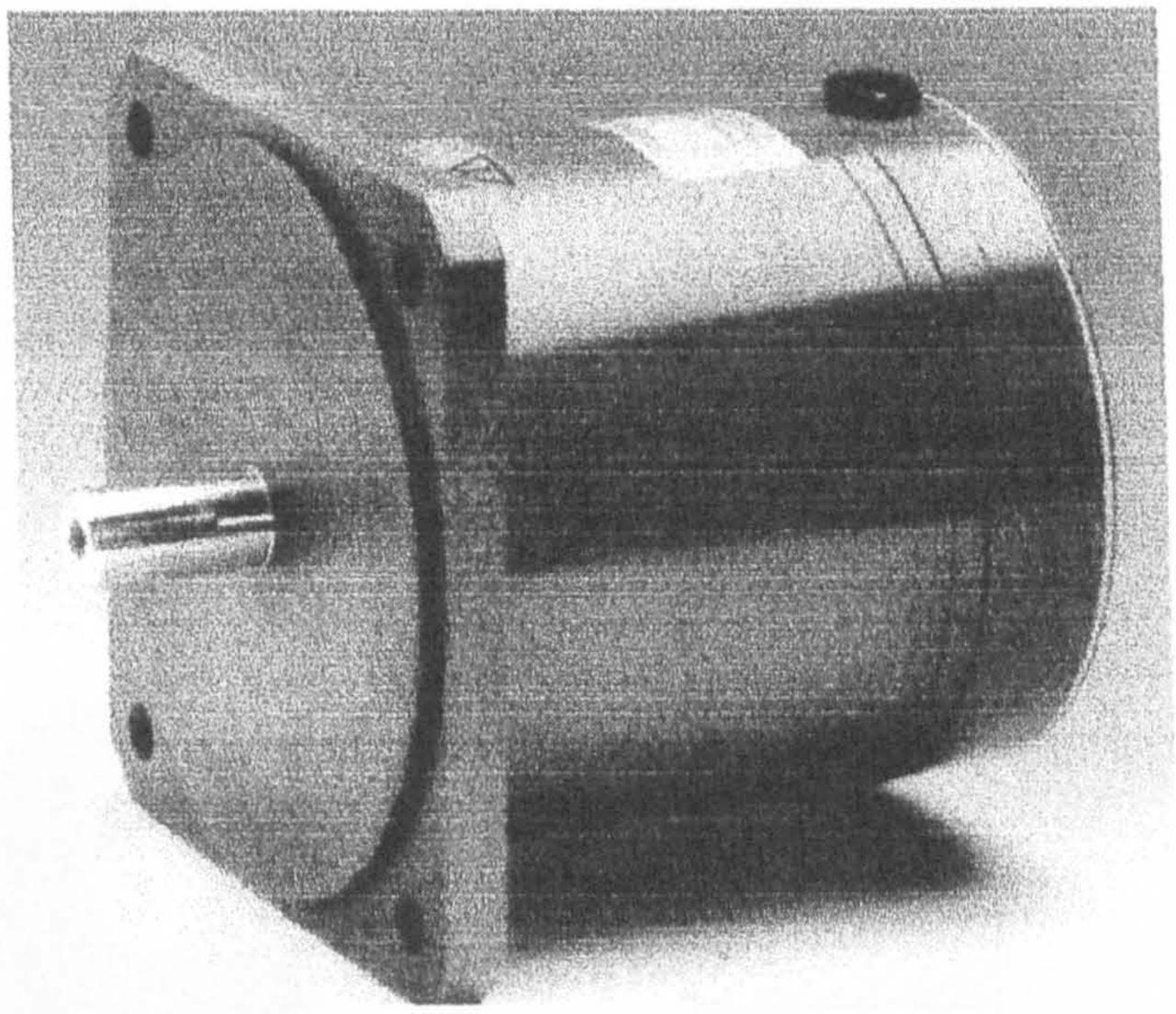


FIGURE 1.4 - Hybrid Stepping Motor *[Photograph Courtesy of Stebon Ltd.]*

The name is derived because it combines the operating principles of the permanent magnet motor and the variable reluctance type. The stator (figure 1.5) is on the periphery of the motor and is similar to the multi-tooth variable reluctance stator. In the most common implementation the laminations of the stator make up a set of eight electromagnetic poles that are wound in two phases to produce a four pole magnetic field. The windings around the stator poles are used to encourage or discourage the flow of magnetic flux through the poles depending on the rotor position required. As with all stepping motors, the function of the hybrid type is to move the rotor through a precise angular step when the current in one of the windings is switched. The stator poles produce separate latching points or steps for the motor.

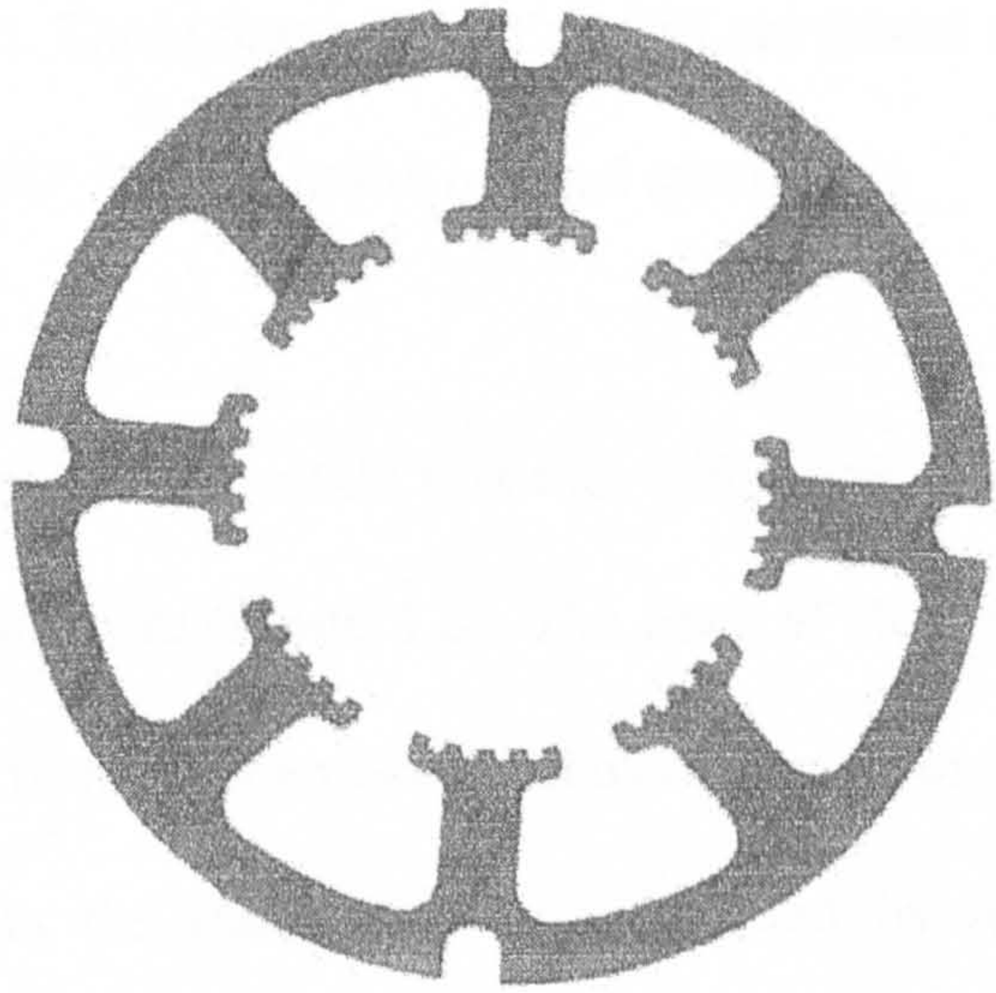


FIGURE 1.5 - Typical Hybrid Motor Stator

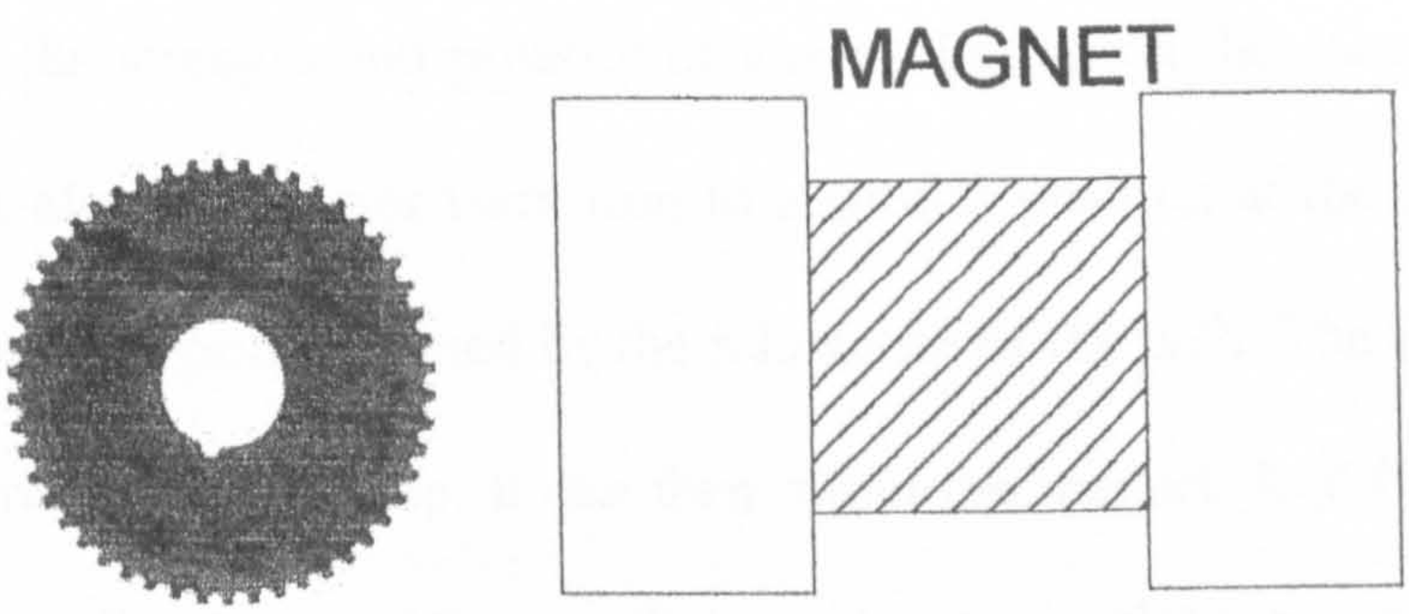


FIGURE 1.6 - Rotor make up of Hybrid Stepping Motor

The rotor (figure 1.6) of the hybrid motor is the important distinction between the hybrid and variable reluctance type stepping motor. A cylindrical magnet is sandwiched between two rotor end-caps. The magnet is axially magnetised causing one of the end-caps to act as a series of north poles and the other end-cap to be a set of south poles. The end-caps of the rotor are a series of steel laminations which are very gear like in appearance. The teeth on the two end-caps are mis-aligned with respect to each other by half a rotor tooth pitch. When the rotor teeth of one end-cap of the rotor are aligned with teeth of a particular stator pole, the teeth on the other rotor end-cap are mis-aligned with the teeth of the same

stator pole (figure 1.7). These rear end-cap's teeth are found to be aligned with another stator pole which is at 90° relative to the original stator pole.

The rotor's magnetic field will generate flux that will travel from the end of the magnet to another through the stator core (figure 1.8). The lines of flux will leave the north pole of the magnet through the front rotor end-cap and travel across the air-gap into the stator. The position the flux enters the stator will be influenced by the lowest reluctance path available. This lowest reluctance path will depend on how much the rotor and stator teeth are aligned, and the strength and polarity of any excitation on the stator coils. The flux path then travels along the stator back iron to a similar position at the rear of the stator. The actual position is again governed by the reluctance of the path. The flux will cross the air-gap into the rear rotor end-cap. It can then rejoin the magnet. In different parts of the motor the magnetic flux has significant radial, axial or tangential components.

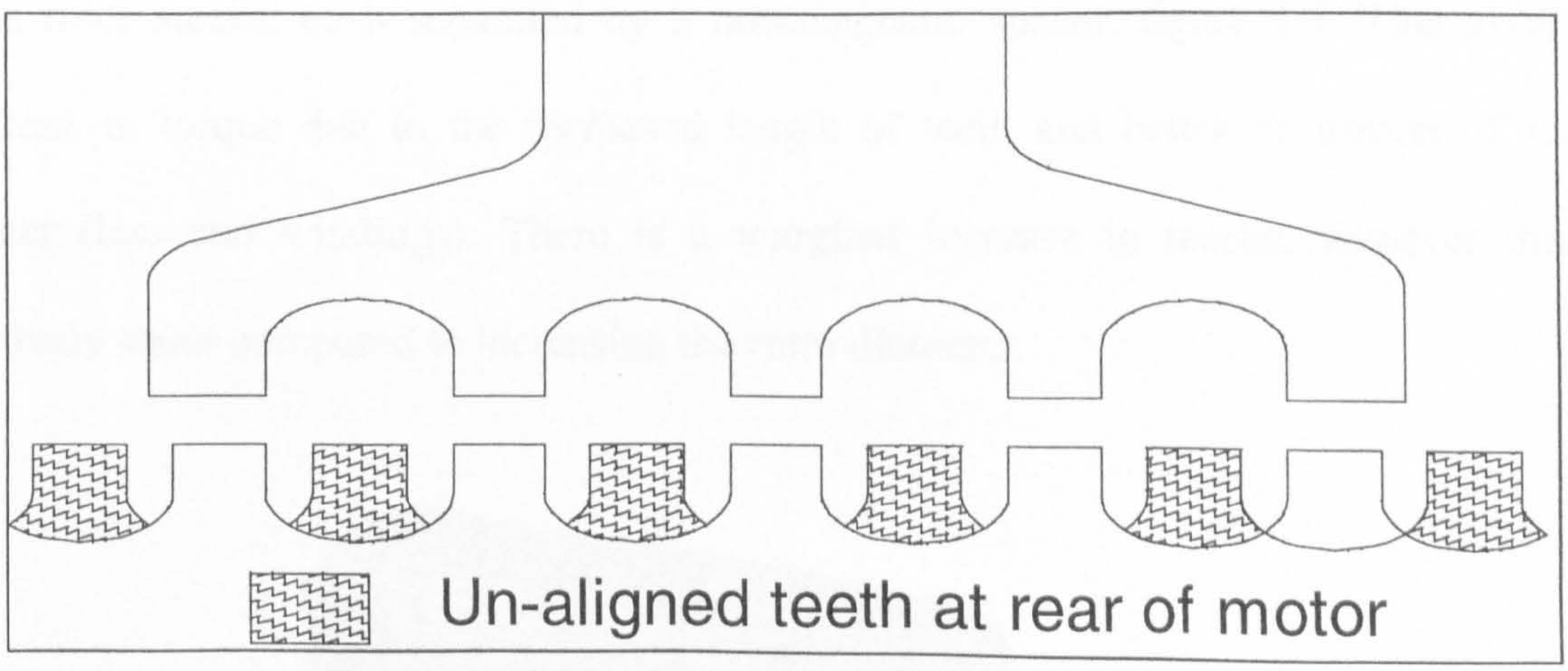


FIGURE 1.7 - Pole Castelations and Rotor

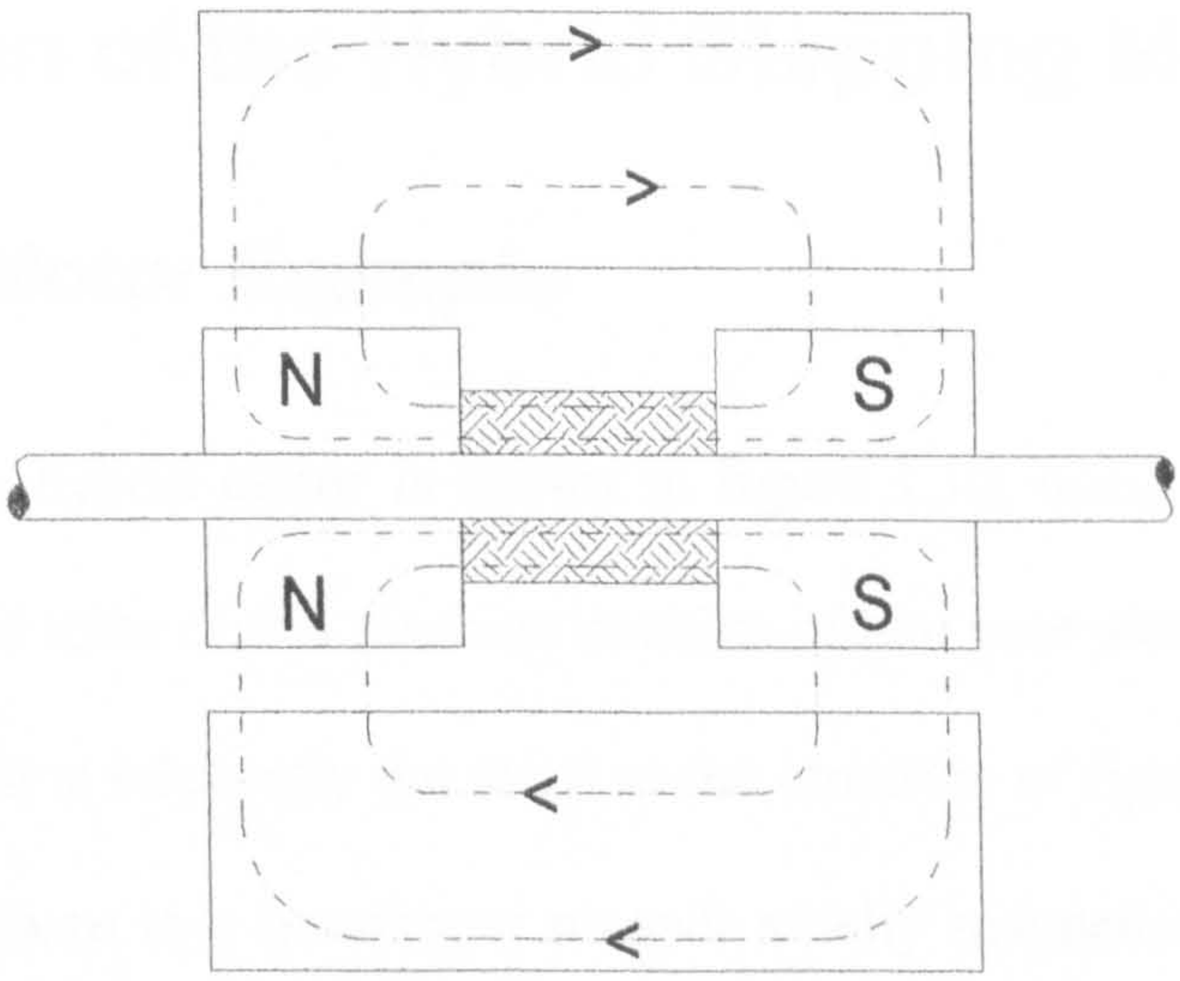


FIGURE 1.8 - Cross Sectional View of Magnetic Flux Path Containing Axial and Radial Components (Tangential effects are not shown).

To increase the torque output for a particular frame size diameter, multi-stacking can be employed. The construction of each stack of a multi-stack motor is essentially the same as for a single stack. The stator is longer to cover the extension of the rotor. Instead of the rotor consisting of a single magnet and two end-caps, the rotor now has two or more of these rotor stacks, each separated by a non-magnetic spacer, figure 1.9. This gives an increase in torque due to the increased length of teeth and better utilisation of stator copper (less end windings). There is a marginal increase in inertia, however this is relatively small compared to increasing the rotor diameter.

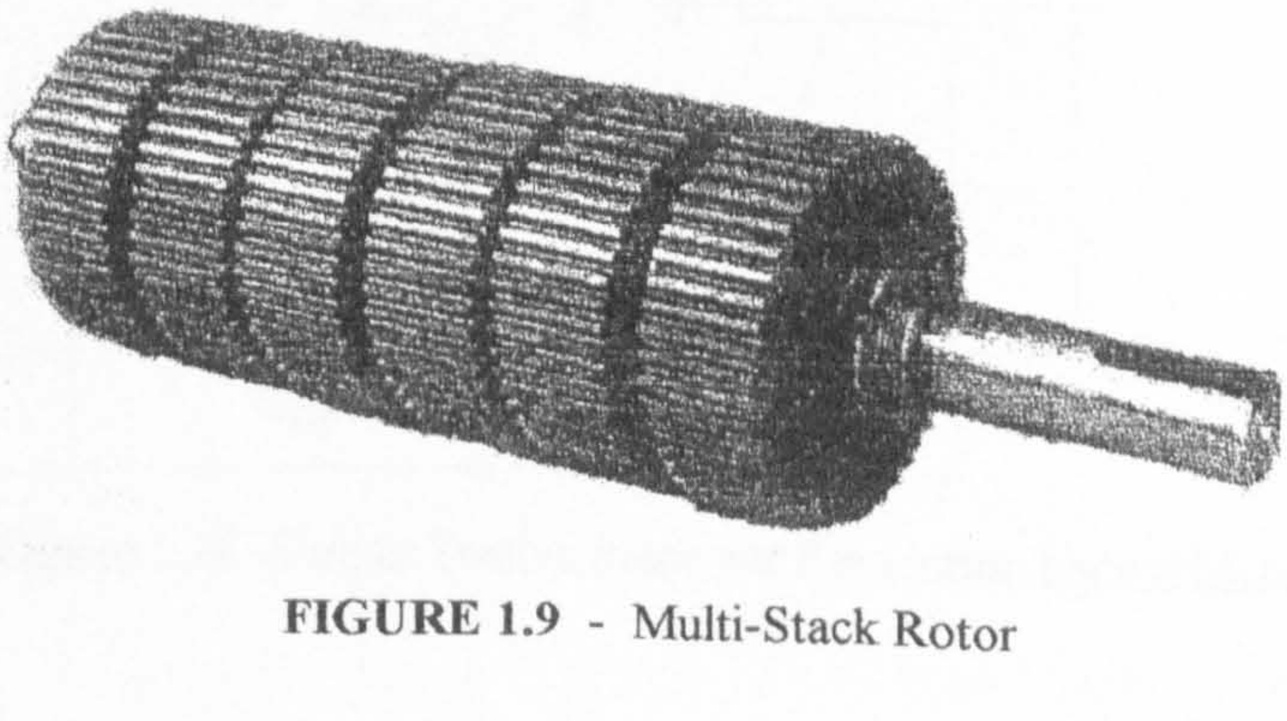


FIGURE 1.9 - Multi-Stack Rotor

1.3 Operation of the Hybrid Stepping Motor

1.3.1 Simple Motor Example

The operation of the hybrid motor is shown in figure 1.10, using a simple 12 step per revolution motor. The rotor of this machine consists of two pole pieces with three teeth on each, the construction is inherently the same as the structure of figure 1.4. In between the two rotor end-caps there is a permanent magnet axially magnetised, producing a north pole at the back and a south at the front. The stator has four poles with coils wound on each. Phase 1 is made of poles 1A and 1B, and Phase 2 is made of 2A and 2B.

With no current flowing the rotor will take one of a possible 12 positions. This is because of the detent effect of the magnet trying to minimise the reluctance to its own magnetic flux path. This occurs when a pair of north and south rotor teeth align with two of the stator poles.

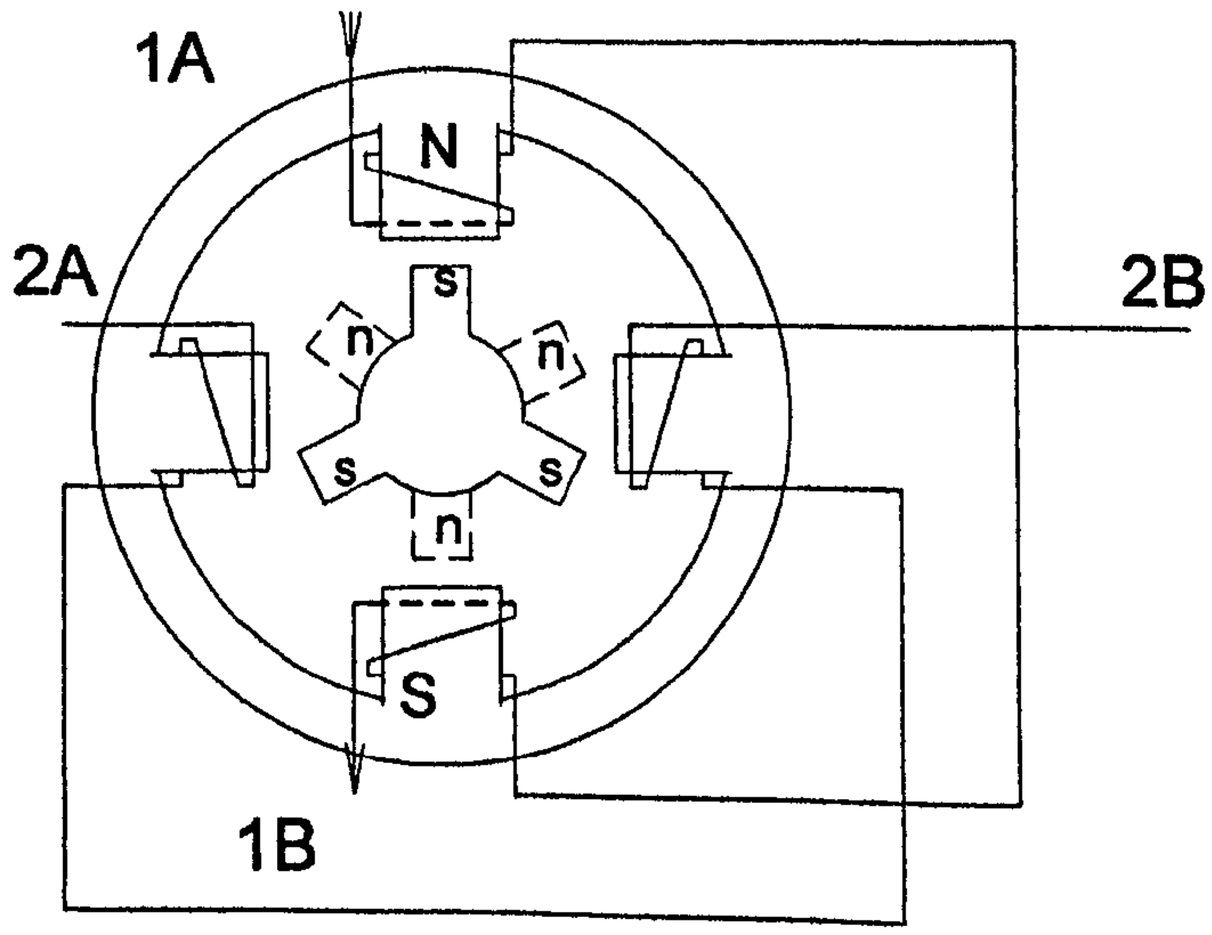


Figure 1.10 -Simple Twelve Steps per Revolution Hybrid Motor

i Single Phase Full Stepping

If current now passes through one of the stator phase windings as shown in figure 1.11A, the resultant north and south stator poles will attract teeth of the opposite polarity on each end of the rotor stack. There are now three stable positions for the rotor, the same as the number of teeth on each rotor end-cap. The torque required to deflect the rotor from its stable position is now much greater, and is referred to as the holding torque.

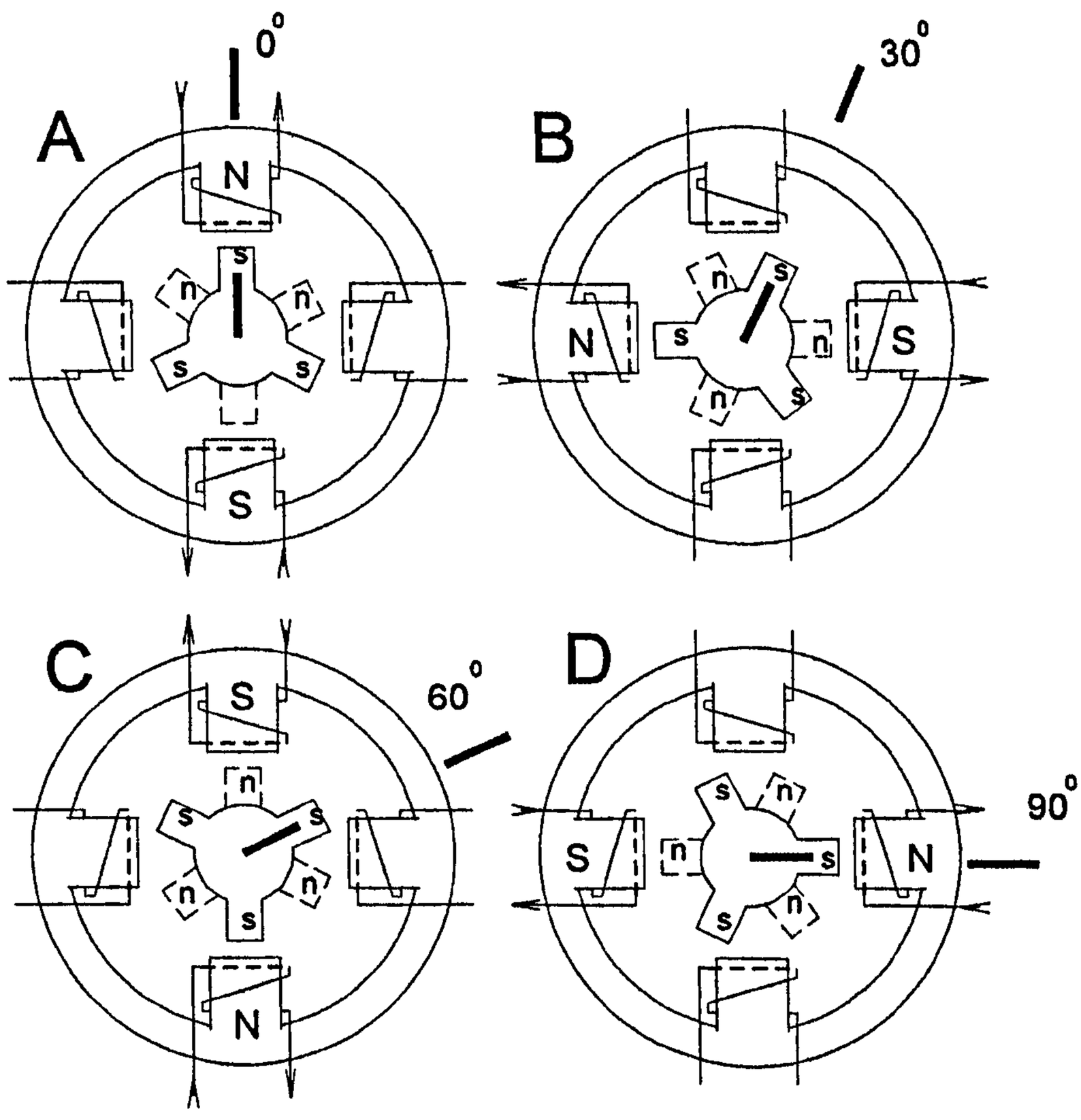


FIGURE 1.11(A,B,C,D) - Full Stepping, Single Phase

By changing the current flow from the first to the second set of stator windings (B), the stator field rotates through 90° and attracts a new pair of rotor poles. This results in a rotor step of 30° referred to as one full step. Applying current in phase 1 again but in the

opposite polarity, will cause the field to shift by another 90° and the rotor takes another full step of 30°, figure 1.11.C. If phase two is energised with a negative polarity current figure 1.11.D position will occur, then the condition in figure 1.11.A can be repeated. In completing these four steps the rotor is said to have moved through one tooth pitch. This simple motor has therefore a 12 steps per revolution in single phase full stepping mode.

ii Two Phase Full Stepping

If two coils are energised simultaneously (figure 1.12), the motor is said to be operating under two phase excitation. The rotor has now an ability to take up intermediate positions, since it can be equally attracted to two stator poles. Greater torque can be achieved in this way as all the stator poles are influencing the rotor. The motor can be made to take a full step by reversing the current in one of the phases, producing a 90° movement in the stator field. This is the normal method of two phase full stepping.

1.3.1.3 Two Phase Half Stepping

By alternately energising one phase and then two (figure 1.13), the rotor moves through only 15° at each stage and the number of steps per revolution will be doubled. This is called half stepping, and many industrial applications make much use of this mode. There is some loss of torque as all the windings are not in constant use, but motion is smoother at low speeds, and there is less positional overshoot.

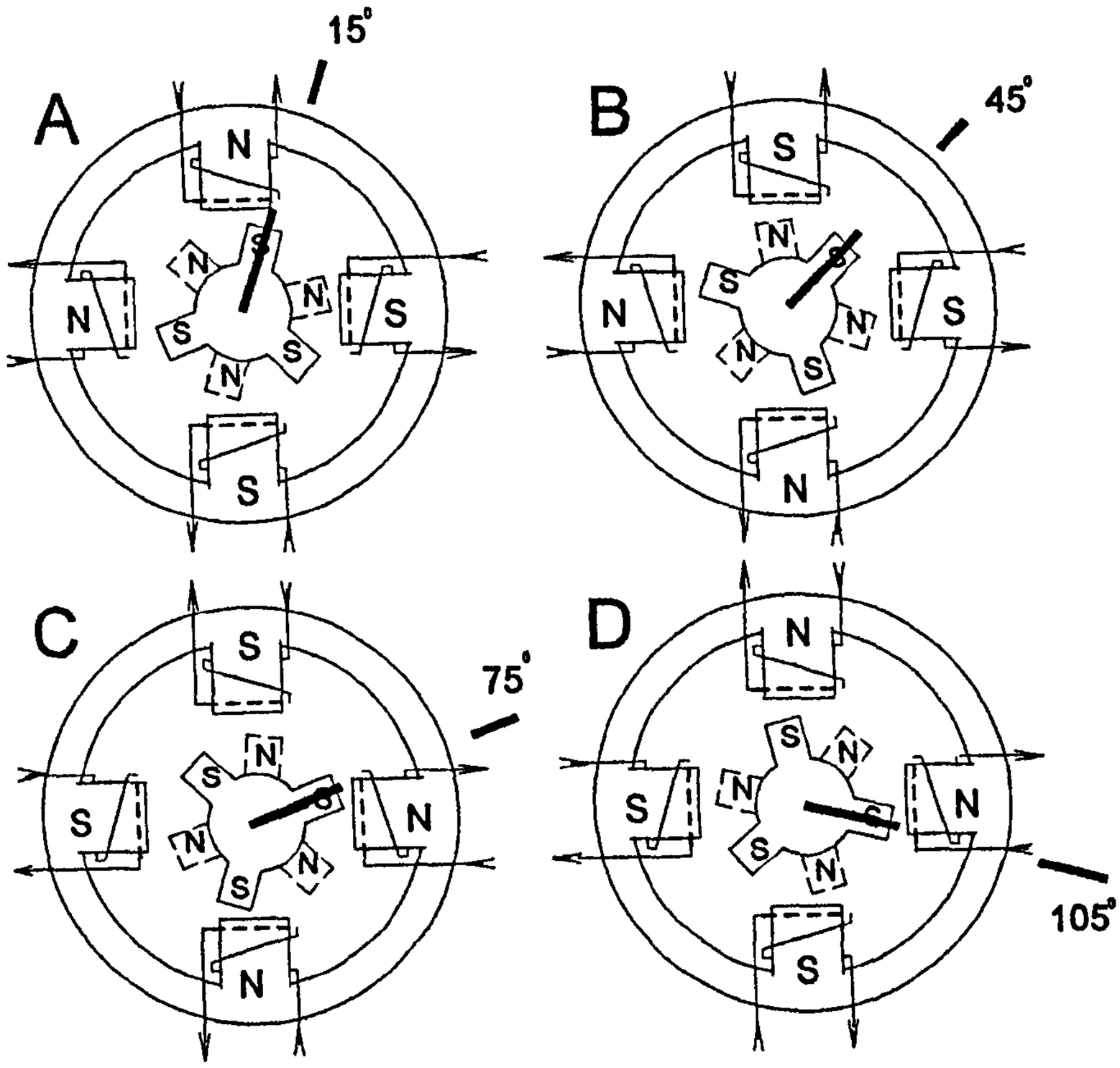


FIGURE 1.12 - Full Stepping, Two Phases.

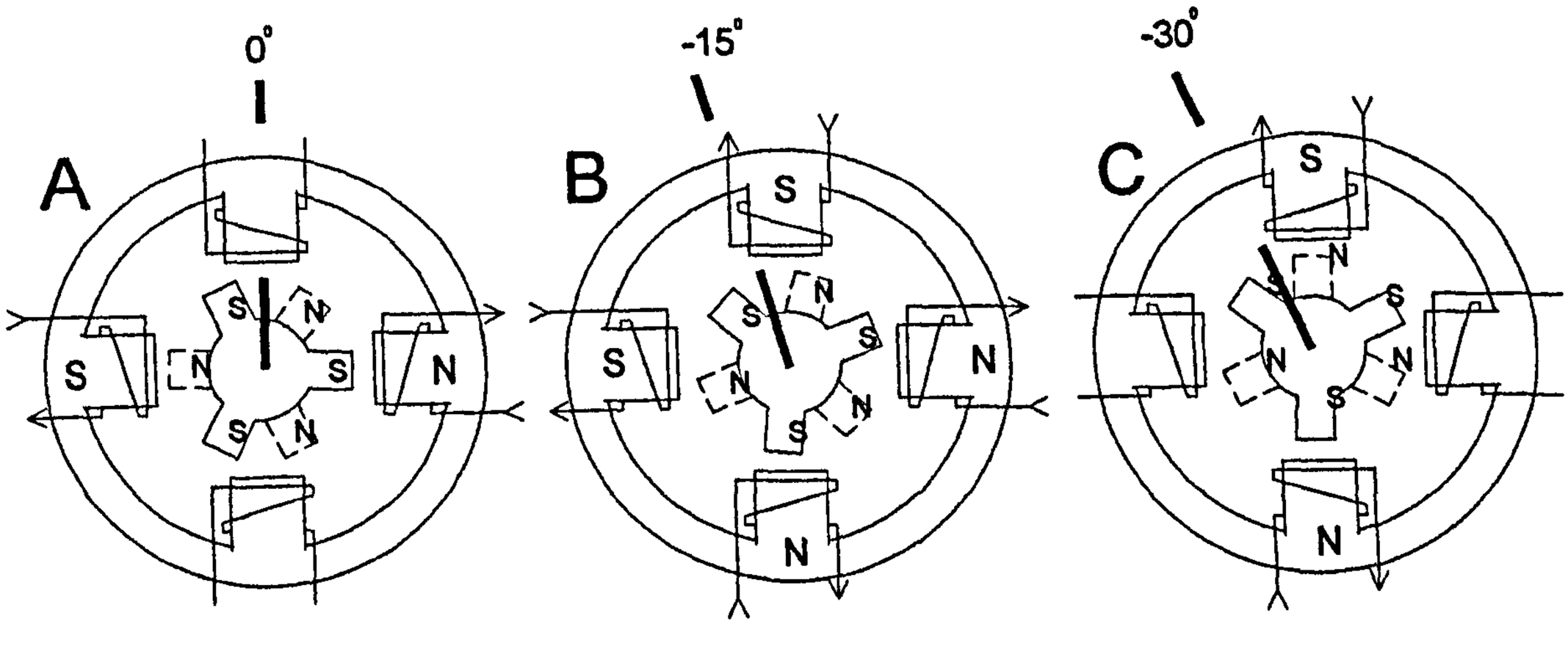


FIGURE 1.13 - Half Stepping

1.3.2 Current Patterns

When the motor is driven in its full step mode by energising two phases at a time (figure 1.14), the torque available to each step will be the same. In half step mode, two phases are energised alternately and then only one (figure 1.15). This will cause greater torque to be produced when the two phases are on. The alternate steps will therefore be strong (two phase on) and then weak (single phase on). The available torque is limited by the weaker step, even though the motion will be smoother. It would be beneficial to have equal torque on each step. A different current sequence can be used, which increases the phase current when only one phase is on. This will not burn out the winding, as motors are generally rated for two phases on. With one phase on the same total power will be dissipated in the windings if the current is increased by 40%. Using higher current when the single phase is on produces approximately equal torque on alternate steps, figure 1.16.

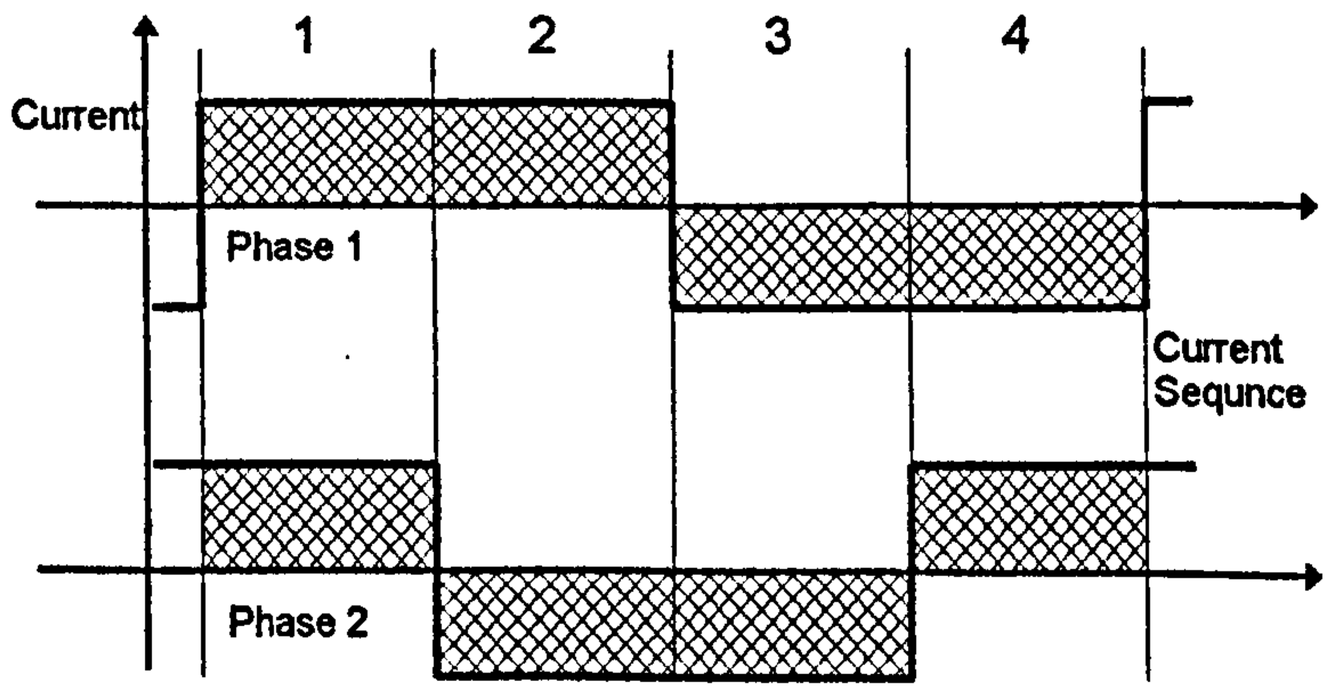


FIGURE 1.14 - Full Stepping - Two Phase

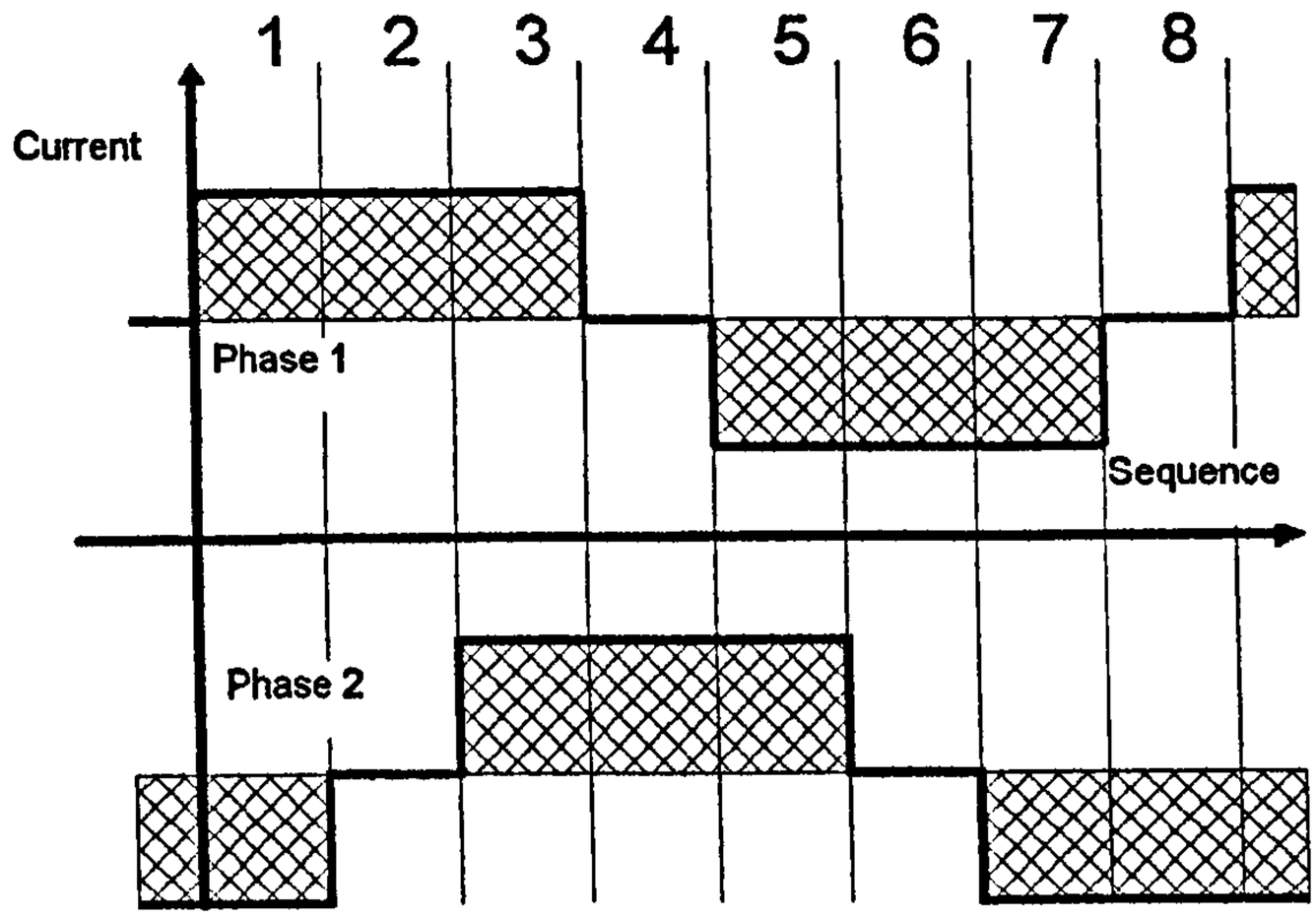


FIGURE 1.15 - Half Step Current

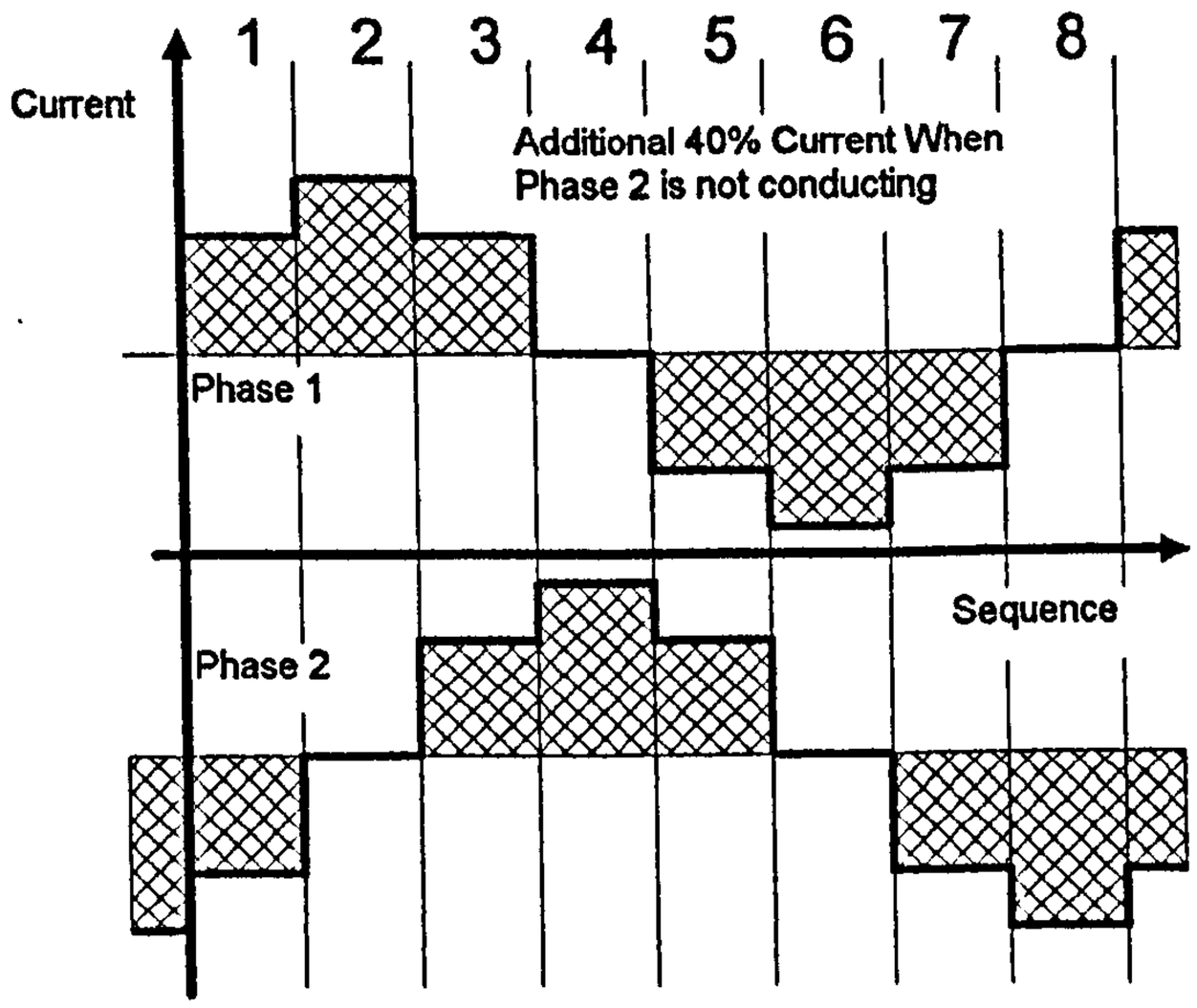


FIGURE 1.16 - Profiled Half Step Current

1.3.3 Microstepping.

Energising two phases with equal current produces an intermediate step position half-way between the one-phase on positions. If the two phase currents are unequal, the rotor will be shifted towards the pole with higher energisation. This effect is utilised in the microstepping drive, which subdivides the basic motor step by proportioning the current in two windings. In this way, the step size is reduced and smooth rotation is obtained. The

current pattern in the windings resembles two sine waves with a 90° phase shift between them as in figure 1.17.

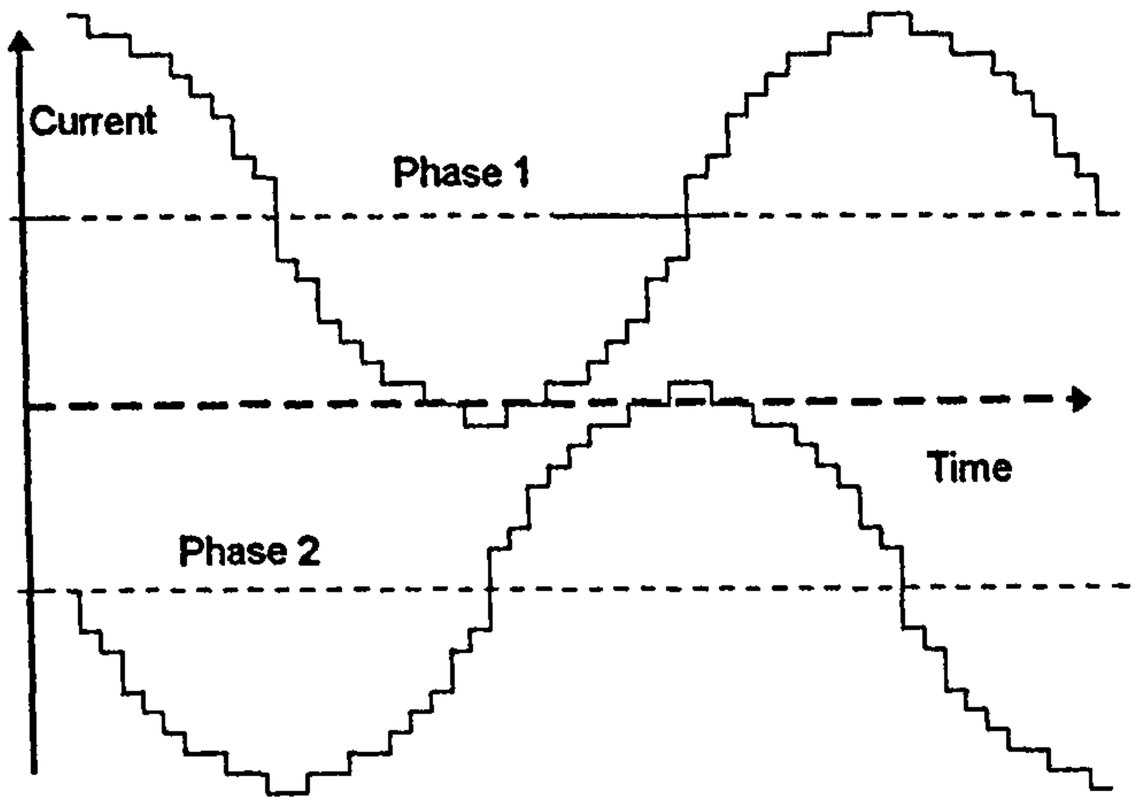


FIGURE 1.17 - Microstepping

1.4 Standard 200-Step Hybrid Stepping Motor

The common 200 step hybrid stepping motor described in section 1.2.3, operates in a similar way to the simple 12 step model, but with 50 teeth on each rotor end-cap, and multiple teeth on each stator pole. It produces a step angle of 1.8° (200 steps) in two phase on mode, equation 1.1, and 0.9° (400 steps) with half stepping.

$$S = 2mn_r, \qquad (1.1)$$

where S is the steps per revolution,

m is the number of phases, and

n_r is the number of teeth on the rotor stack.

There are as many detent positions as full steps per revolution of the motor. The hybrid stepping motor may also be driven direct from a single phase supply using a capacitor in

series with one of the phases (figure 1.18). In the UK the 50 Hz supply will produce a rotation of 60 rpm, equation 1.2.

$$\omega_{rpm} = f\frac{60}{n_r}, \qquad (1.2)$$

Where ω_{rpm} is the speed in revolutions per minute, and

f is the frequency of the supply.

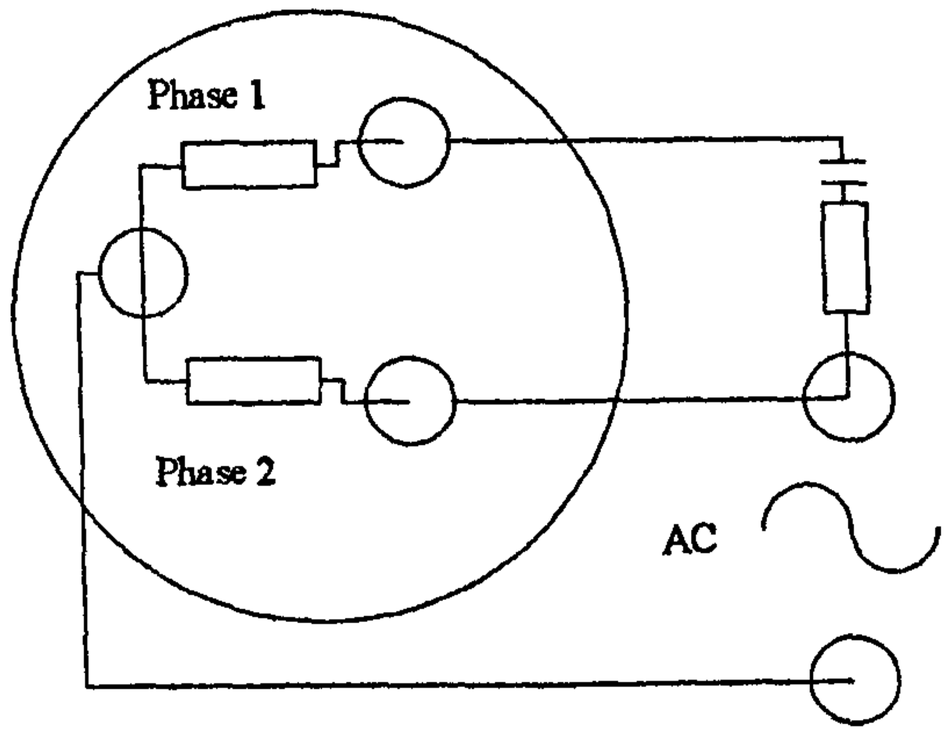


FIGURE 1.18 - AC Synchronous Motor Circuit

1.4.1 Windings

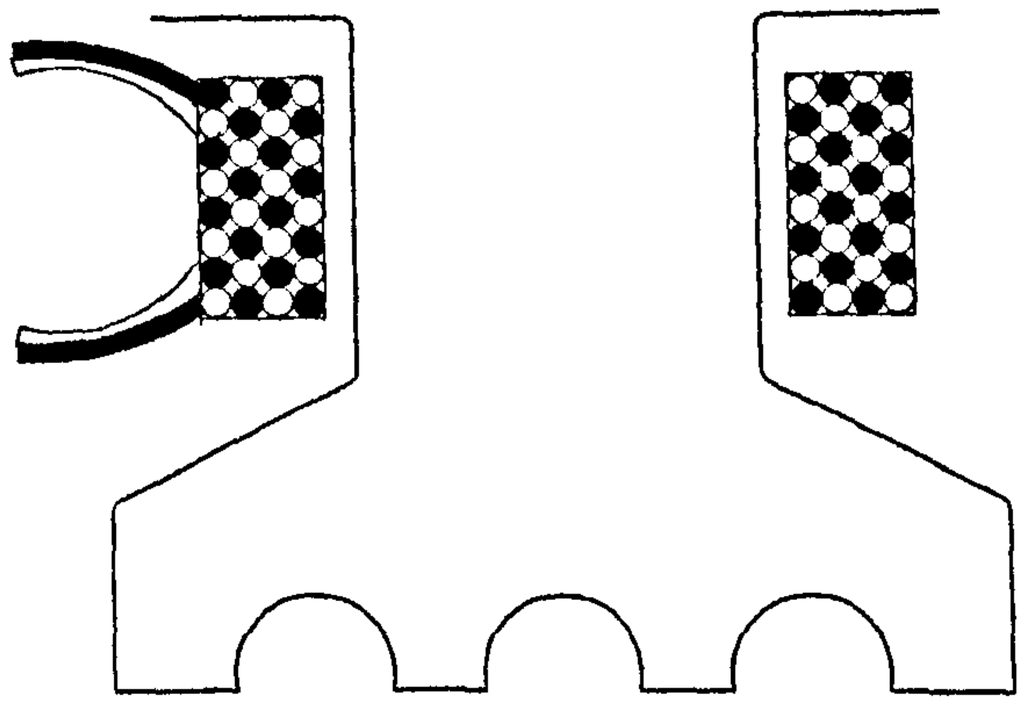


FIGURE 1.19 - Bifilar-Winding

Traditionally motors have had bifilar windings, which means there are two sets of identical windings on each pole, as shown in figure 1.19. These are wound as if they were single coils, producing two coils that are almost electrically and magnetically identical.

The origins of this type of winding go back to the use of the unipolar drive, which reverses the polarity of the magnetic field by energising the other coil on the stator pole instead of reversing the current. This allowed a simpler drive to be constructed. Today's drive technology has improved and as a result the bipolar drive has nearly eliminated the unipolar drive.

Other types of wire are exclusively used with bipolar drives. Each of the eight stator poles has a single coil placed on it. Poles 1,3,5, and 7 (figure 1.20) have coils of one phase, whereas the remainder have the second phase. Each consecutive coil of a phase has an alternate polarity. The coils on poles 1 and 3, and 5 and 7, are connected internally in pairs in parallel or series. This allows 4 wires to be brought out per phase.

Both the bifilar and exclusively bipolar winding allows two types of speed/ torque characteristics to be obtained by the use of the 8 leads that come from the two phases. These are referred to as parallel and series connections. Connecting the winding in series causes a phase current to flow through twice as many turns. This doubles the ampere turns or MMF and causes a corresponding increase in torque. However this doubles the effective number of turns and increases the inductance by a factor of 4 relative to a single winding. Inductance L is proportional to the number of turns N squared.

$$L \propto N^2, \tag{1.3}$$

This causes the torque to drop off as the motor speed is increased, as the peak current level in the winding will fall. Series connections are used when low speed torque is required and may well produce low speed resonance because of the high torque produced in this region. Connecting in parallel allows the current to divide between the two coils. It does not change the inductance (relative to one of the coils) as the number of effective ampere turns remains the same. The resistance of the coils is however halved, which produces a lower power dissipation. The current will be able to run 40% higher than in a single winding. Parallel connections are generally preferred as they produce a flatter torque curve and greater shaft power. Typical series and parallel performance curves are shown in figure 1.21.

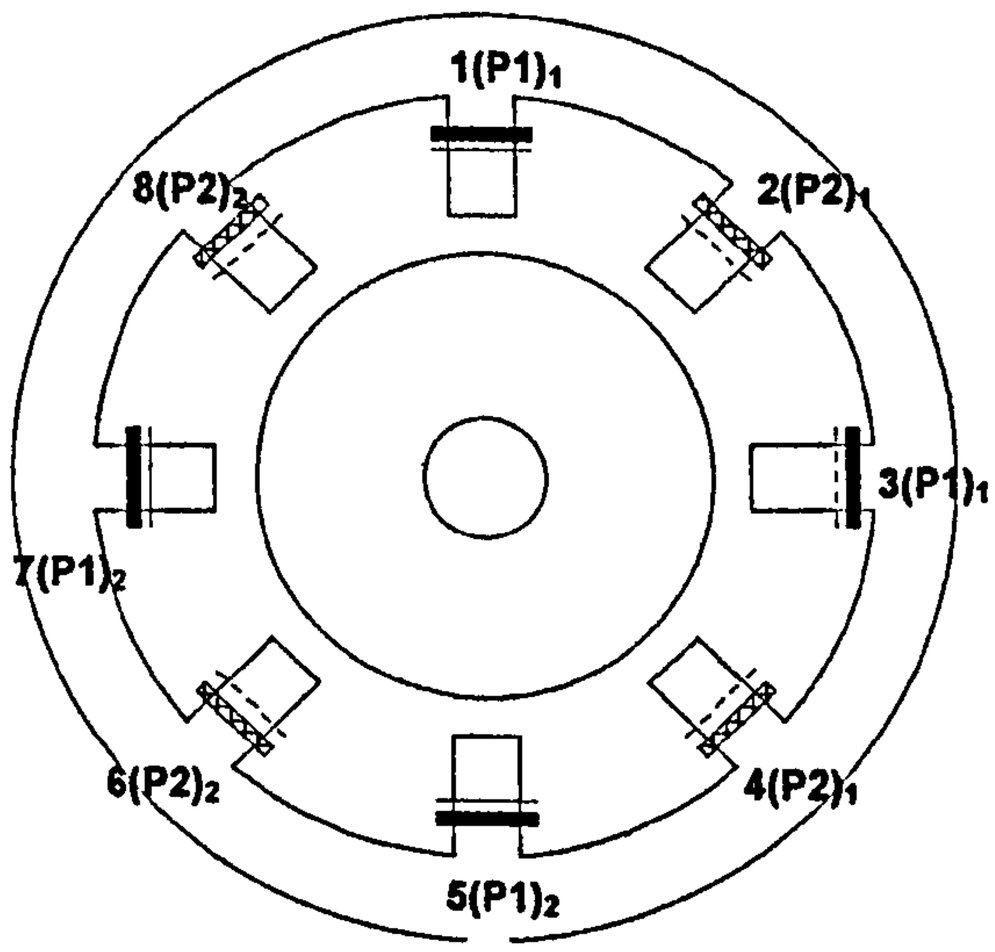


FIGURE 1.20 - Bipolar Exclusive Winding

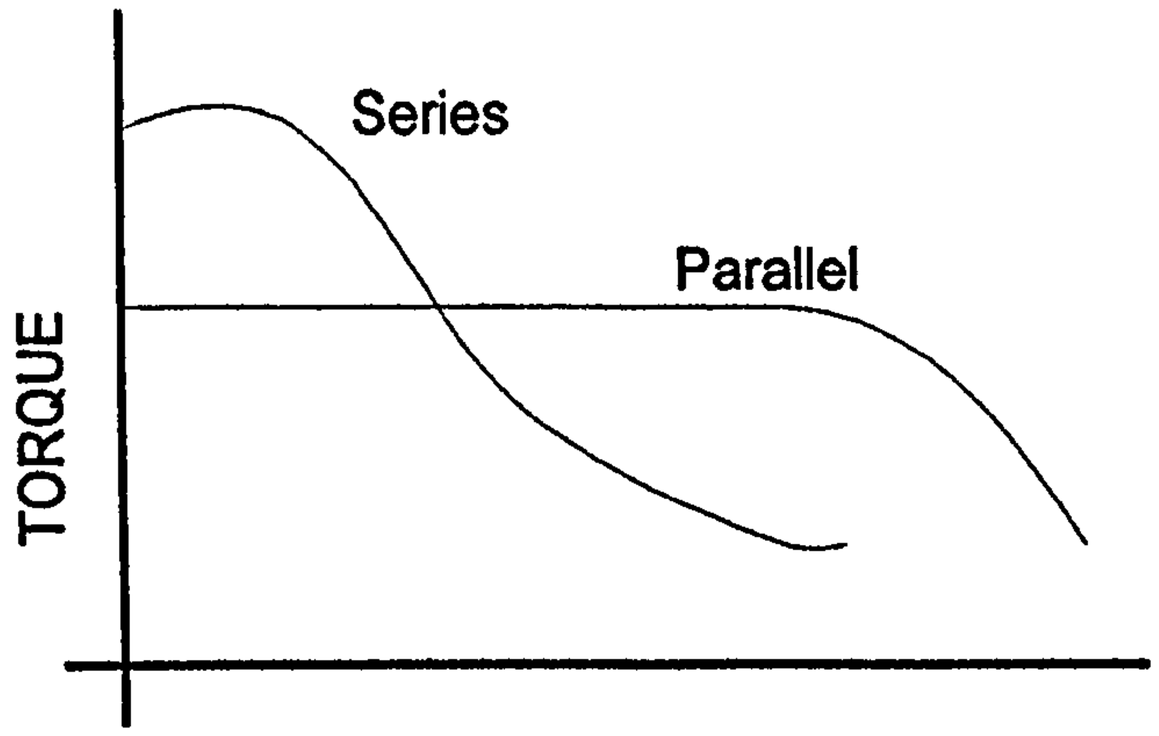


FIGURE 1.21 - Typical performance curves for Series and Parallel Connections

1.5 Stepping motor drives.

The stepping motor drive delivers the electrical power to the motor in response to a low level signal sent from a controller. The applied voltage is significant in controlling the current and has a dramatic effect on the speed range of the motor. Input signals to the drive consist of step pulses and direction information; One pulse for every step the motor is to take. This is true regardless of the motor resolution chosen. The motor will therefore require 200 pulses for full stepping or 50,000 pulses to produce a revolution for a 50,000 high resolution drive. Speed control is achieved by the rate at which the digital pulse train is fed to the drive. The rotor is expected to increment steps at the same rate as the signal.

1.5.1 Unipolar Drive

The simplest type of drive is the unipolar drive as shown in figure 1.22. Current in the drive can only flow in one direction. To reverse the magnetic field of a stator pole, the drive must operate with bifilar wound coils where the excitation is switched to the second coil. Generally in this simple drive the current is determined by the motor phase resistance

and the applied voltage. This drive responds well at low speeds, but torque rapidly decreases as speed increases due to the inductance of the phase windings.

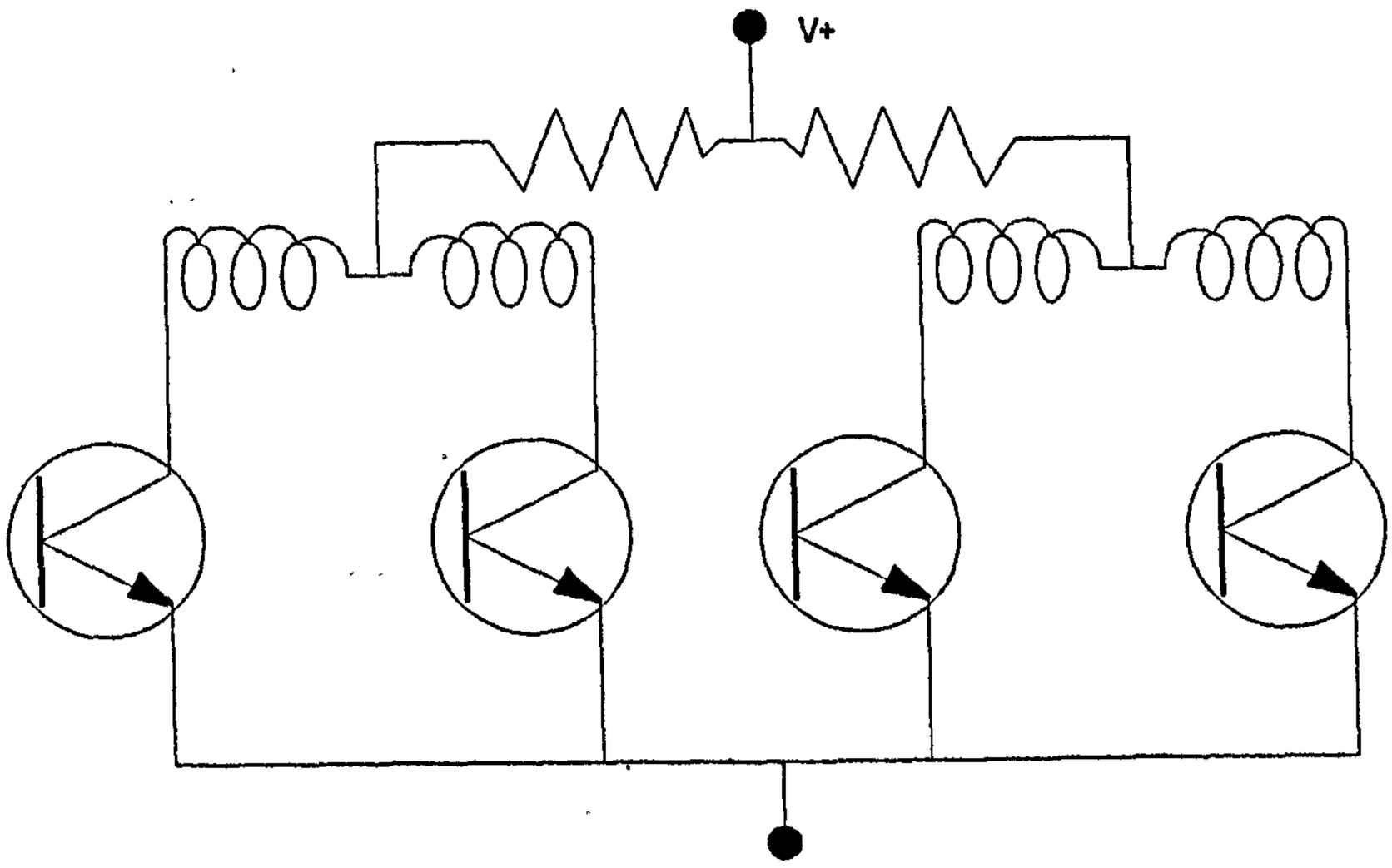


FIGURE 1.22 - Basic Unipolar Drive[3]

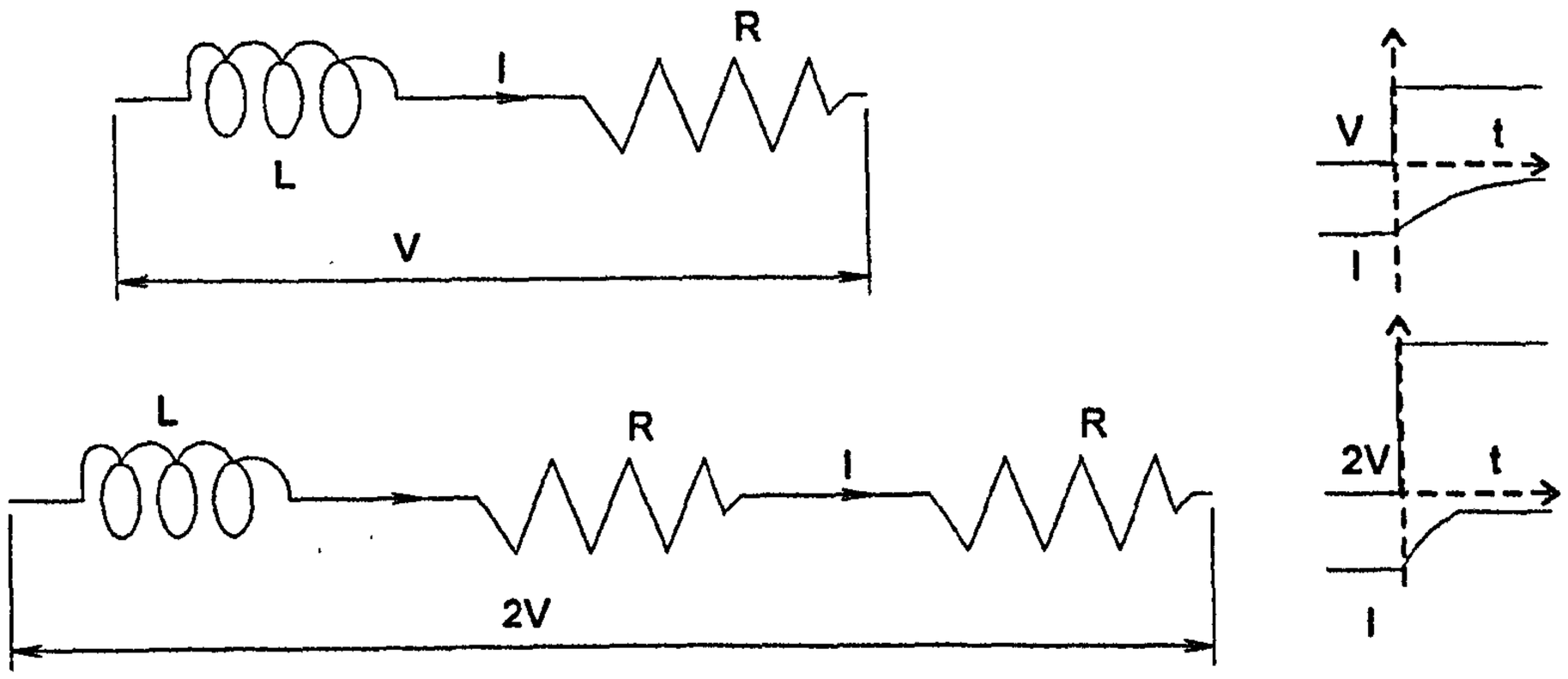


FIGURE 1.23 - R-L Drive Principle of Reducing Current Rise Time[3]

1.5.2 R-L Drives

In the unipolar drive the current build up is determined by the voltage, winding resistance, and winding inductance, figure 1.23. To increase the build speed of the current the voltage

would need to be increased. The final current reading would also be higher. By adding a resistor in series the current level can be limited. The electrical time constant $t_{constant}$ can be calculated by;

$$t_{constant} = \frac{L}{R} \quad , \tag{1.4}$$

Where R is the resistance.

Adding an additional resistor will cause increased power to be dissipated in the series resistance, producing a significant amount of heat. This extra power must come from the DC power supply.

1.5.3 Bipolar Drives

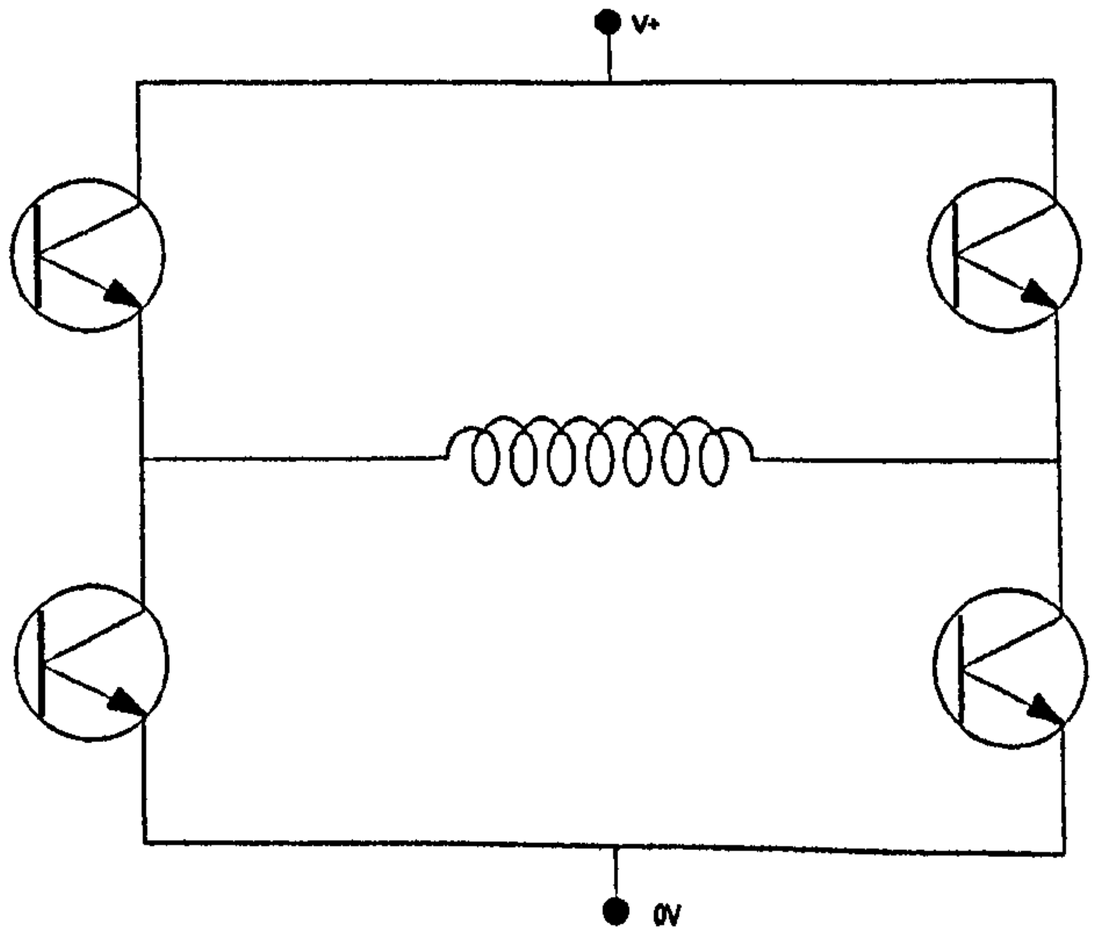


FIGURE 1.24 - Bipolar Bridge [3]

One drawback of the unipolar drive is its inability to utilise all the coils on the motor's stator poles. At any one time only half the coils on a pole are being used. If both coils

were being used there would be a 40% increase in amp turns for the same power dissipation. To achieve this higher performance a bipolar drive is used. One phase of this type of drive is shown in a simple form in figure 1.24, and can drive current in either direction through each motor phase. The standard arrangement used is called the bridge system, which uses four transistors per phase winding.

1.5.4 Power Dumping

Because the motor has a permanent magnet, the stepping motor can act as a generator if the shaft is driven by an external source. Therefore care must be taken when the energy associated with the inertia of the load is returned to the drive during retardation. A large increase in supply voltage may damage the drive transistors and a power dump circuit must be incorporated to dissipate the regenerated power [3].

1.6 Stepping Motor Control Systems

1.6.1 Example of Open Loop Control (Figure 1.25)

The operator interface allows the user to input some required characteristic of the system, such as speed, acceleration, and distance. The interface may be simply a thumb wheel control or a complicated computerised serial interface. The information is passed to the controller that converts the information into the pulse and direction signal that are fed to the drive. Conditioning and selection of the resolution may be applied to the signal from the information given by the user or may be pre-set in hardware or software options in the controller. The drive will use a form of pulse width modulation to control the current

delivered to the motor windings. The rotor will rotate in a condition dependent on the sequence and style of the energising of its phase windings. The current delivered at a speed is not load dependent. If the motor misses a step or stalls there is no feedback to the controller that this has happened.

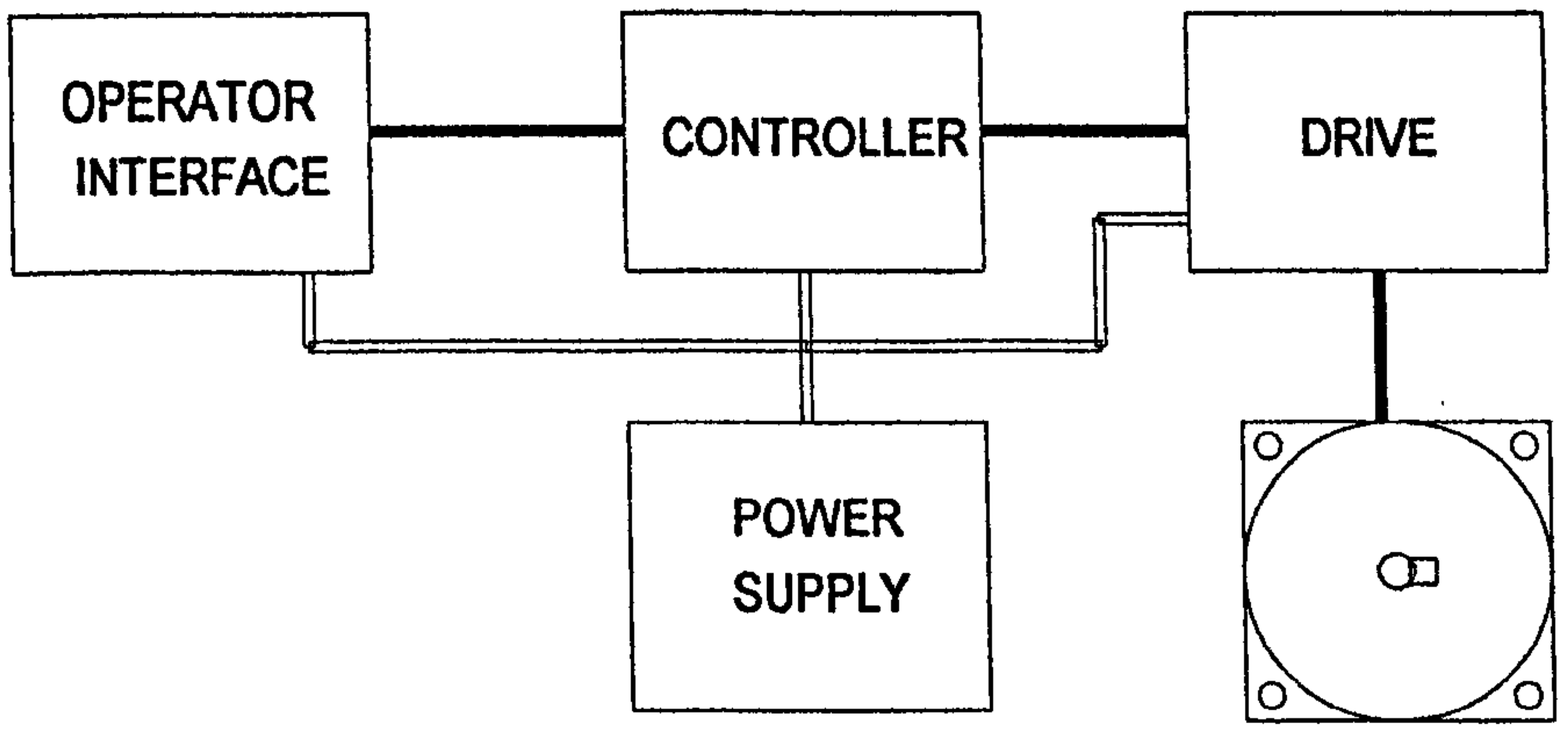


FIGURE 1.25 - Open Loop Motor/ Drive System

1.6.2 Example of Closed Loop Control (figure 1.26).

In addition to the function blocks of the open loop system, this system incorporates a feedback signal to the controller. Typically an encoder or resolver is used. In their simplest form provide information on position and step integrity. However they can be used to provide information for phase commutation and precise current control to achieve servo performance [4,5].

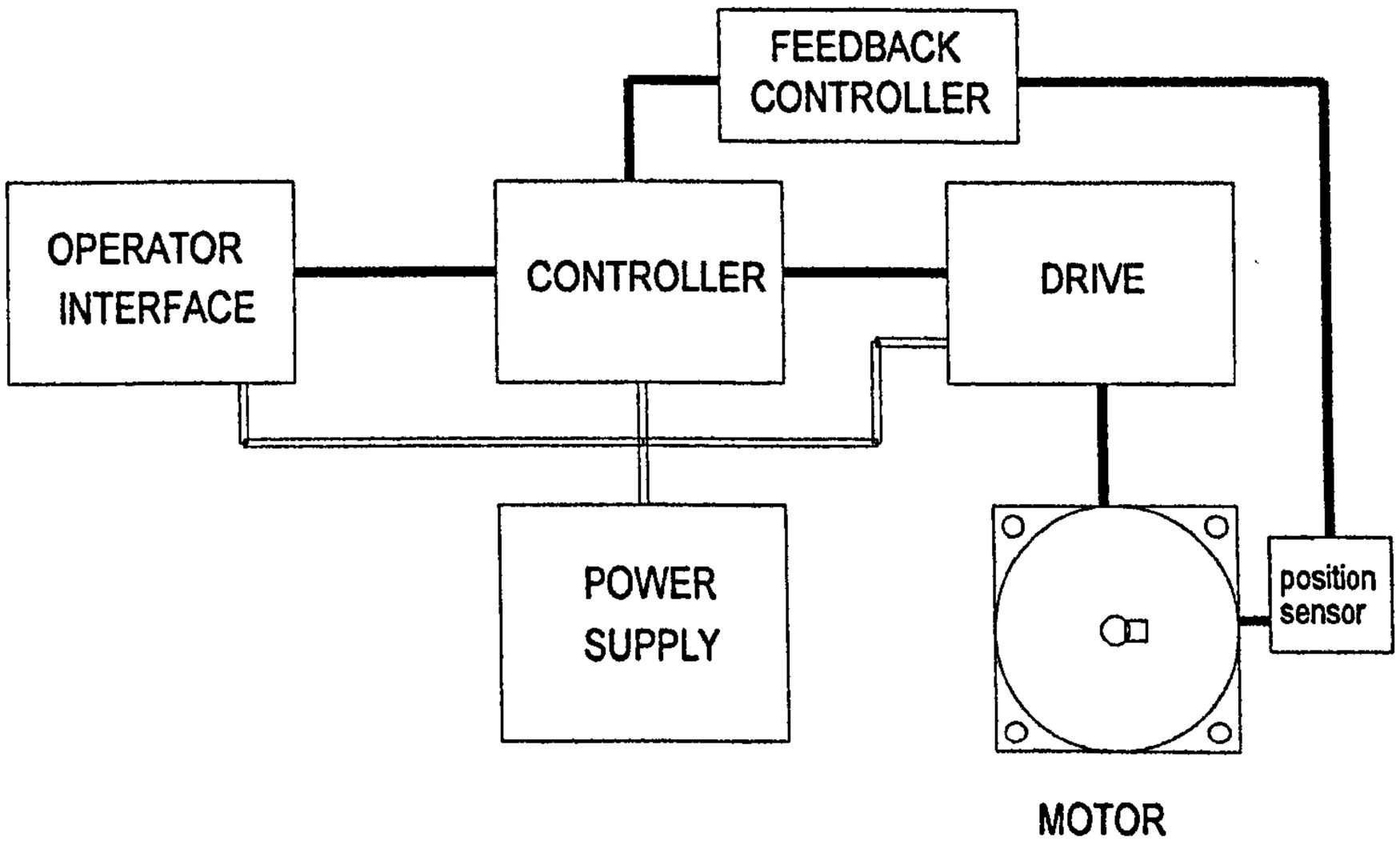


FIGURE 1.26 - Motor Drive System with Closed Loop Control

1.7 Why Investigate the Hybrid Stepping Motor.

Though the hybrid stepping motor has a long and proven history, in terms of toughness, accuracy of position and the ability to operate in open loop [5], motor performance improvements are still to be made in terms of power drives, controllers, and the physical motor construction. There is now increased competition from brushless DC servo systems, which have been aided by the continual price reduction of positional devices, especially encoders and resolvers.

Motor constructors and design engineers are increasingly interested in how tooth configuration, mechanical geometry, and magnetic material compositions affect motor performance [6,7]. These all jointly affect the generation and distribution of flux and therefore torque.

Some novel designs of the hybrid stepper motor have come on to the market in recent years, notably 3 and 5 phase excitation schemes [8,9,10,11] and also the use of so called

flux focusing stator magnets [12,13,14]. Three and 5-phase design claim to have a better settling time and winding copper utilisation due to the increase number of phases. Referring to equation 1.1 it can be seen that 3 and 5 phase hybrid stepper motors will have smaller natural step angles. Flux focusing stator magnet motors have a neodymium magnet placed in the stator teeth troughs. It is claimed that these force the leakage flux into the tooth steel, allowing more flux to generate torque. This method produces approximately 20% increase in holding torque.

The hybrid stepping motor has what is considered to be an unconventional magnetic construction. The magnetic fields involved produce a truly 3-dimensional electromagnetic problem, as significant radial, axial and tangential components of magnetic flux occur throughout the motor. The complex magnetic flux pattern makes performance predictions difficult. Stepping motors have been traditionally designed using empirical methods, relying on experimentation and experience [5]. It is now possible to consider studying these complex magnetic problems with new software packages, because of recent advances in computing power [15,16]. Using tools such as three dimensional finite element analysis software, a designer can produce an accurate model of a hybrid stepping motor. They are able to view the behaviour of flux paths, effects of material types, and generation of torque, without having to construct a prototype.

1.8 Format of this Thesis

A key component of this thesis describes the use of finite element analysis with the power to overcome some of the traditional problems encountered with such analysis. The use of

finite element analysis for the stepping motor provides the foundation of this doctoral project. This will validate assumptions commonly made about the hybrid stepping motor. This work is described in chapters 2, and 3.

Using data and knowledge gained from the three dimensional finite element analysis, improved analytical techniques for prediction of motor performance will be developed. This will involve simpler but accurate models that will eventually be incorporated into a computer aided design package. The simpler models are necessary so that they can be easily modified during the design phase without the requirements for complex 3-D modelling. This is presented in chapter 4 and 5.

In chapter 6 and 7, the modelling observations from the various tools are used to design two new hybrid stepping motors, which are constructed and tested.

Chapter 8 completes the thesis with some conclusions.

CHAPTER 2

THE HYBRID STEPPING MOTOR; ASPECTS OF CONSTRUCTION, TESTING, AND MODELLING

This chapter will briefly detail the manufacturing process of the hybrid stepping motor and will then describe some of the hardware apparatus and tests that are used in both manufacturing and experimentation to confirm that an existing motor meets set performance criteria, or that a prototype motor conforms with design or software modelling. An introduction is given to some of the computational methods used in motor design.

2.1 Manufacture of Modern Hybrid Stepping Motors

The hybrid stepping motor has few dedicated manufacturers in the UK. This project was sponsored by Stebon Ltd. and hence the following section is based on the experience gained on the shop floor and test area at Stebon and the work detailed by Stebon's technical manager, Mr D. Baker[17].

As described in chapter 1, the motor consists of a stator pack, stator winding, and a rotor assembly. The assembly of each part will be described before discussing the complete motor assembly and testing.

2.1.1 Stator Construction

The major component of the stator is the core pack. The stator laminations are brought in pre-punched and annealed. The laminations are typically 0.35 mm Transil. To construct a core pack, half of the required laminations are placed one on top of each other on a jig. The laminations are held in the same position by use of a datum point on the lamination, customarily positioned between the middle slot of the castellations. A layer of adhesive is placed between each lamination. The adhesive layer contains typically two types of glue, each with a different temperature stability. After the first half has been assembled on the jig the stack is turned over and the second half added to allow for lamination punch curvature. The complete stack is compressed to improve the packing factor. The packing factor is the ratio of steel material to total stack length. The stack length is checked and more laminations may be added to match the specified length.

The completed stator lamination core packs may be placed into a steel shell, depending on the design and protection rating of the motor. Each lamination in the stator core pack has an inner diameter that is undersized. This infers that if an un-machined rotor lamination was placed on to a stator lamination there would be no air-gap as the inner diameter of the stator lamination and the outer diameter of the rotor lamination would overlap. The completed stator core is then machined (honed) to produce a concentric inner diameter.

Copper windings are pre-wound on bobbins and taped to hold their shape. A pair of windings is selected for the requested performance, determined by the desired torque/ speed performance and thermal requirements. A pair of windings is placed on to two of the stator poles of one phase. A second pair of coils is placed on the remaining two poles

so that each phase is made up of four coils in two electrical circuits. Each phase is terminated with four wires that are appropriate for either terminal box or flying lead termination, providing the ability for the coils to be connected in series or parallel. The stator packs are then impregnated with varnish to provide mechanical rigidity. The varnish is cured by placing the stator pack in an oven.

2.1.2 Rotor Construction

Rotor construction starts by producing two rotor end-caps both made up of oversized outer diameter laminations. Like the stator, the rotor laminations are held together by a combination of adhesives whilst being built up on a jig. Each rotor requires a pair of end-caps for each stack. For example a 3-stack motor requires 6 end-caps. One of the rotor end-caps is placed on a jig, and an un-magnetised permanent magnet is placed on top. The permanent magnet material is generally alnico, samarium cobalt, or neodymium iron boron. The choice of magnet material is dependant on the application, or the motor's working environment. The second rotor end-cap is then placed on the jig, and is correctly positioned using a second datum. This ensures the top end-cap has its teeth lined up with the bottom end-cap's troughs.

The rotor assembly is now continued by placing the end-caps and magnet components on a non magnetic stainless steel shaft. To make a multi-stack rotor construction, a spacer is placed on the shaft between the stacks. This spacer is typically made of aluminium.

The outside rotor diameter and shaft configuration (keys etc.) are now machined along with the bearing seats to maintain concentricity. Bearings are then pressed onto the completed rotor assembly.

2.1.3 Rotor and Stator Joining

The rotor / stator air-gap will typically be 0.1 mm from stator to rotor tooth. The air-gap must be kept as small as possible to ensure maximum torque generation. The rotor is kept in the correct orientation and position relative to the stator and mated to the stator by the use of end brackets. Firstly, the rotor is pressed into the drive end bracket (front flange), where the shaft extends. The front flange is used to hold the motor in position on the motor's final application. The flange is typically square and is usually standardised to a NEMA (North-American Electrical Manufacturers Association) configuration that dictates the motor frame sizing, though other flanges may alternatively be specified. This front flange is joined to the stator assembly by lining up spigot holes and pressing the stator onto a lip, positioned by through bolts. Further spigots are used to join the stator and rotor assembly to a rear end bracket. The leads from the phase windings are pulled through this end bracket. They are either left as flying leads or are alternatively crimped with spade terminals for use with a terminal box. This rear end bracket has provision for the placement of a position verification device or an extended rear shaft.

The complete assembly is now placed in a high energised field that axially magnetises the rotor in situ. This magnetisation in situ offers a significant advantage over other permanent magnet motors as it allows the rotor to be assembled completely before it is magnetised.

2.2 Experimental Arrangements

This section describes the experimental arrangements that are commonly used to verify that a constructed motor meets performance criteria. Testing arrangements are also used to confirm modelling techniques and the accuracy of design predictions that are used later in this chapter and thesis. Testing covers both static and dynamic arrangements.

2.2.1 Open Circuit Back EMF Testing (Figure 2.1)

A simple test to calculate the rotational EMF produced from a motor, is achieved by connecting another precise speed motor (typically another hybrid stepping motor), to the shaft of the motor to be tested. The driving motor is run at a set speed which causes the test motor to generate an EMF across its phase windings due to the rotating magnetic field. The RMS value of the EMF is recorded and the near-sinusoidal wave form may also be captured. This test is often used in manufacturing to check that the permanent magnet has been fully magnetised, the winding pattern is correct, and the rotor has concentricity.

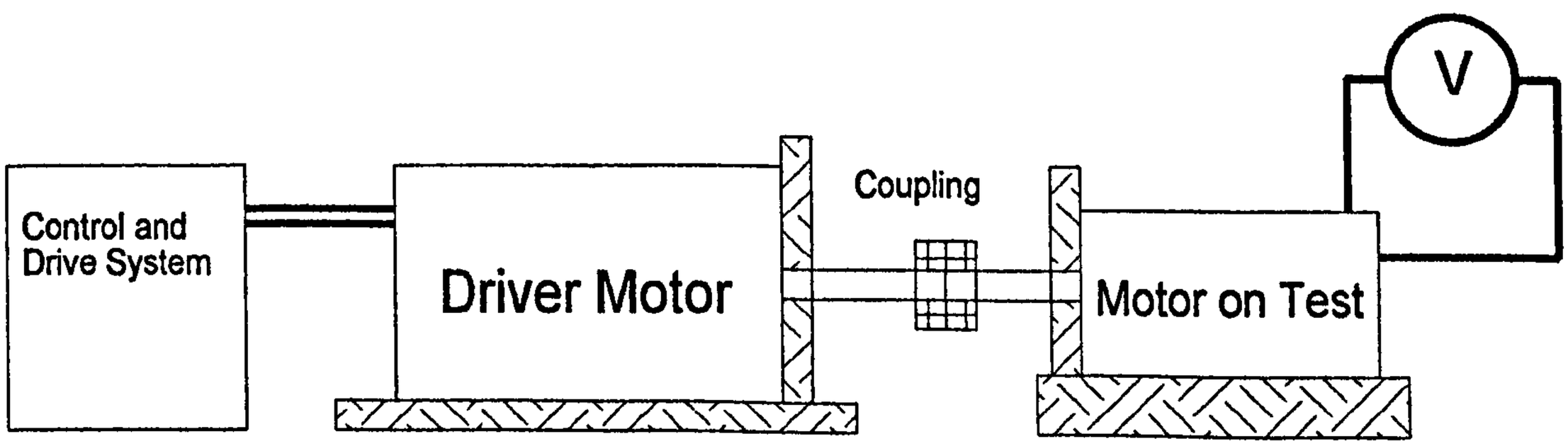


FIGURE 2.1 - Back EMF experimental Arrangement

2.2.2 Static Torque

Static torque may be measured in terms of single phase or two phase torque. Here the normal drive circuitry is replaced by a controlled DC current, supplied into the phases.

With no load the motor will sit in an equilibrium position. If load is applied, the rotor will rotate to a new rest position. If the measurement system is a simple string and pulley the torque produced will be equal to the force applied multiplied by the radius of the pulley. The force is recorded along with angular displacement to produce the approximate sinusoidal curve for torque/ angular displacement. The torque will reach a maximum, close to mid-way between the un-aligned and aligned equilibrium points (figure 2.2A). The point of maximum torque is found to be typically just after the mid point slightly closer to the un-aligned equilibrium position. As the angle between equilibrium points is extremely small, the string and pulley measurement method can be very inaccurate and measurement on a special test rig is preferred..

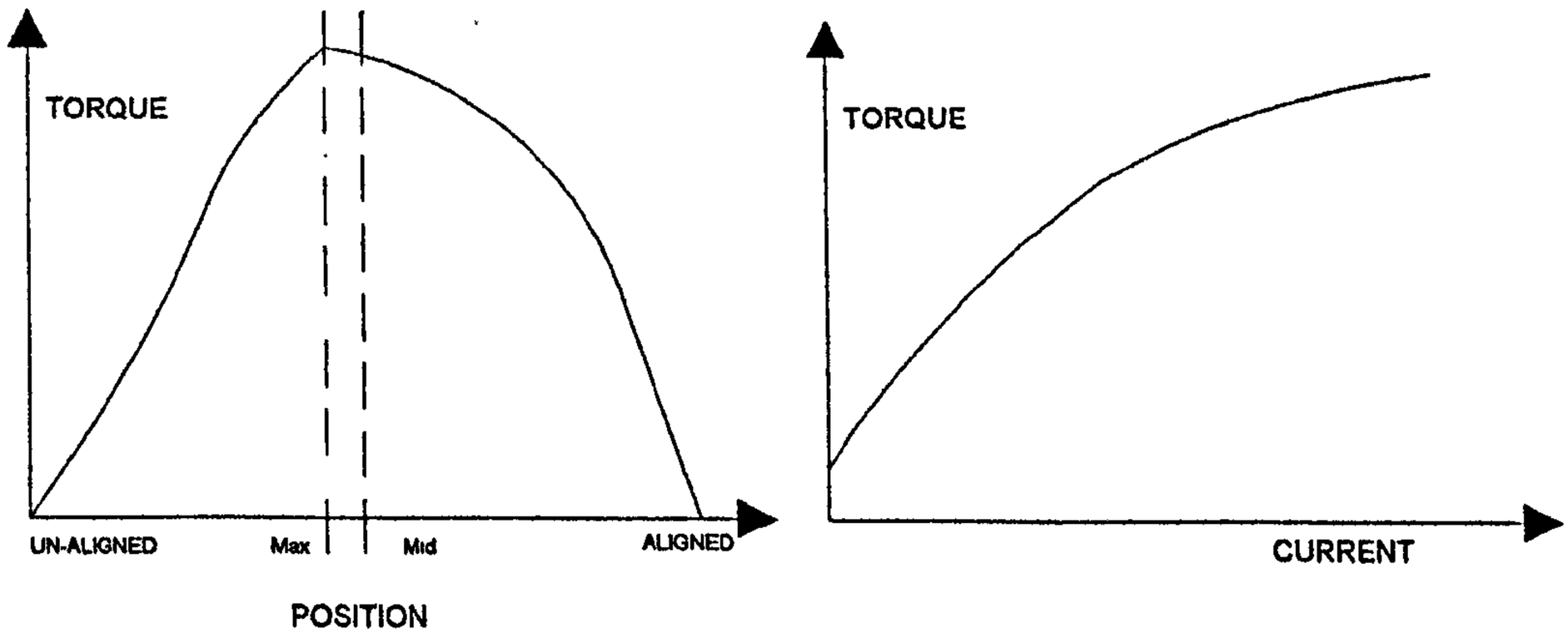


FIGURE 2.2 (A) - Static Torque against Position **(B)** - Static Torque against Current

The maximum static holding torque increases with current until the motor's steel saturates. A representative curve is shown in figure 2.2B. Here the motor is shown to start at a value greater than zero. This is due to the detent torque produced by the permanent magnet. The motor torque rises linearly at low current and 'flattens' off as the current increases and the motor steel saturates.

2.2.3 Dynamic Torque

In the industrial world of stepping motor applications the dynamic torque is often seen as the critical aspect of performance. Standardising dynamic performance of motors is difficult to achieve. For example there are dynamic differences between a test rig and a motor driving a conveyor belt system in a factory. The inertia differences of the systems affect starting (pull in) and stalling characteristics. Drive circuitry, and in particular differences in inductance, efficiencies, and pulse input circuitry, from manufacturer to manufacturer will also dictate the way a motor will perform. The choice of current step patterns, anti-resonance circuitry, and drive voltage further complicate the situation.

A set of illustrative performance curves are shown in figure 2.3. The curves shown represent both parallel and series motor connections. Since most drives have a pre-set current limit at low speeds the series connection can be seen to produce almost double the low speed torque as the active MMF is effectively doubled. However this torque drops off at a lower speed due to the rapid build up of back EMF. The effect of increasing voltage (dashed lines) can be seen to increase the speed range (figure 2.3). In figure 2.3, the effect of instability or resonance is shown This may be due to motor construction/ design, coupling and backlash effects of the measuring system, or motor/ drive inductance instability. This can be reduced with the effect of microstepping or use of an anti-resonance circuit. An anti-resonance circuit may be either passive or active. A passive circuit will cap the peak current at low speeds. An active circuit monitors the current in the phase windings, and will reduce the current when instability is present.

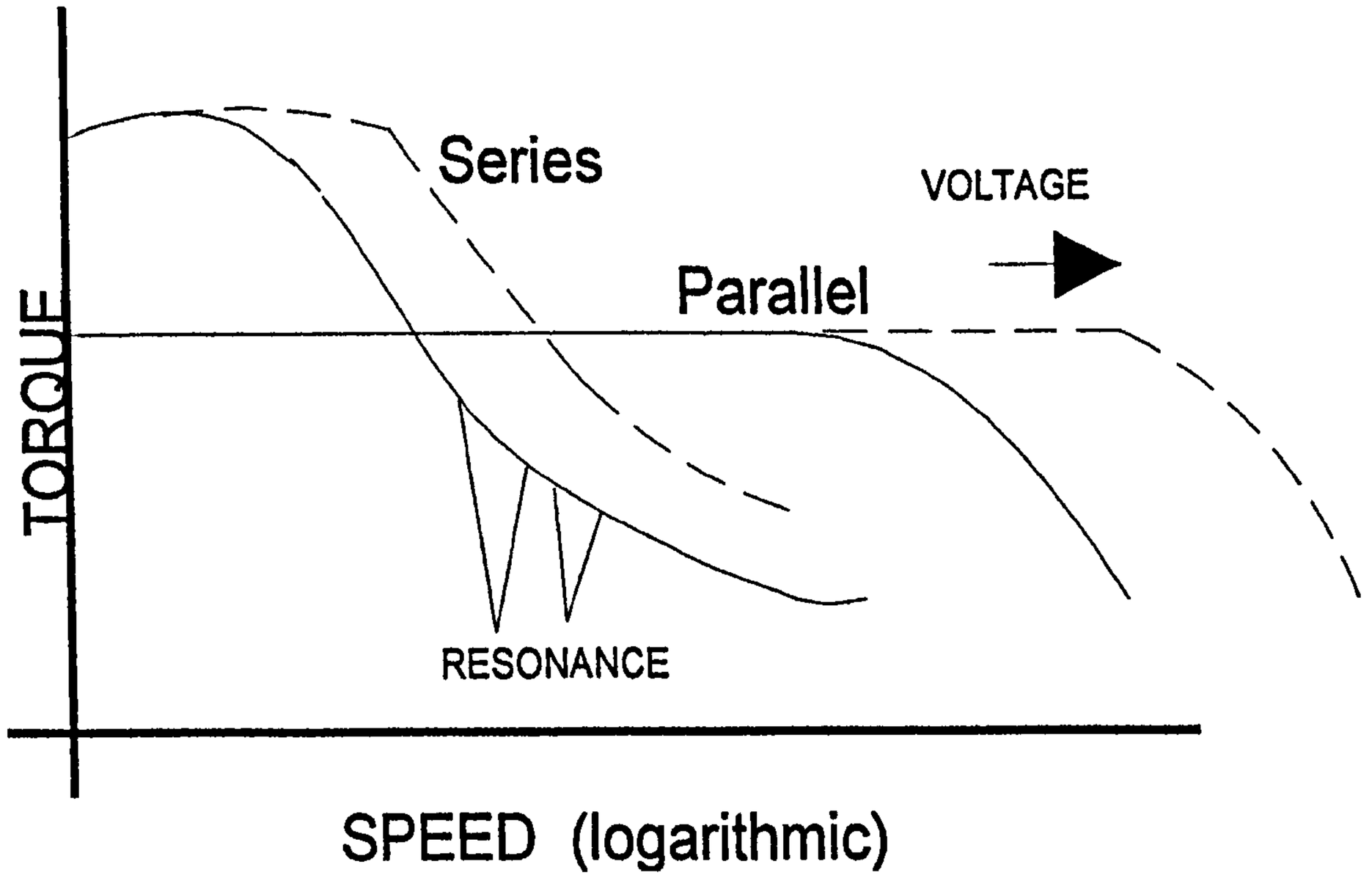


FIGURE 2.3 - Dynamic Torque/ Speed Characteristics

2.3 Test Rig Apparatus

In an attempt to standardise testing, a test rig was constructed. The test rig was designed so that EMF, static, and dynamic torque could be measured on one platform. The basic construction is shown in figure 2.4. A motor is positioned at one end which can be used to drive the shaft, which in turn is mechanically coupled to the motor on test. This can be used to drive the motor on test and hence obtain back EMF information.

Between the motors is a powder brake dynamometer which acts as a load and speed torque transducer. This transducer will also feed-back the torque and speed to a measurement display. On the end of the shaft is an encoder which allows position feedback for static torque measurement. The encoder also allows stall detection for automated dynamic testing.

A dual channel current probe is also fitted which allows the current waveform from the phase windings of the motor under test to be monitored and fed back to the control /display system. Whilst much of the hardware was constructed by Stebon Ltd. the author was responsible for all the test rig control and measurement software.

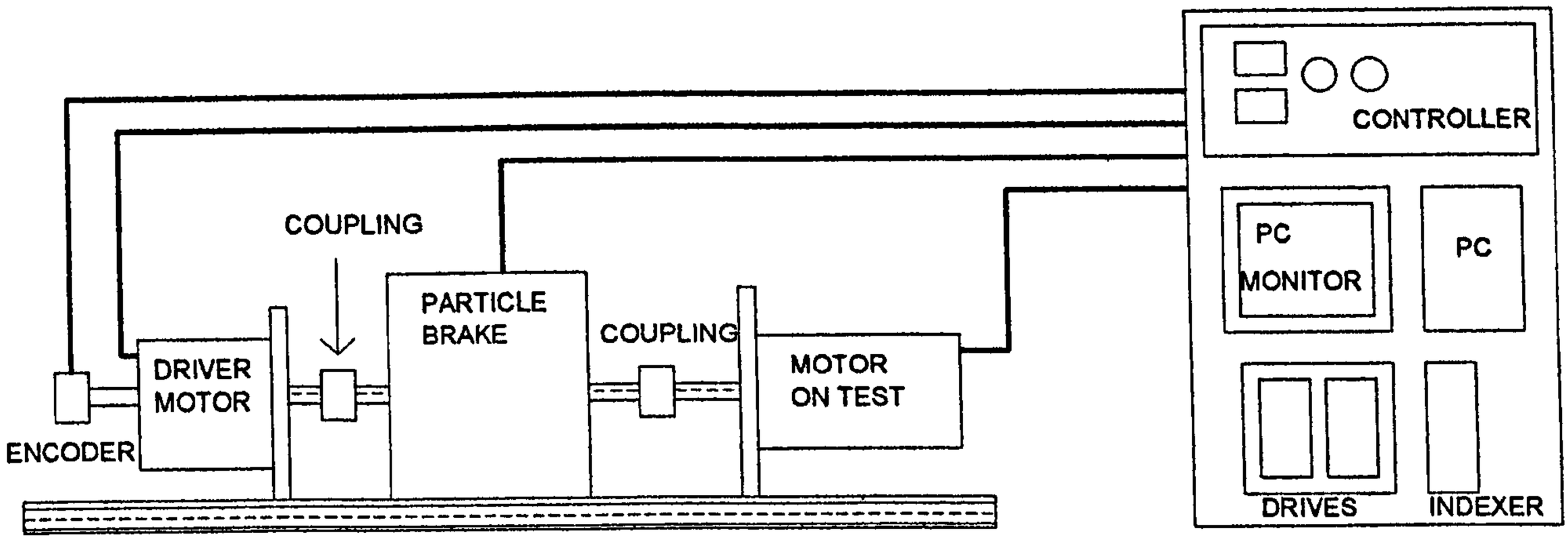


FIGURE 2.4 - Test Rig Layout

2.3.1 Control and Display

The control and display system are made up of a direct feedback display and control panel for the speed/ torque transducer. This is also routed through a personal computer that has suitable input, output, and data acquisition PC cards. This enables the speed, torque, motor controllers, current probes, and encoders to be read or fed by software running on the PC. The software that operated on this test rig was written in the Visual programming language Labview™.

2.4 Specialised Tests

2.4.1 Current Waveforms and Inductance Calculations [18]

The current in the phase winding can be captured by the PC on a Labview™ virtual instrument that acts very much like a digital oscilloscope. This allows the two phase currents to be monitored, captured and then examined. The current waveforms are caught using a Hall effect current transducer. This tool is also useful for capturing the rise of the current following the input of a voltage step. This allows the inductance of a winding at a fixed rotor position to be found. Using the circuit of figure 2.5 and capturing the current (and voltage) waveform the inductance at a fixed rotor position can be calculated by the relationship between instantaneous current and inductance which is;

$$v = iR + L\frac{\mathrm{d}i}{\mathrm{d}t}, \qquad (2.1)$$

where v is the instantaneous voltage,

i is the instantaneous current,

L is the inductance of the circuit,

and R is the resistance of the coil.

The inductance can also be represented by equations 2.2 and 2.3.

$$L = N\frac{\phi}{i}, \qquad (2.2)$$

$$L = \mu_0 \mu_r N^2 \frac{A}{\ell}, \qquad (2.3)$$

where L is the inductance,

N is the number of turns linked by the magnetic field,

ϕ is the magnetic flux,

i is the current in coil,

μ_0 is the permeability of free space,

μ_r is the relative permeability of the magnetic circuit,

A is the Cross-sectional area,

and ℓ is the mean flux path length.

It can be shown that as the rotor turns the reluctance of the magnetic flux path and hence the winding inductance will change. Whilst the inductance calculation is an important reading in the theory of motor testing and performance prediction it is non-linearly related to the flux linkage of the motor, a term commonly referred to in the design of all types of stepping motors [19,20]. It is therefore preferable to use the test rig to measure flux linkage rather than inductance.

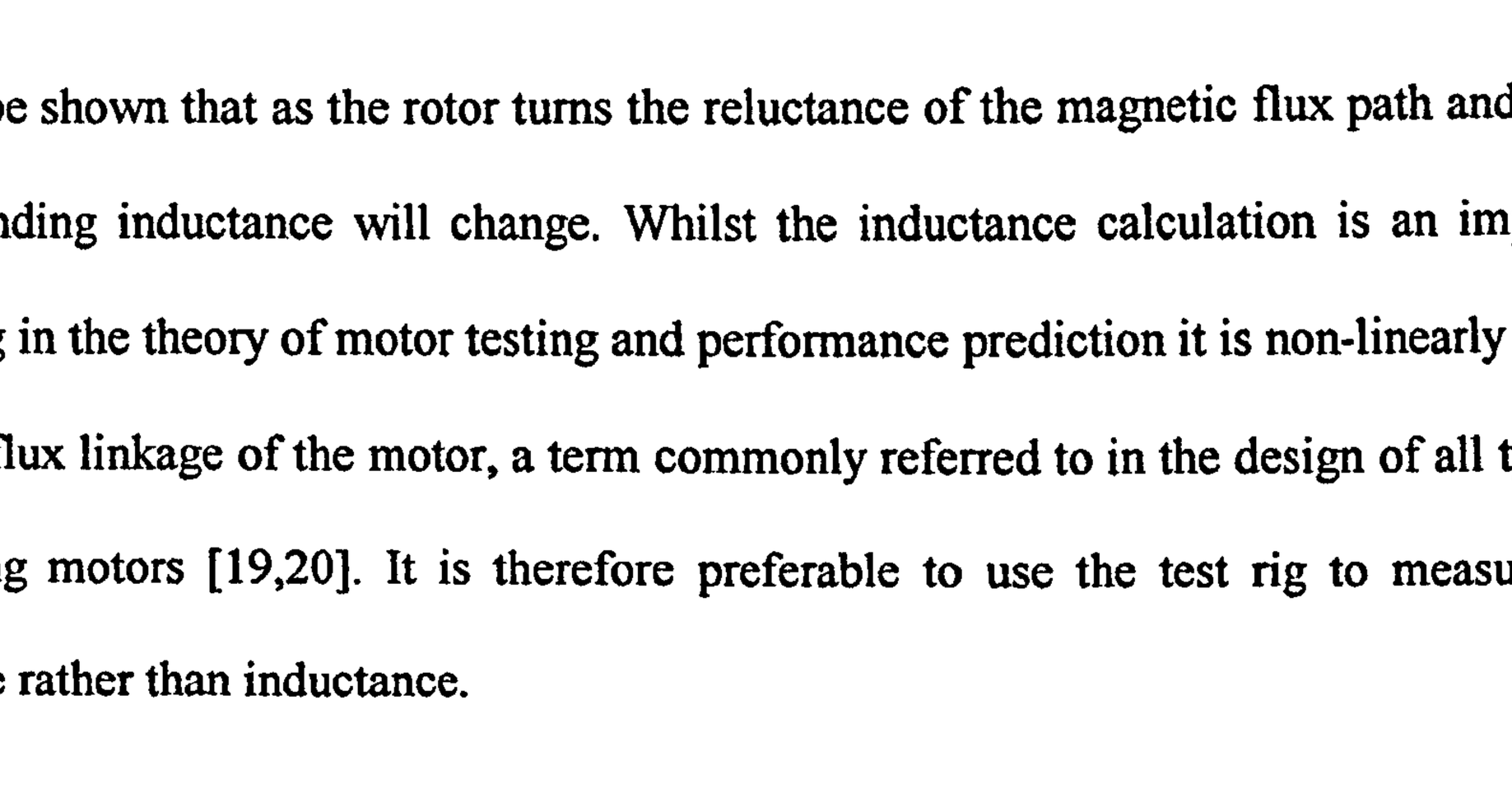


FIGURE 2.5 - Inductance Calculation Circuit

2.4.2 Flux Linkage Measurement [21[1]]

In its most common form, an electrical machine converts energy from electrical to mechanical or vice versa via a coupling magnetic field. Flux linkage is a key to describing this coupling characteristic. Measurement of the flux linkage in an electrical machine is useful for validation of design models. The knowledge of the magnetic energy stored in the coupling field is useful for optimising machine design. The relationship between phase current, flux linkage and mechanical output can be seen as representative of the whole energy process in the motor.

The modelling of these relationships can be achieved by modern computing tools such as finite element analysis methods described later in this thesis. However accurate measurement of flux linkage is still essential to validate these models.

In a machine where the excitation system is only by current in the windings, such as for a variable reluctance motor, the measurement of flux linkage can be obtained by integrating a measurement of the phase winding voltage. In the hybrid stepping motor the measurement is complicated by the presence of another MMF source, the permanent magnet. The measurement of the magnetic field inside the motor is not easily obtained. Sensing coils or magnetic sensing probes could be added to the motor but these complicate the motor design and cannot be added to existing built motors. In any case data collection would be extremely troublesome due to the uneven distribution of flux in the air-gap and steel, making it difficult to obtain accurate flux linkages. One method to acquire the magnetic characteristics of a physical motor is to calculate the stator flux-

---

1.This section is based on a paper presented at 1994 IEE Power Electronics and Variable Speed Drives conference

linkages indirectly by integrating measurements obtained from the phase winding circuit (figure 2.5) [22].

2.4.3 Theory of Flux-Linkage Measurement

One of the most common methods for measurement of flux-linkage is to apply a step voltage to a phase winding and record the resulting current. The measurement is based on the circuit equation described in equation 2.1. This may be expressed so to include flux-linkage $\lambda(\theta,i)$, (flux-linkage in terms of position and current).

$$v(t) = i(t)R + \frac{d\lambda(\theta, i)}{dt} , \qquad (2.4)$$

and integrating and rearranging equation 2.4,

$$\lambda(\theta, i) = \int_0^t [v(t) - i(t)R]dt + \lambda(0) , \qquad (2.5)$$

where $\lambda(0)$ is the flux-linkage at time t=0.

The change in flux linkage over a period of can be obtained from the measurements of the winding voltage and current, if the phase winding resistance is known. Initial conditions have to be chosen carefully with a known initial flux linkage $\lambda(0)$ at the time the integration process starts. In a system in which the current in the phase winding is the only source of flux, and providing the remnant flux can be neglected, the initial condition $\lambda(0)$

(integration constant) is conveniently chosen to be zero when the current in the phase winding is zero, regardless of rotor position θ, i.e.

$$\lambda(0)=\lambda(\theta, i=0)=0. \qquad (2.6)$$

In the hybrid stepping motor system which has two independent current sources, and a permanent magnet, the flux linkage associated with a particular phase is now a function of the two current sources and position. The circuit equation of one phase can be written as:

$$v_1(t)=i_1(t)R_1+\frac{\mathrm{d}\lambda_1(\theta, i_1, i_2)}{\mathrm{d}t}, \qquad (2.7)$$

where i_j is the current in phase j, (j=1,2).

If it can be assumed that i_2 and θ are constant, integrating equation (2.7), provides

$$\lambda_1(\theta, i_1, i_2)=\int_0^t [v_1(t)-i_1(t)r_1]\mathrm{d}t+\lambda_1(0) \quad , \qquad (2.8)$$

where

$\lambda_1(0) = \lambda_1(\theta=\theta_0, i_1=i_{10}, i_2=i_{20})$,

i_{10} is the initial current in phase 1,

i_{20} is the initial current in phase 2.

$$\lambda_m(\theta, i_1 = 0) = \int_0^t emf_1(t)\mathrm{d}t + \lambda_m(\theta = \theta_0,\ i_1 = 0, i_2 = 0), \qquad (2.9)$$

where

$\lambda_m(\theta=\theta_0,\ i_1=0,\ i_2=0)$ is the initial condition at which the integration starts (arbitrary initial condition),

$\lambda_m(\theta,\ i_1=0,\ i_2=0)$ is the flux linkage associated with the stator phase due to the permanent magnet at any position θ,

$emf_1\ (t)$ is the instantaneous motional EMF induced in phase 1 by the permanent magnet rotor.

The arbitrary initial condition $\lambda_m(\theta=\theta_0,\ i_1=0,\ i_2=0)$, was chosen to be zero when $\theta=\theta_0$. It represented the position of minimum flux-linkage in stator phase 1. All the integration results were therefore relative to this initial value.

2.4.4 Experimental Measurement Method

The experimental method includes two separate tests.

1) Permanent magnet EMF.

2) Locked rotor current waveform test.

1) This test is described in section 2.2.1. The EMF waveform is then integrated using equation 2.9 to obtain the flux linkage due to the permanent magnet at each angle θ.

1) This test is described in section 2.2.1. The EMF waveform is then integrated using equation 2.9 to obtain the flux linkage due to the permanent magnet at each angle θ.

2) This method follows the theory of section 2.4.3. The rotor is locked at known rotor positions. A short voltage pulse is then applied to the phase winding and the current and voltage waveforms are captured. The current waveform is then used in conjunction with the terminal voltage and winding resistance to calculate the change in flux linkage arising from the applied voltage using the initial condition for that position obtained from the magnet EMF test. This test was repeated for all the desired flux-linkage rotor positions with positive and negative phase current. The results obtained will be shown in chapter 5.

2.5 Prediction of Stepping Motor Characteristics

Stepping motors have been designed in the past by working on knowledge gained from experience, testing and experimentation. Engineers designing motors (and other electrical devices) wish to improve mathematical analysis without the use of time consuming experimentation. Traditional analysis involved simple geometry, assumptions and imagination. This approach has been outmoded although not replaced, as demands for faster and more accurate solutions for the complex geometry found in such motors as the hybrid stepping motor are sought.

The prediction of the performance of the hybrid stepping motor has been widely published by some key authors and researchers. Chai is a well-respected author for work concerning

modelling, especially on permanence based models [6]. Many papers have been published on attempts to provide equivalent circuits for the motor [23-34]. These equivalent circuits rely on tooth and air-gap permanence models that link the mechanical geometry of the motor to the magnetic circuitry, by typically breaking the motor into key parts. Once established, the equivalent circuits attempt to predict the inductance, flux-linkages, and torque, for differing phase currents, drive voltages and motor speeds.

2.5.1 Use of Electromagnetic Analysis Software

The many factors that affect the performance of the motor have been rarely modelled together, due to the diversity of the effects which occur throughout the motor. A total solution lends itself to use of computerised calculations for electromagnetic fields. Methods which fall into these categories are Finite Element Analysis (FEA), Boundary Element Methods (BEM), Boundary Integral Methods (BIM), Finite Difference Methods (FDM), and time-domain methods [35,36,37].

2.5.2 Work carried out on Finite Element Modelling of Hybrid Stepping Motors

The number of publications specifically about the use of FEM in the analysis of the hybrid stepping motor is very small. Two notable works have been by B. Forghani of Montreal [38], and E.C.T. So and S.J. Yang of Heriot Watt, Edinburgh [39]. B. Forghani's paper describes a 3-D solution of the hybrid stepping motor. A brief discussion of the ability to use a 2-D method coupled to an equivalent circuit is given. However the three dimensional modelling is one of the only few attempts at an overall solution. The hybrid stepping motor's geometrical shape is simplified to reduce the computational effort

required. The laminated nature of the motor is ignored and the motor is modelled as a non-laminated component. Because of simplifications made the detail of analysis is concentrated on the main flux path's in the motor. A method for calculating the torque-rotor position using the virtual work method is also discussed. The paper by E.C.T. So and S.J. Yang [39], though primarily concerned with stator vibration calculations, introduces methods for modelling a stator tooth and corresponding rotor teeth of a hybrid stepping motor. It includes points on simplifications made, boundary conditions applied and limitations of their meshed model. It gives a useful insight into the modelling of a stepping motor. However it does not obtain a complete solution for the motor.

2.5.3 Simpler Computational Methods in The Design of Stepping Motors.

Finite element software and other electromagnetic field software are still relatively expensive. Generally companies contract out design work when the modelling involves extensive electromagnetic field analysis, or because of the initial software/ hardware and training costs. However from the knowledge gained by using FEA and by improving equivalent circuits, more complete and accurate models may be created. These can form the basis of a computer aided design environment that uses these improved mathematical algorithms to predict the behaviour of the motor. Similar attempts have been made by Wang *et al* [24], who have published a paper on the study of CAD for stepping motors. It addresses the establishment of what the authors call an improved CAD model. Optimal design problems associated with the motor are also discussed.

2.6 Discussion

The physical construction and the methods employed to manufacture the hybrid stepping motor must be understood before design methods can be usefully employed. The start of this chapter reviewed the typical processes of manufacturing the motor and described the methods used to maintain concentricity of the motor.

Experimental arrangements were discussed for validating models and measuring a motor's static and dynamic performance. All experimental results that are presented in subsequent chapters have been obtained by adopting the procedures described.

The chapter concluded with a brief introduction to the use of mathematical modelling for the prediction of the performance of the hybrid stepping motor. These include equivalent motor circuit models and commercially available computational products such as FEM software. Both are described more fully in the following chapters.

Through the use of three dimensional analysis, a full three dimensional representation of the hybrid stepper motor will be developed. This thesis will describe a simpler algebraic model that has been developed and verified with the use of 3-D FEA and experimental testing. This algebraic model aims to produce accurate solutions with the advantages of faster computation times and simpler data input.

CHAPTER 3

ELECTROMAGNETIC MODELLING OF THE HYBRID STEPPING MOTOR[1]

3.1 Introduction to Advanced Numerical Techniques

Modern electrical machines must be designed to meet specified operating conditions that are set by the load demand, while achieving good efficiency and reliability. The power and torque output of the motor must be balanced by input power, motor volume and cost. In order to meet such requirements, accurate machine performance at the design stage must be established. It is difficult to build a complete solution of the hybrid stepping motor using simple analytical functions or equivalent circuit representations. This is due to the difficulties introduced by the motor's highly non-linear three dimensional magnetic structure, for which the doubly salient tooth structure, axial magnet, and back iron all complicate the situation.

In recent years numerical techniques have been developed that can overcome the limitations of linear and steady state problems and have provided accurate solutions to a wide range of electromagnetic problems. For example the forces exerted on the stator poles of the hybrid stepping motor have been calculated, by taking into account the three

[1] This Chapter is constructed from several papers published by the author;
Presented at IEEE conference on Magnetics, Okayama, Japan, March 1997.[40]
Proceedings of Vector Field User Meeting, September 1994, Blenhiem Palace, Oxford [41]

dimensional geometry and saturation effects of the materials using the finite element method [39].

Finite element analysis is the most widely used numerical method for transient and steady state solutions to two and three dimensional electromagnetic problems. The enormous capabilities of this technique are largely due to considerable advances in desktop computers. These advances allow the use of three dimensional techniques to be considered for the modelling and analysis of the hybrid stepping motor.

The computer based numerical technique for solving partial differential equations is implemented by representing the area of the problem under consideration by a collection of finite elements. The nodes associated with each element are the points in space where the field values are calculated. Governing equations for each element are set up and subsequently combined in order to describe a global property.

The governing laws of electromagnetism can be concisely expressed by Maxwell's equations [43]. Throughout the project the software used for the electromagnetic modelling of the hybrid stepping motor was supplied by Vector Fields Ltd [15,16,44]. The basic electromagnetic field theory on which the packages are based follows from the laws of electromagnetism and may be found in the Appendices.

3.2 Finite Element analysis of the Hybrid Stepping Motor

As reviewed in chapter 2 the amount of publications specifically on the use of three dimensional finite element methods (FEM) for the analysis of the hybrid stepping motor is small. A simplified geometry solution has been produced by *B.* Forghani where the fundamental flux paths in the motor have been shown [38]. E.C.T. So and S.J. Yang introduced methods for modelling a stator tooth and pole, and corresponding rotor teeth for vibration analysis [39]. The paper describes the simplifications made, boundary conditions applied and limitations of their meshed model. It gives a useful insight into the modelling of a hybrid stepping motor. Other papers, which discuss finite element analyses of stepping motors do so briefly or investigate the problem with 2-dimensional analysis. An excellent study of the hybrid stepping motor using 2-dimensional analysis is carried out in [45,46].

3.2.1 The Need for Three Dimensional Modelling of the Hybrid Stepping Motor

Finite element analysis (FEA) is considered to be highly suited to handle the modelling complexities introduced by the extremely saturated, doubly salient iron structure of the hybrid stepping motor. The question arises as to whether the hybrid stepping motor can be modelled, with sufficient accuracy, using a two dimensional code and an equivalent lumped circuit. As the amount of articles solely on three dimensional FEA on the hybrid stepping motor is limited, the variable reluctance stepping motor will be used to highlight common fundamental problems in two dimensional finite element modelling of doubly salient motors. Simkin and Trowbridge [47] reported that inductance calculations using a

two-dimensional computer program on a stepping motor had shown good agreement with measurements when the rotor teeth were aligned with the excited stator teeth. When the rotor was in the un-aligned position, values for computed and measured inductances disagreed. In the unaligned position the air-gap facing the excited stator poles is relatively large causing a significant axial component of flux to arise, due to end winding effects. As the axial direction cannot be modelled by two dimensional code the discrepancies were thought to be caused by this.

The hybrid stepping motor is a truly 3-dimensional electromagnetic problem with significant radial, axial and tangential components of magnetic flux occurring throughout the motor. Therefore it is impossible to obtain a complete solution by solely using a two dimensional package. Like most electrical machines, the hybrid stepping motor is usually laminated in order to minimise eddy current losses. Two dimensional analysis cannot model inter-lamination effects, which change the torque output of the motor. In a paper by Huard [46] it was discussed how the laminations made it extremely difficult to model a hybrid stepping motor with 2-D FEA and a comprehensive lumped model. Three dimensional FEA allows the user to input a packing factor for the steel laminations. This allows the program to calculate the effect of the laminations using anisotopic material properties, that is by assigning a relatively high permeability in the direction parallel to the laminations compared to a considerably lower permeability in the direction perpendicular to them.

The magnet characteristics can not be included in two dimensional FEM analysis because of its usual orientation, perpendicular to the laminations. Such effects must be modelled by lumped circuit models, in combination with two dimensional models.

Despite these observations, finite element analysis packages that employ two dimensional formulations are established as the primary tool in the design of the hybrid stepping motors. This is notably due to the complexity of the hybrid stepping motor as a complete model, but also due to computing power and man hours required for a three dimensional model. Users of finite element analysis software packages have found that two-dimensional models require far less CPU time to solve and occupy little disk space in contrast to three dimensional models. At one position at a particular excitation, three dimensional FEA would require 8 hours on an IBM RS6000 workstation and require around 45 Mb of disk space for the solution. For a lumped element network solver the 2-D data required to solve for three positions and ten varying excitations at each would be around 25 Mb of disk space and 2 hours total CPU time. In addition most 2-dimensional codes are available on PC platforms and are relatively inexpensive compared to their more complicated and expensive workstation cousins.

3.3 Typical Finite Element Software Package

A typical finite element software package contains *a Pre-processor, a Solver,* and *a Post-processor* [15,16]. The Pre-processor is where the user defines the model. The Solver solves the system equations, and the post-processor allows the user to view and analyses the results.

3.3.1 Pre-Processor

The first step in modelling, is to draw what is similar to a 2-D orthographic front view of the model. Usually this consists of the motor lamination with a surrounding air boundary. This view is drawn and defined initially by use of points and construction lines. Once the points are created, they are used as corners to create areas (facets), which are the different sections of the model. Once all of the 2-D area has been sectioned into facets, further sub-division can be achieved by splitting the facets into elements. To create a 3-D model the 2-D face is extruded to develop volumes in the Z-direction. Again, these extrusion layers can be subdivided. Now instead of areas the model consists of volumes. Each volume can be defined as a different material, for example, air or steel.

3.3.2 Solver

Here the Algebraic model is created. The user selects certain conditions to define the type of solution. Definitions for the units and materials are also required, i.e. non-linearity, anisotropy, periodicity and iterations. The solver then computes the numerical solution, which can then be post- processed. Once the solver is running there is no user interface.

3.3.3 Post-Processor

A solution file is read into the post-processor. Here the model can be viewed with the field potentials drawn or coloured onto its surface. Line graphs and histograms can also be created to show field values across parts of the model. Torque can be computed by Maxwell stress analysis, or through measurement of co-energy at different positions.

3.4 Modelling of a Simplified Hybrid Stepping Motor

To ascertain the functionality and abilities of the finite element software this exercise was used to get familiar with the finite element software. The primary aim was to develop a model that would correctly solve the basic fundamentals of 3-D flux distribution in the hybrid stepping motor.

The motor to be modelled was to be based on the laminations for a 4 phase switched reluctance motor with eight teeth on the stator and six on the rotor (8:6). Therefore the motor lamination could be simply adapted to create a basic hybrid stepping motor. In the simple model there are no teeth castellations on any of the poles. The lamination is shown in figure 3.1.

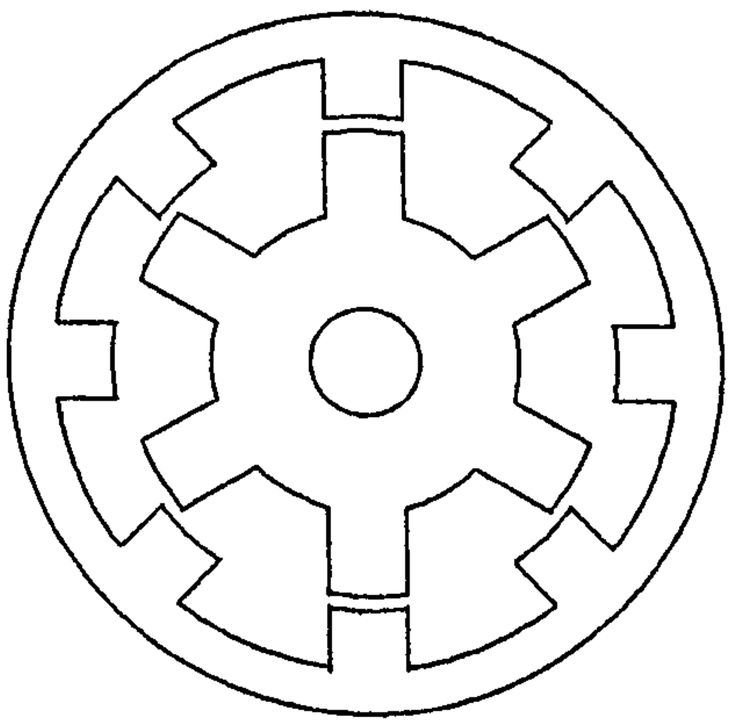


FIGURE 3.1 - 8:6 Switched Reluctance Lamination used for the Simple Stepping Motor

3.4.1 Tooth Construction

A key issue of modelling the hybrid stepping motor is the ability to efficiently replicate the teeth. Significant issues for modelling are that the teeth on each end-cap are out of phase with the other, plus the number of teeth being numerous on each rotor end-cap.

Even though the latter does not directly apply with this particular exercise, it was a notable consideration. Because of these key issues an investigation was made where a pair of rotor teeth (front and back end-cap positions) were modelled separately. This introduces the problem of creating the front and back teeth on the rotor of a hybrid stepping motor which are displaced relative to each other by half a tooth pitch.

Two methods in which the teeth can be practically created in a three dimensional model were investigated, these were respectively;

(i) Hidden Detail Method,

(ii) Twisted Extrusion Method.

3.4.2 Hidden Detail Method

Figures 3.2 and 3.3, show the first method . In this method all detail is drawn on one layer, the base plane, figure 3.2. This layer includes all hidden detail. The base plane would practically be the same as the face view on a draughtsman's orthographic drawing. The next stage is to subdivide the plane into 2-D elements. Once completed the plane is then extruded in layers in the Z-direction, to produce a model in three dimensional elements. To obtain a tooth structure as in figure 3.3, the elements in figure 3.2, are correctly associated with the material which is present in each layer. For example the elements which are solid on the top layer are defined as steel, the other elements are defined as air. This process is continued through all the layers and builds up the three dimensional structure of figure 3.3.

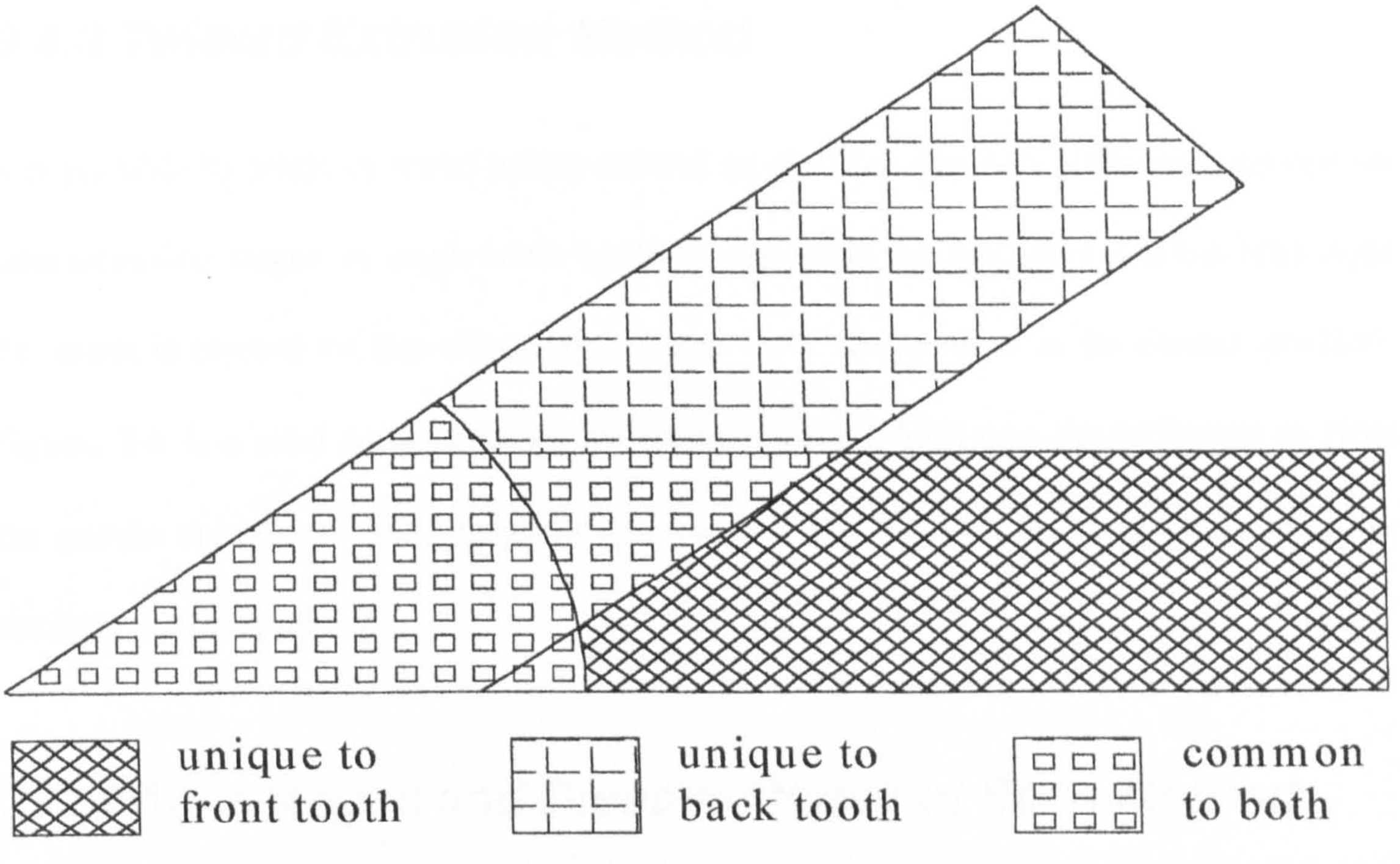


FIGURE 3.2 - Hidden Detail Method for Creating Teeth

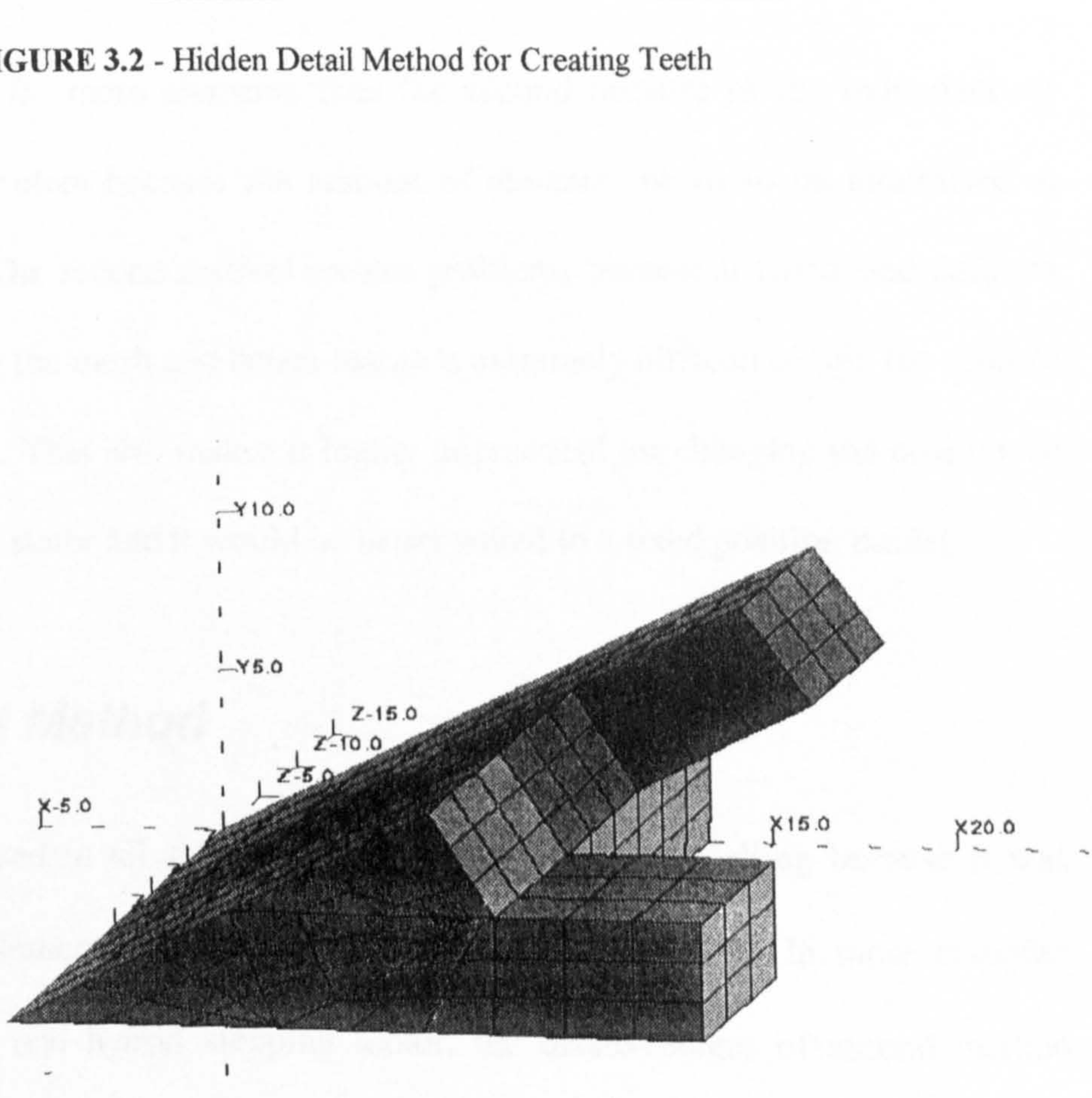


FIGURE 3.3 - Teeth Created by Hidden Detail Method

3.4.3 Twisted Extrusion Method

It is possible to rotate or move points created on the base plane to different positions on later extrusion stages. A single tooth could be drawn on the first layer. On the next layer the mesh is twisted so that on the final tooth layer the tooth is in its correct position. Figure. 3.4, is a rotor constructed by the second method. This uses the technique to twist the middle layer (permanent magnet) thus locating the teeth on the front and back rotor stacks in correct positions.

3.4.4 Advantages and Disadvantages of Each Method

The first method uses far more elements than the second because of the redundant air elements. This is a problem because the amount of elements needs to be minimised in modelling the motor. The second method creates problems because it twists and deforms the air gap surrounding the mesh and hence makes it extremely difficult to join the rotor to the stator in modelling. This also makes it highly impractical for changing the position of the rotor relative to the stator and it would be better suited to a fixed position model.

3.4.5 Preferred Method

The first method is used in all subsequent three dimensional modelling because it was much simpler to implement and it avoided corruption of the mesh. In more complex geometries such as a real hybrid stepping motor, the disadvantages of second method make it too complicated to reap the benefits of the reduced elements. The second method would be nearly impossible to use as there would difficulty matching element nodes on the rotor to those on the stator. The second method is best matched to modelling complete

models at specific positions. For different rotor positions, an individual model would need to be created.

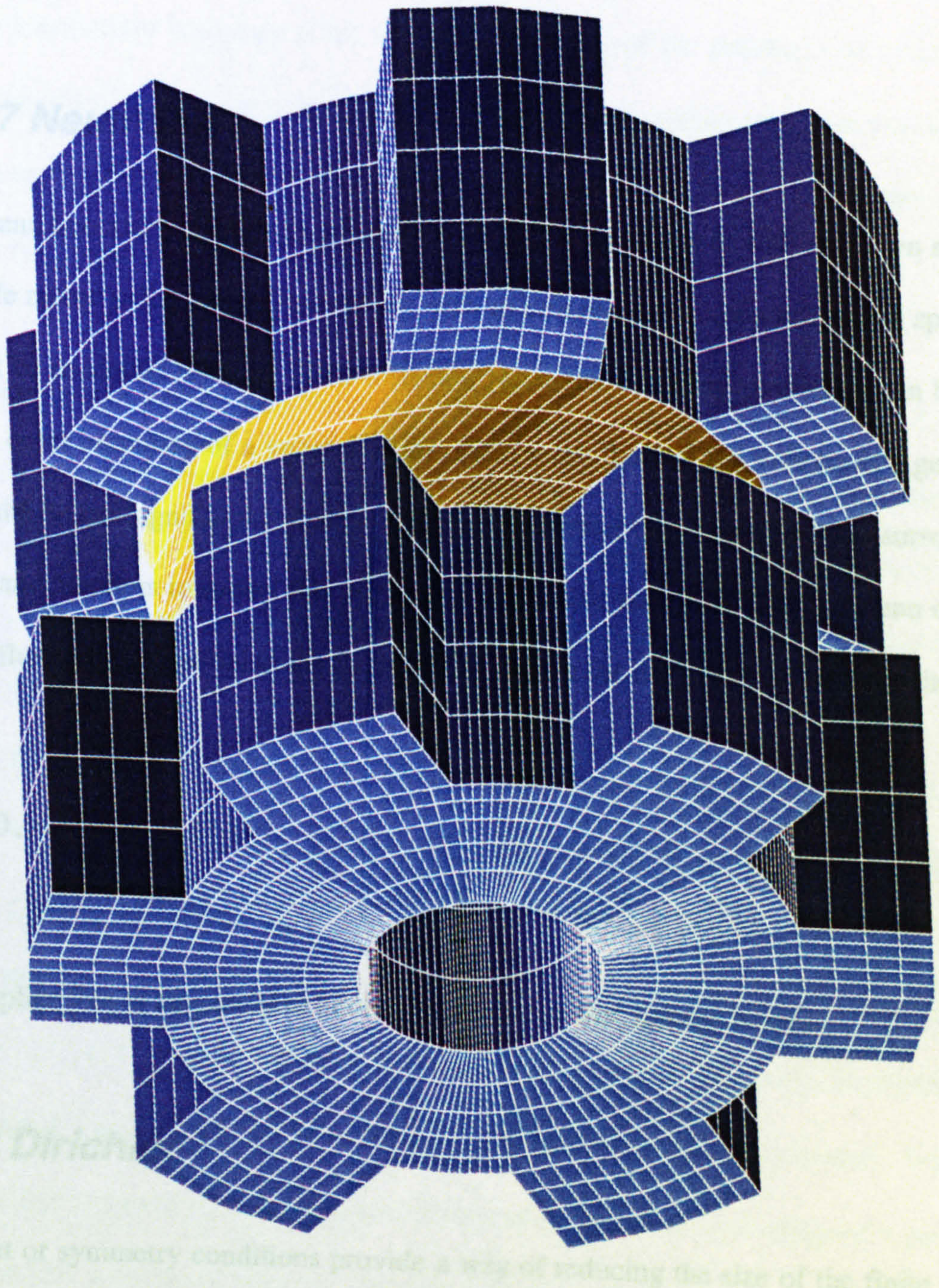


FIGURE 3.4 - Twisted Rotor Method

3.4.6 Boundary Conditions

There are two main types of boundary conditions available in the Finite element software package. These are known as the Neumann and Dirichlet conditions and are used to constrain and define the model boundaries.

3.4.7 Neumann

Neumann boundary conditions are used to constrain the motor to within its own system. A real life motor's flux interacts in theory with an infinite amount of surrounding space. This cannot be modelled with the FEM software. Neumann boundary conditions can be placed on the outer boundary of a volume of air (three times the motor's volume is a good value as results do not significantly change with results greater than this[48]). This surrounds the motor and attempts to model this effect to a reasonable accuracy. The Neumann condition forces flux to be tangential to the boundary by setting the derivative of the potential to 0;

$$\frac{\partial \phi}{\partial n} = 0, \qquad (3.1)$$

this implies flux is tangential to boundary and hence flux linkage is zero.

3.4.8 Dirichlet

Dirichlet or symmetry conditions provide a way of reducing the size of the finite element representation of symmetrical problems by setting the potential on the boundary to zero;

$$\phi = 0, \qquad (3.2)$$

which implies that flux is normal to the boundary.

When a symmetry boundary is set up, before analysis of the problem can be undertaken (field equation solver), the periodicity of the symmetry boundary is required to be defined. Symmetry boundaries may be either positive or negative. A negative boundary condition means the potential of one node is minus the potential of its pair.

An example of a simple stator/ rotor structure is given to illustrate periodicity. Figure 3.5. Concentrating primarily on the stator, the stator has been subdivided by lines labelled A to E. The letter in boxes indicate whether a particular pole is excited as a north or south field. A pole pitch of the stator is defined as the section between A and C. Geometrically the section between A and B is symmetrical, but it is not magnetically. Negative periodicity occurs between section A and D. Here there is geometric symmetry, but the poles are of opposite polarity. Positive result between A and E, again there is geometric symmetry, and the magnetic polarity is the same. Concentrating now on the rotor which is magnetised on its front face as a south pole, geometric periodicity is found between a pole pitch (E to G) at F. At position H we would not find geometric symmetry. However at I we would again find both geometric symmetry and positive magnetic symmetry. Within the software this symmetry boundary is considered as a single line. It is assigned a periodicity condition and a centre of rotation . For this example both stator and rotor could allow the condition of symmetry to occur at 180°, which allows half the machine to be modelled.

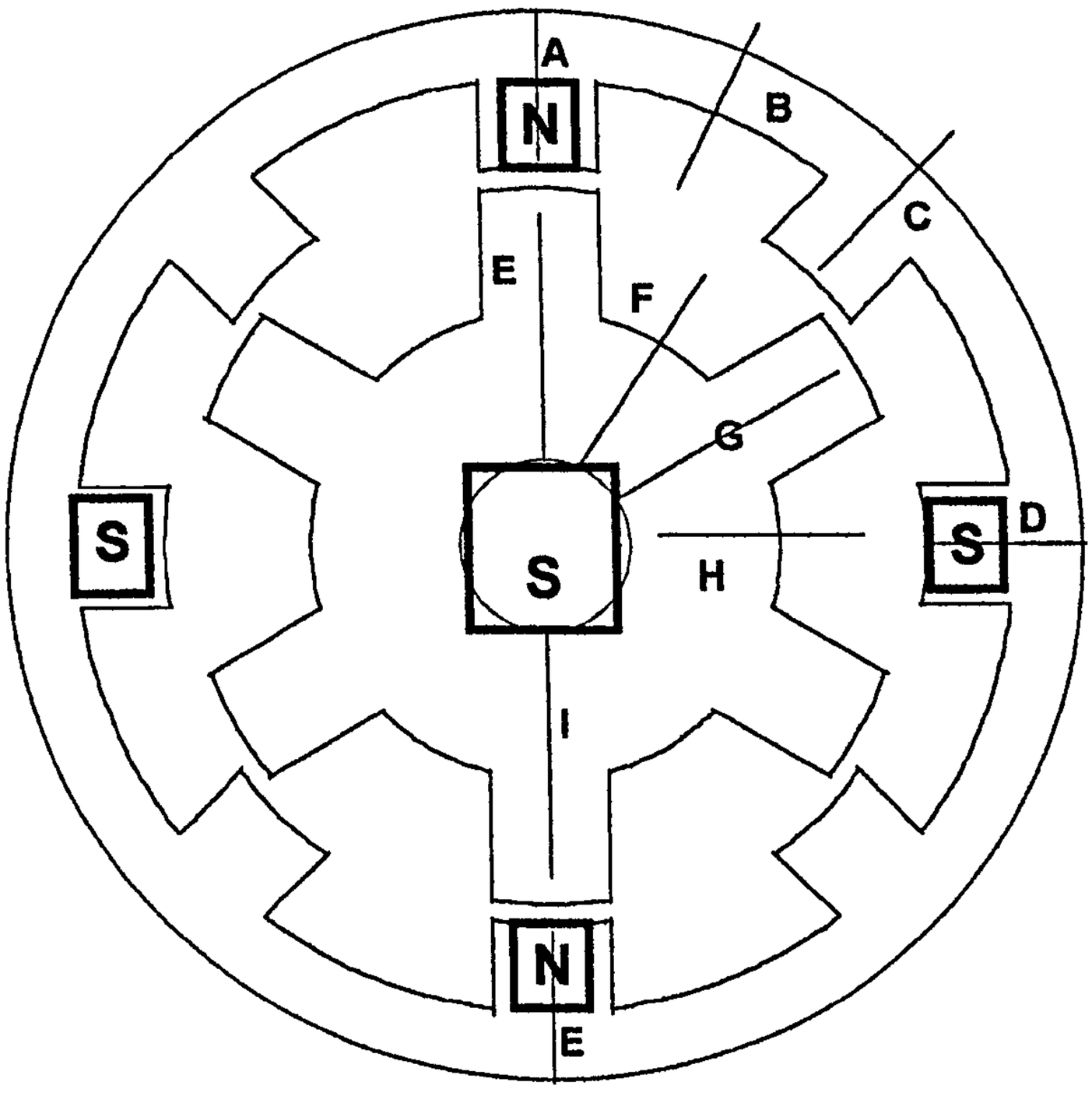


FIGURE 3.5 - Geometric and Magnetic Symmetry

3.4.9 Hybrid Stepping Motor Periodicity.

Because of the axially magnetised rotor magnet, flux travels down through the rotor and then in or out of the stator poles depending on the current direction in the windings. If one pole was pushing flux into the rotor, the directly opposite stator pole would be doing likewise. The flux would be travelling down the rotor and out through the stator poles that are attracting in flux. This type of action makes the positive periodicity correct for the hybrid stepping motor. By using positive periodicity half of the hybrid stepping motor needs to be constructed.

3.4.10 Magnet Definition

The material of the magnet in Stebon's hybrid stepper motors is typically Alnico and has been used in this example. The FEA software requires the magnet material's B-H curve to

be entered. From the second quadrant of the B-H curve, the software will calculate the permeability of the material by using the equation;

$$\mu = \frac{B}{(H-H_c)} \qquad (3.3).$$

Where μ is the permeability,

B is the flux density,

H is the field strength,

and H_c is the field strength coercivity.

The software starts with an initial value of μ. For linear problems this is constant. For non-linear problems the initial value used is the value of μ at B=0, i.e. the gradient of the B-H curve at B=0. The software then solves the equations and calculates values of the fields in the elements. The value of H is determined and B calculated from equation 3.3, with μ still the same value. The values of B and H are then compared to the B-H curve given. If they do not lie close enough to the B-H curve, a new value of μ is estimated using a Newton-Raphson scheme. The solution process is repeated using the new value of μ. The estimate of μ happens separately for every element in the mesh. This continues until all elements contain field solutions that match the specified B-H characteristics. This allows the load point of the magnet to be determined.

3.4.11 Construction of Model for the Simple Stepping Motor

The motor is constructed from the laminations used in a four phase switched reluctance motor as shown in figure 3.1. The rotor would be constructed by having two stacks of rotor laminations separated by an axially magnetised magnet. The rotor's teeth on each stack are shown to be mis-aligned by half a tooth pitch relative to each stack figure 3.6. Each layer is built up using the hidden detail method described in section 3.4.2. For this exercise the stator and rotor details are modelled together on the top pre-extrusion plane. This plane is then extruded to create several volumes. These volumes are defined as having particular electromagnetic behaviours in terms of materials and boundary constraints.

The rotor in this model is defined as having 5 distinct volume layers (figure 3.7). The air layers of 1, and 5, are outside boundaries, which simulate the air around the model. These limit the actual surrounding environment to a finite area which acts as a boundary to leakage flux. The limits of this area constrain the flux paths by the use of boundary conditions. In this case a 'Neumann' type boundary is set [48]. The stator has 3 layers defined, the second layer is the steel and has an identical inside node collection as that of the rotor's layers 2 to 4, figure 3.7.

The completed model consists of a rotor meshed by a common air-gap to the stator. Current carrying coils are positioned on the stator with the correct polarity. The complete model for solving is shown in figure 3.8.

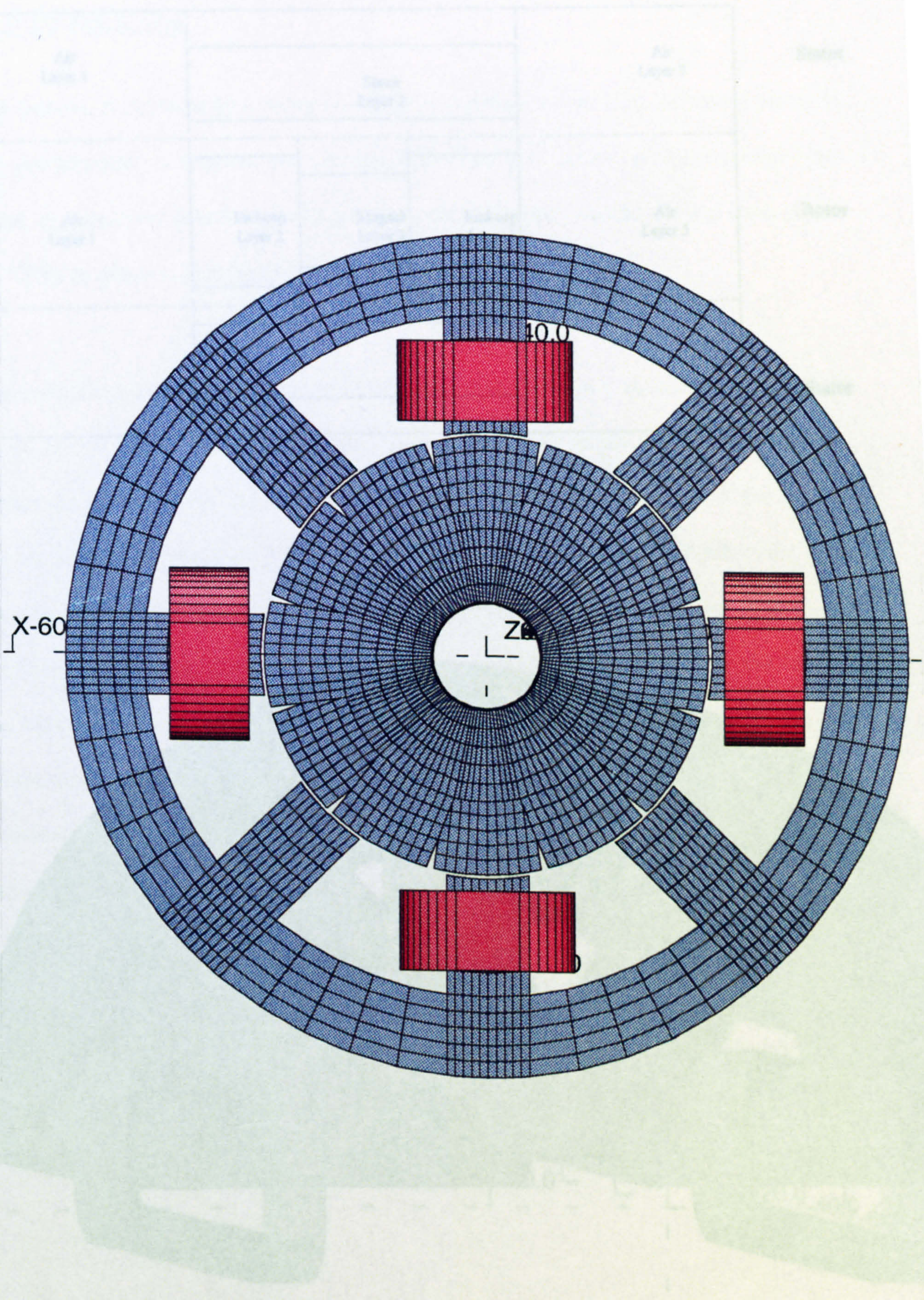


FIGURE 3.6 - Front On View Showing Rotor Teeth Misalignment

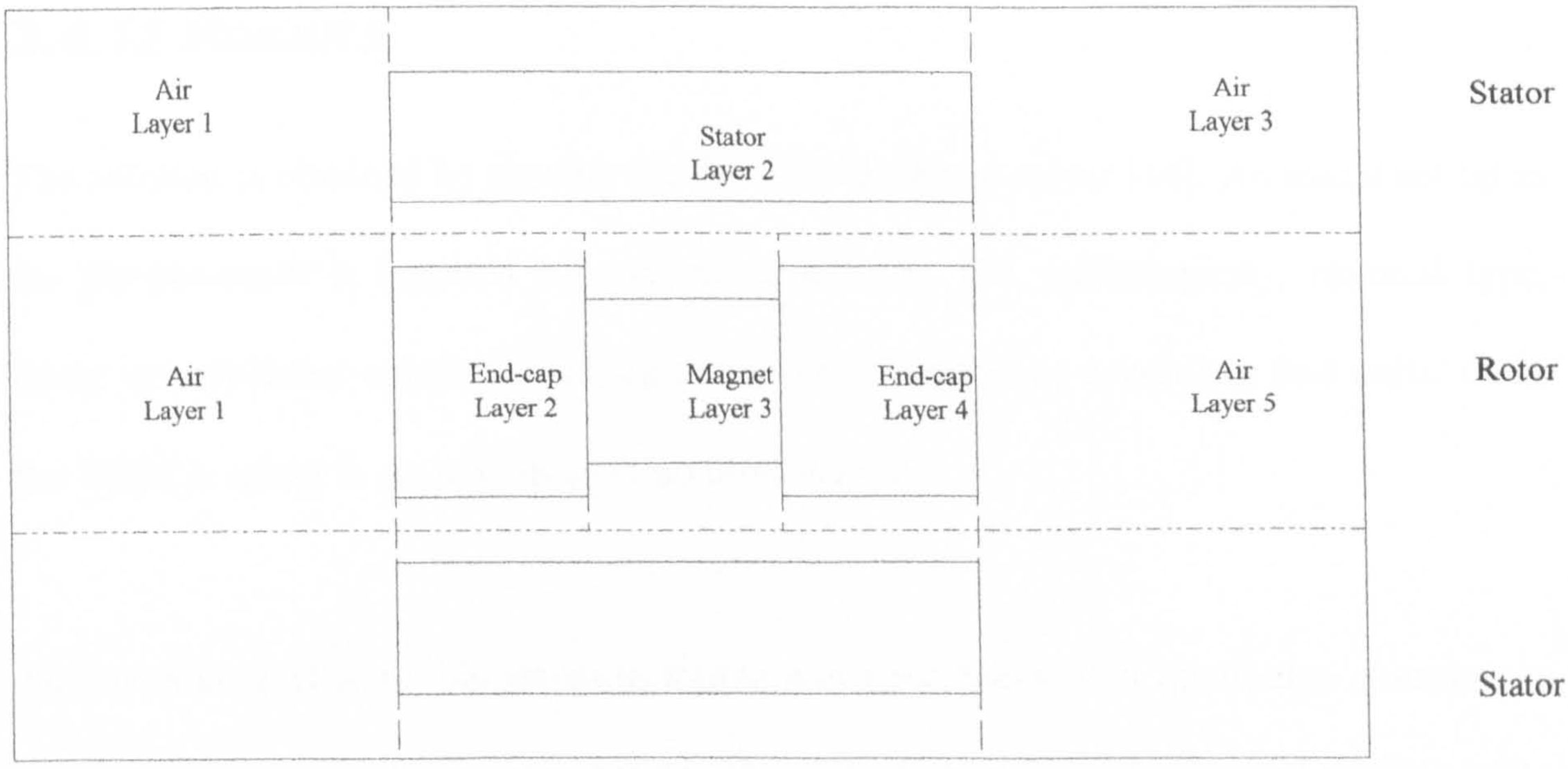


FIGURE 3.7 - Layer Definitions

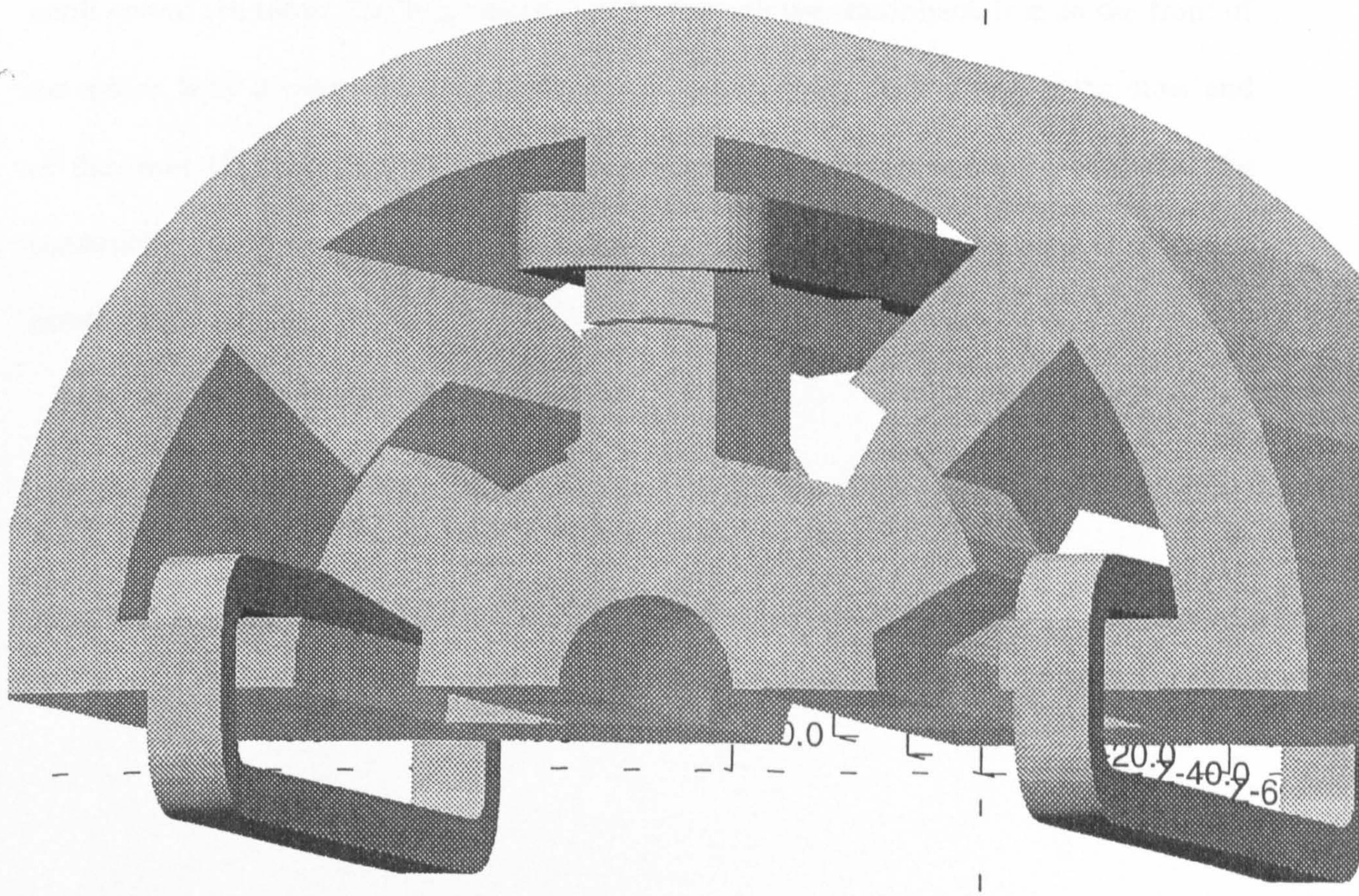


FIGURE 3.8 - Solid 180° Model of the Simple Stepping Motor

3.4.12 Results

The solution is obtained by running the TOSCA algebraic solver [16]. An initial set-up in the pre-processor is required for material conditions (i.e. anisotropicity, material type, linear or non-linear solution, packing factors, etc.) boundary conditions and units. Once the TOSCA solver is running there is no user interface.

The modelling of a simple stepping motor was undertaken as a qualitative exercise, to show that the flux paths are as expected. The post-processor results showed the flux paths within the motor by the use of flux vectors, (figures 3.9, 3.10.) The flux in the magnet travels south to north (figure 3.9). It then leaves the rotor by the aligned teeth of the rotor and stator. A south pole is induced in the stator poles aligned with the rotor poles in the north end of the rotor. The flux travels axially through the stator back iron to the front of the motor. Here it enters the rotor teeth at right angles to the lines of flux at the other end of the rotor (figure 3.10). The results for this simple stepping motor showed that the construction methods used in 3-D were correct. This methodology was used to model an actual hybrid stepping motor.

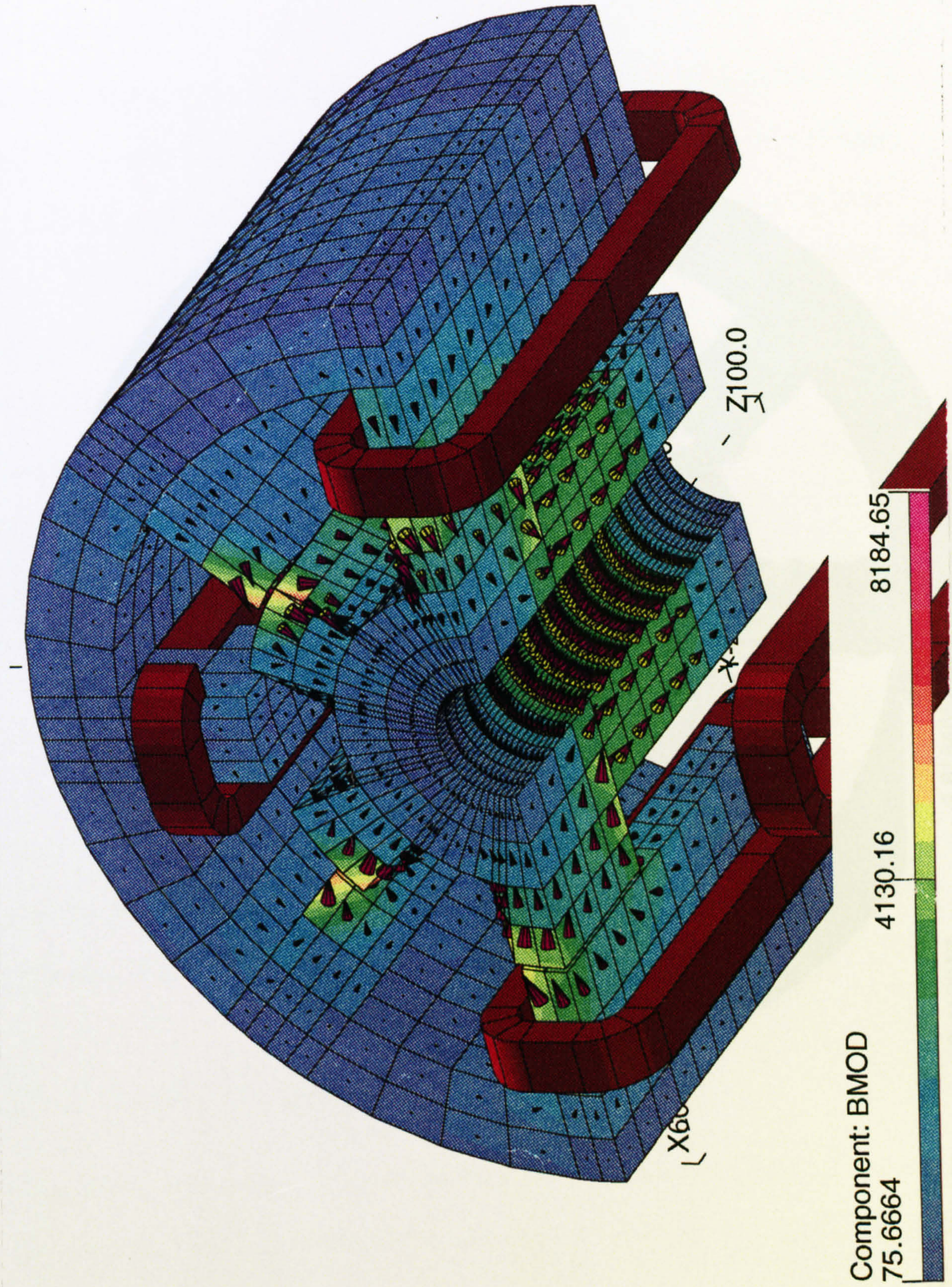


FIGURE 3.9 - Simple Stepper, under view

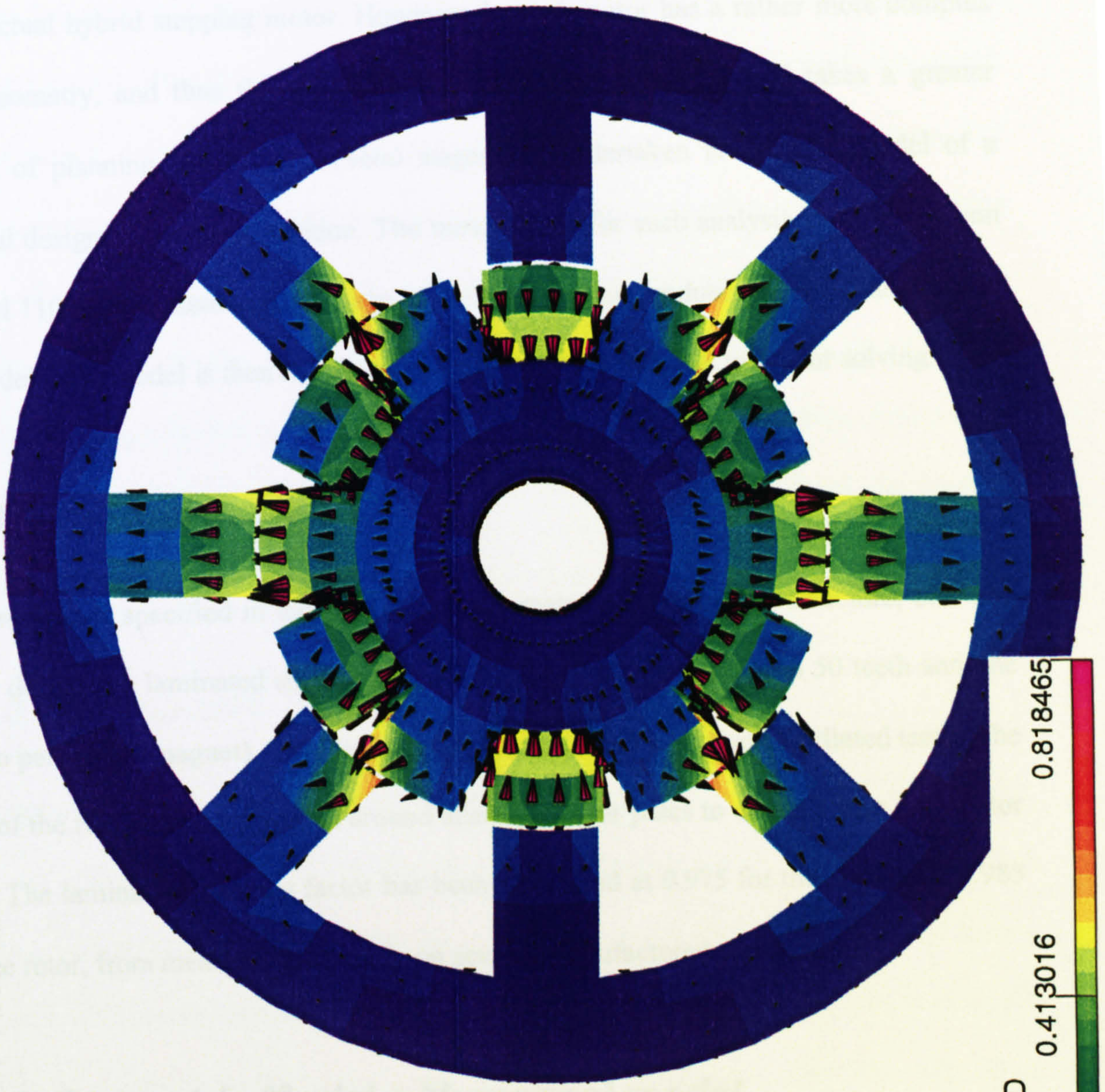


FIGURE 3.10 - Simple Stepper, Full view

3.5 Implementing a Model of an actual Hybrid Stepping Motor

The simple stepping motor examined previously does not differ in fundamental operation to the actual hybrid stepping motor. However the real motor has a rather more complex tooth geometry, and thus the modelling is rather more advanced and takes a greater amount of planning and effort. Several stages are undertaken to create a model of a practical design, ready for a solution. The motors used for such analysis were the Stebon 850 and 1100 series motors. Each stage of the construction produces a particular status of the model. This model is then redefined or appended to before it is ready for solving.

3.5.1. Physical Model

The problem is specified in terms of geometry, materials, lamination structure, etc. The motor design is a laminated single stack, (two rotor end caps, each with 50 teeth and one Alnico permanent magnet). The stator has eight poles each with five castellated teeth. The coils of the two phases are wound around alternate stator poles to create a four pole stator field. The lamination packing factor has been calculated at 0.975 for the stator and 0.985 for the rotor, from measurements made on several manufactured components.

3.5.2. Geometric Model + Numerical model

The physical model needs to be translated onto a two dimensional top plane. This plane is then sub-divided into 2-dimensional areas called facets. This plane is extruded into a three dimensional representation of the motor by creating volumes from each facet. These volumes are known as elements.

If computation time is not to be excessive, 100,000 elements is a sensible maximum limit and generally cannot be exceeded. For element numbers above 50,000, the original software required part of the model code to be modified outside the design environment. However the number of elements was still limited to 100,000 and hence simplifications to the model had therefore to be found which had a minimal effect on analysis. In the motor, areas of similar dimensions were made the same to reduce the number of elements. For example the magnet outer diameter was found to be 1 mm smaller than the diameter of the trough of the rotor teeth, and these were equated for ease of modelling. Elsewhere elements needed to be used effectively, allowing areas of greater interest or importance to be discretised (distribution of elements) finer. In particular it is essential that a fine mesh is used around the teeth and in the air-gap, where significant effects on the motor's characteristics occur.

3.5.3 Air-Gap and Tooth Discretisation

It is important that the rotor and stator be created as two separate models. This allows the rotor and stator to be joined together at different rotor positions. To achieve correct meshing between the stator and rotor the start of the model should use the same points definition in the centre of the air gap. This start of the mesh data file should be copied and be used as an identical start to both the rotor and stator definition files. This will allow error free meshing of the two models.

It is necessary to have at least two layers of elements in the air-gap and it is generally considered to be sensible to have no less than 4 layers of elements [48]. The dissection of the air gap radius should be done so that the nodes on the radius are at equal distances

from each other. The nodes are best defined as polar co-ordinates. This allows rotation of the rotor, with correct mesh linking between the nodes.

The biggest problem of creating rotor and stator models and meshing them together is that there can be far more nodes on the rotor outer diameter than on the stator inner diameter. This is because the rotor nodes are produced from two rotor end-caps, and rotor teeth nodes are found around the complete perimeter of the rotor. The stator nodes are concentrated at the bottom of the eight stator poles and there is less need to have nodes at the air-gap surface of the slots. The high number of nodes required on the rotor is illustrated in figure 3.11. In the motor there would be 50 teeth on each rotor end-cap, in the finite element top plane model there would be twice this number. Figure 3.11 shows how each layer needs to represent both overlapping sections of the front and back rotor teeth.

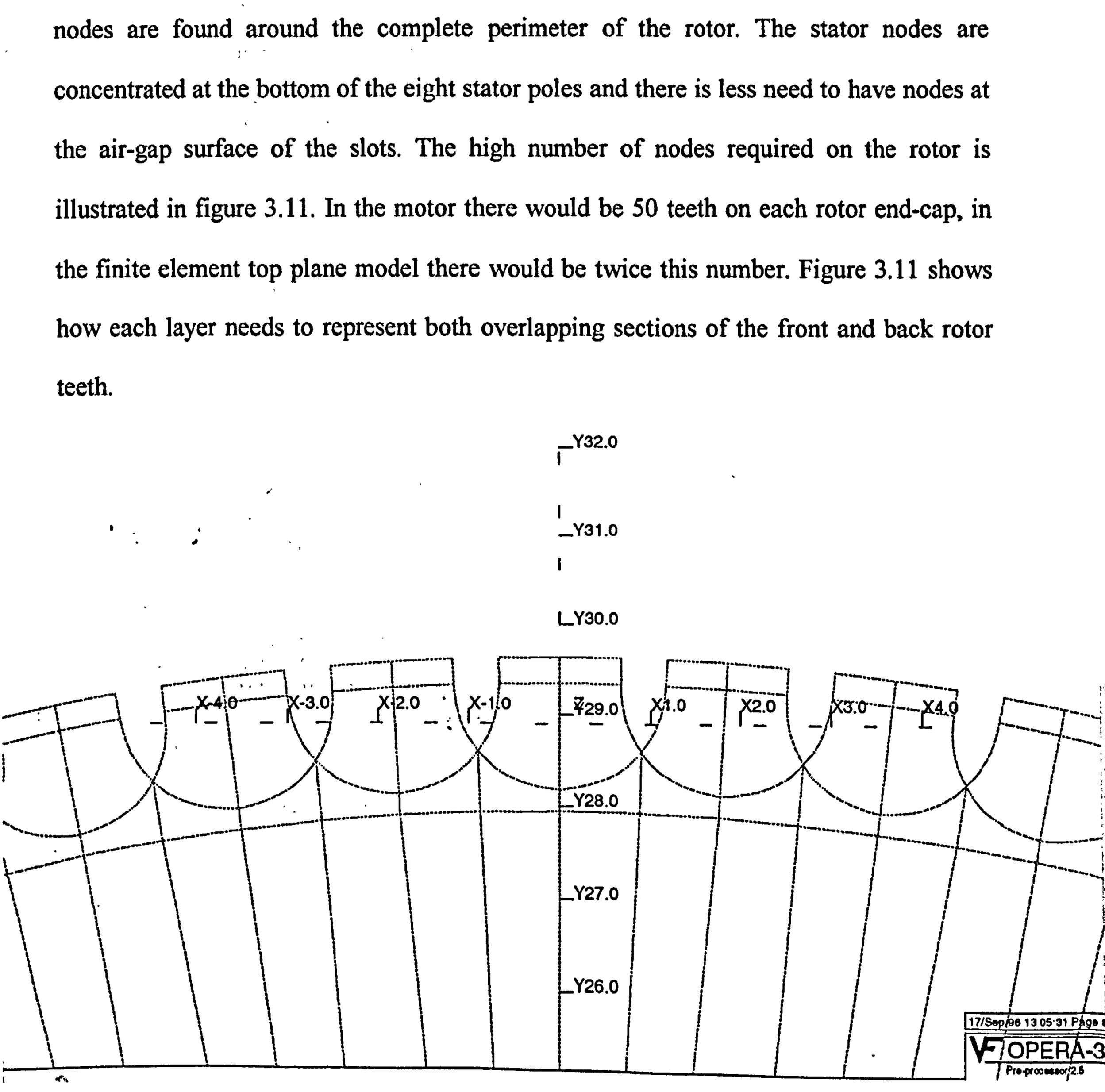


FIGURE 3.11 - Rotor Teeth Elements

It is possible to simplify the number of points on the rotor but this may cause the angle of rotation to become unsuitable. For a particular angle of rotation, the nodes on the pre-defined centre air-gap file must still match. If the hybrid motor was to be analysed simply as a single phase machine, a significant angle of measurement would be 1.8° as this is where maximum torque approximately occurs. Fortunately single phase analysis can take advantage of the fact that the stator poles are 1.8° misplaced from each other. Therefore by selecting appropriate phase excitation an aligned (0°), half aligned (1.8°) and unaligned positions (3.6°) are easily calculated before any specific angle discretisation is required. In single phase analysis 1.2° degree discretisation is a good angle as this produces a low rotor to stator node ratio. It is relatively easy to implement and from it 0°, 0.6°, 1.2°, 1.8°, 2.4°, 3°, and 3.6° angles of rotation can be analysed. However this not completely convenient for two phase analysis. In two phase analysis the maximum torque approximately occurs at 0.9° rotation from an aligned top pole tooth position, as shown in figure 3.12.

If discretisation of the air-gap was 0.9° the rotor to stator node ratio would increase. The angles which could be analysed would be 0°, 0.9°, 1.8°, 2.7°, and 3.6°, a reduction from seven to five angles when compared to the single phase case. It is not really practical to take the angle of rotation any smaller because the ratio would increase significantly.

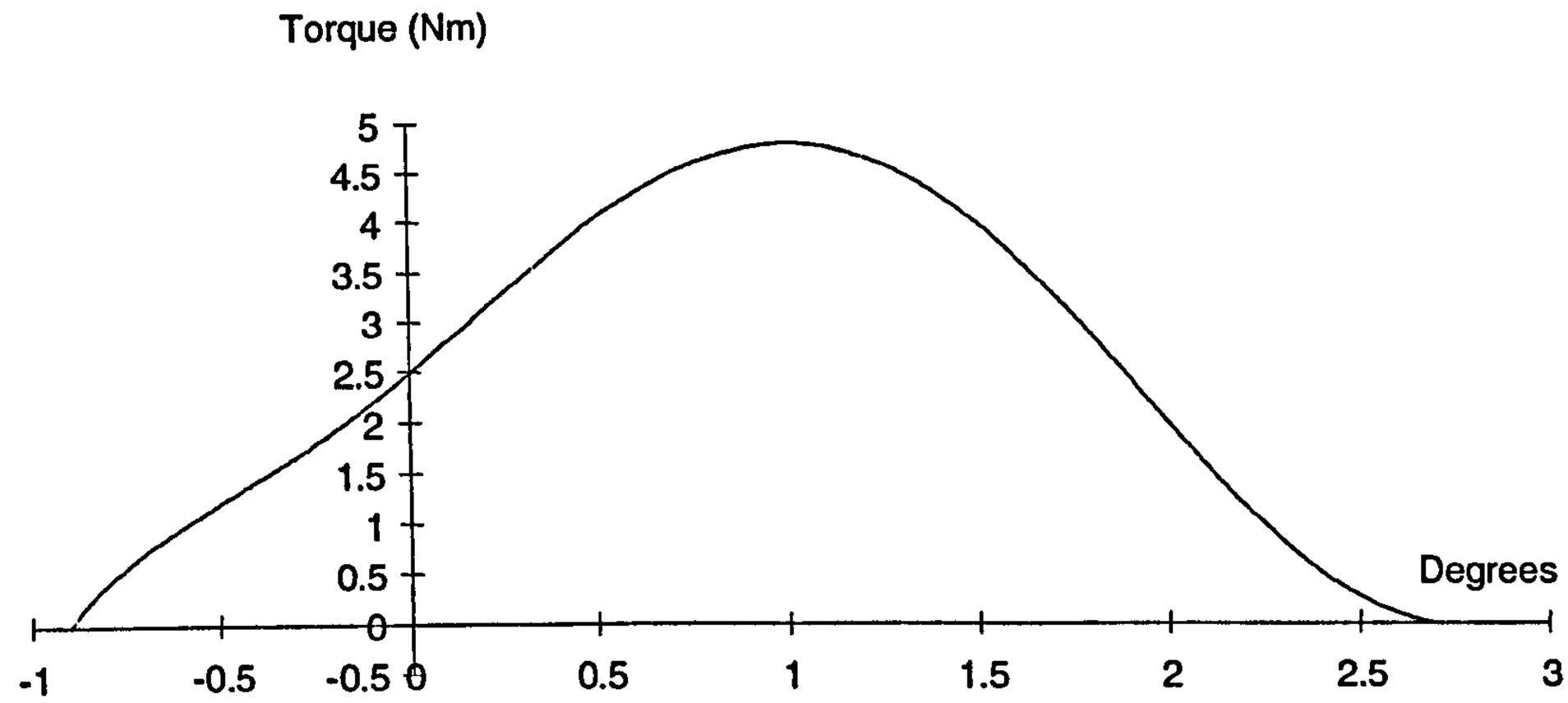


FIGURE 3.12 - Two Phase Torque Representation

The modeller has a choice of either using a 1.2° angle rotation with sine wave interpolation to obtain intermediate results, or to use triangulated elements to obtain 0.9° rotation. The modeller could however if time permitted use two models. In the available software at the start of the project, problems arose with using a discretisation finer than 1.2°. A model created later in the project used 0.9° discretisation for improved two phase analysis when the software was more robust and gave an increased number of elements.

3.5.4 Cogging Torque.

Figure 3.13 shows the cogging torque approximation of a hybrid stepping motor. There are four cycles per tooth pitch. The position of equilibrium occurs 400 times a revolution, with 200 being stable points. If the 0.9° angle discretisation was used, then the torque analysis using Maxwell stress calculation would only indicate the positions of equilibrium. With the 1.2° angle maximum torque would not be directly calculated but other points could be calculated and with the use of Fourier point analysis a maximum torque could be predicted. The analysis would be better suited to a 0.45° rotor movement as this would

allow the 0.9° maximum torque angles to be approximated by a sinewave. The cogging torque indicates a fundamental problem of hybrid stepping motor analysis. Here the modeller needs to consider the excessive computing time, or a reduced analysis.

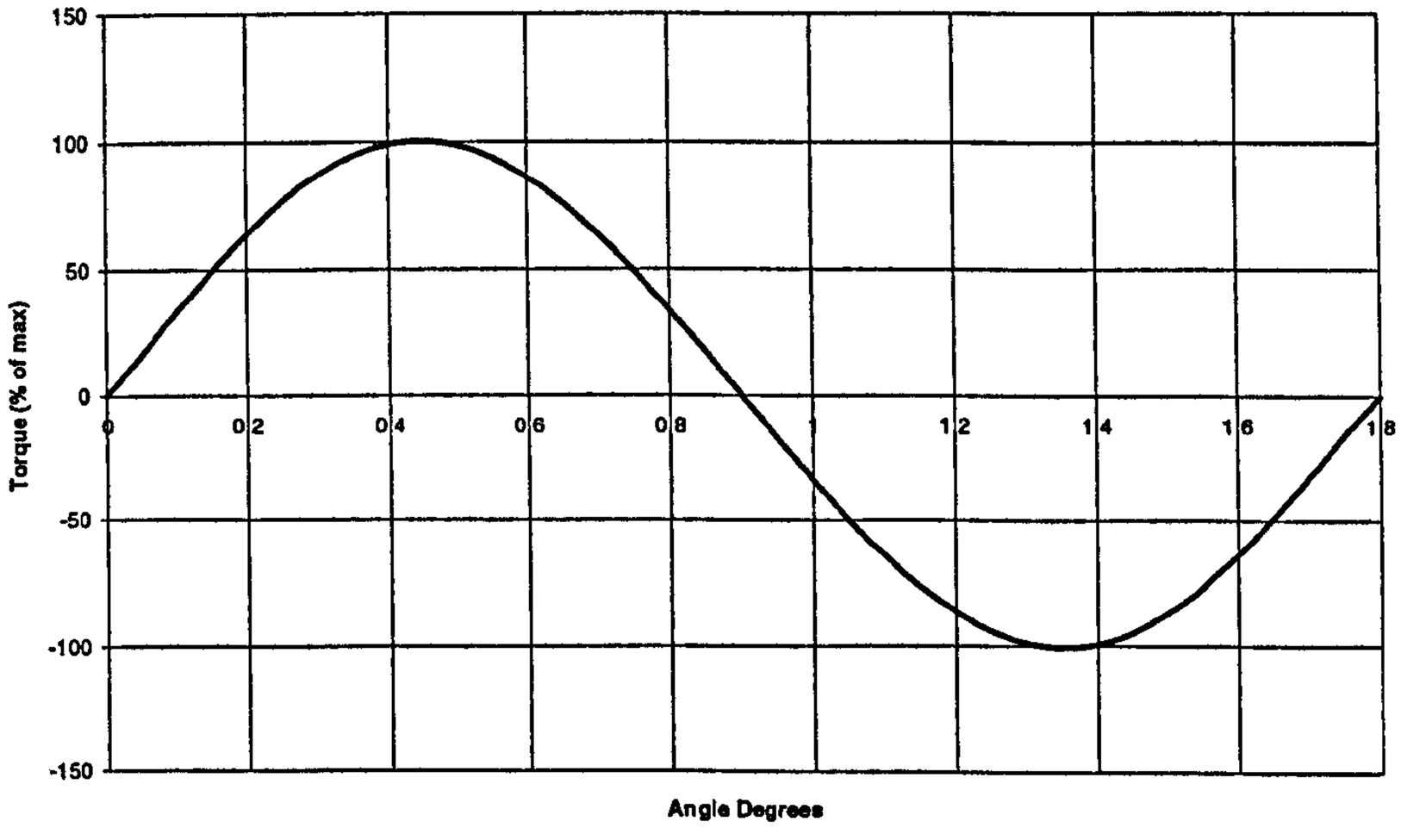


FIGURE - 3.13 - Cogging Torque Assumption

3.5.5 Reduction Method for Elements

In some areas of the model there is a need to reduce the number of nodes. Figure 3.14 displays a technique for reducing the number of nodes. This element-grading technique can be used elsewhere in the model.

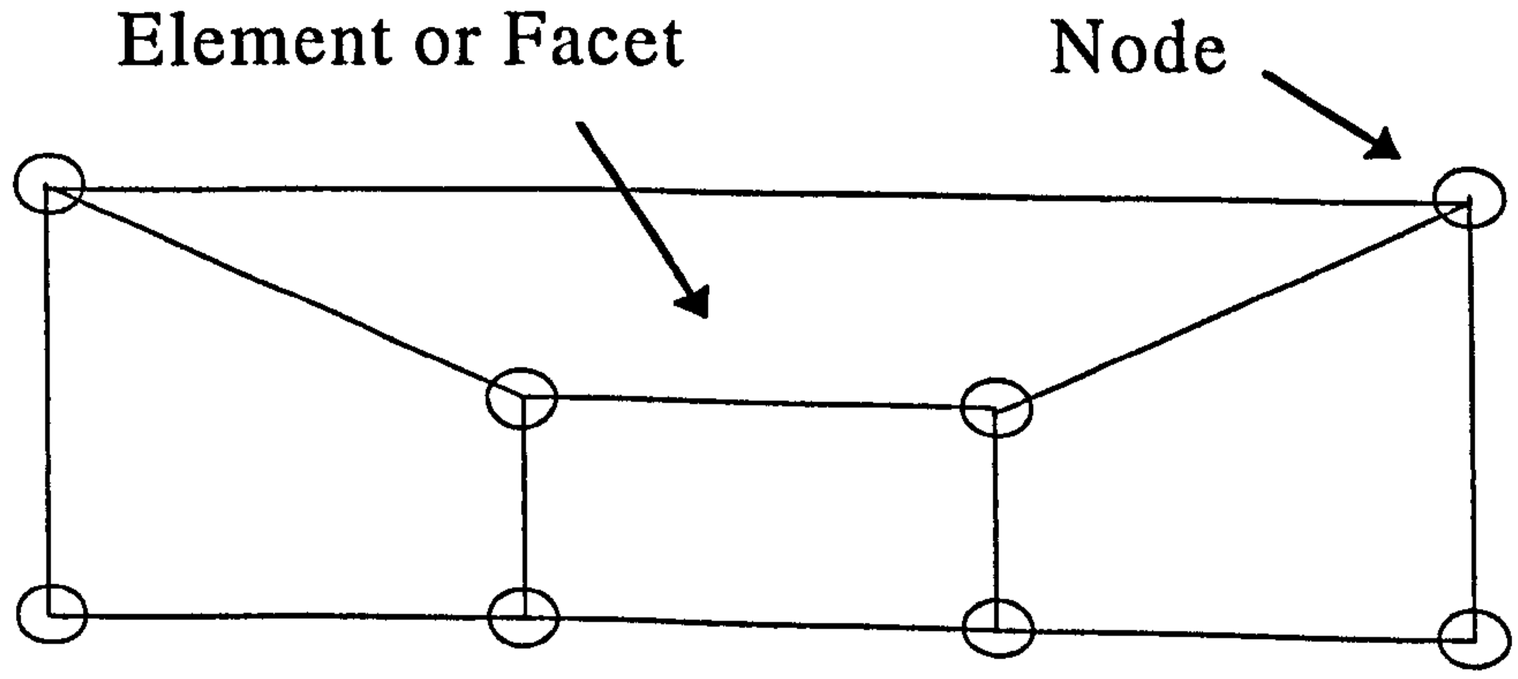


FIGURE 3.14 - Reduction technique for reducing 4 nodes to 2

3.5.6 Material Definition

Material definition assigns a name and a material B-H characteristic to a particular volume. It is wise to call the material by a location name as well as the material type, i.e. stator iron. The modeller may also choose between a linear and non-linear analysis. In the hybrid stepping motor the analysis is required to be non-linear as parts of the steel in the motor become highly saturated. The magnet material is defined by entering a B-H curve, from which the software calculates the magnet's working point, as described in section 3.4.10. This working point will change depending on the phase winding excitation and position of the rotor. The magnet material in the motor is axially orientated and requires a vector orientation to define this. Axial magnets are fortunately defined as having a vector orientation of 0 0 0. The numbers represent Euler angles [48].

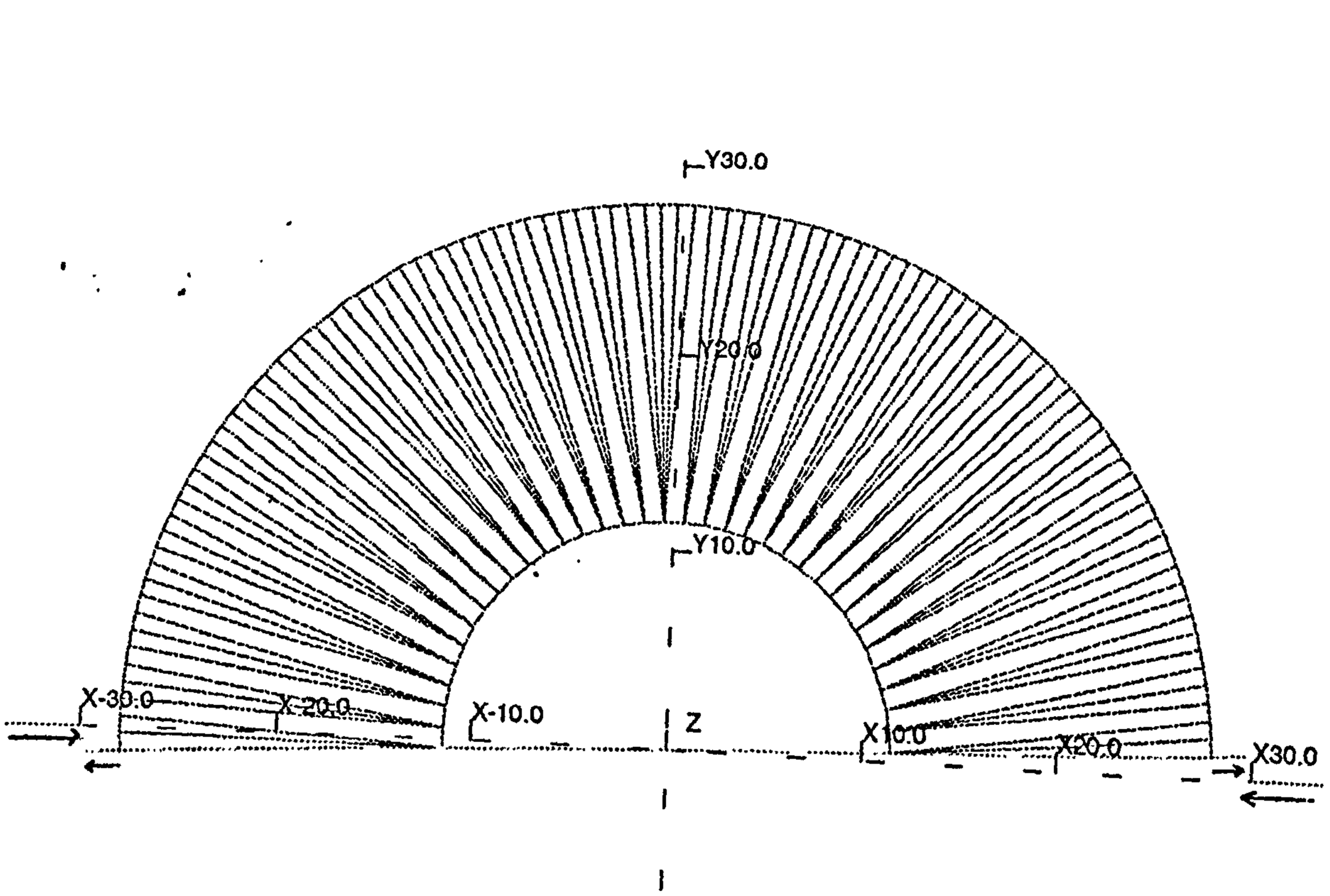


FIGURE 3.15 - Broken Boundary Condition

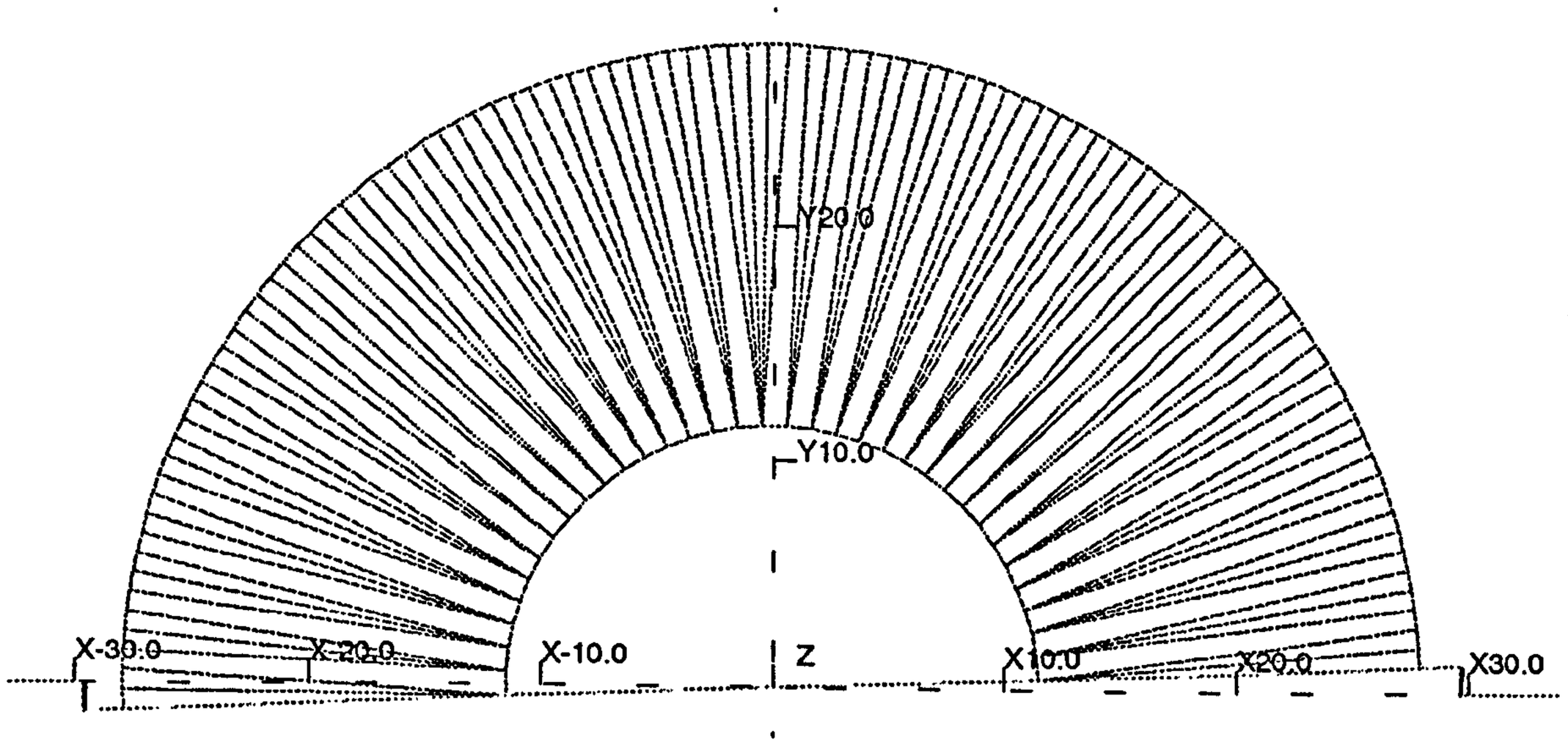


FIGURE 3.16 - Completed Boundary Line

3.5.7 Mathematical Model

Here the choice of solution potentials and boundary conditions are defined. The modeller can also takes advantage of any symmetry in the model. The boundary and symmetry conditions which are used in the hybrid stepping motor have been explained in section 3.4.

A rotation of either the rotor or stator relative to the other can be easily achieved. However, when a 'half symmetry' model is used the rotation of a mesh will cause a discontinuity in the symmetry boundary. The modeller is required to complete the boundary line. The boundary discontinuity is shown in figure 3.15. The rotor has been rotated by 2.4°. The left hand side break of the model is by the X-axis '30' position and this is required to be completed by adding a symmetry condition on the rotor mesh.

Similarly the right hand side is completed by adding a symmetry condition on the stator mesh. The completed symmetry condition is shown in figure 3.16.

3.5.8 Conductor Model

This creates the current carrying conductor regions of the model. In the OPERA-3D package conductors are defined separately to the rest of the model. The current in a conductor is set by inputting a current density in the cross sectional area of a conductor. The current density J is calculated by;

$$J = \frac{NI}{A_{cond}}, \qquad (3.4)$$

Where J is the current density,

N is the number of turns,

I is the current,

and A_{cond} is the cross-sectional area of the conductor.

3.5.9 Complete Model

In general a complete model is made up of three separate files, one each for the stator, rotor and conductors. Typically the stator is loaded first, then the rotor and finally the conductors. However the rotor file may require modifications if a particular rotor rotation is required. The simplest method of achieving the preferred rotor angle is by editing the rotor file. Near the start of each file is a line which defines the co-ordinate system for the model. In its simplest form the line will be described as:

$x\ y$

Referring to an X-Y co-ordinate system. The rotor file should be edited and have the line replaced by:

new | corsys $\#_x$ $\#_y$ $\#_z$ θ_x θ_y θ_z

Where '*new*' indicates a new co-ordinate system,

| a system return code,

corsys, a name for the new co-ordinate system,

$\#_x$ $\#_y$ $\#_z$ are the rotation in the X, Y, and Z axes of the new co-ordinate system relative to the old system,

and θ_x θ_y θ_z which will be a position in degrees for the rotation.

An example to achieve a rotor angle of 1.2° clockwise, for a co-ordinate system called *ROTOR*, the line would read:

new | ROTOR 0 0 0 0 1.2 0

Once the rotor file has been read in the user is required to complete the broken symmetry line as described in section 3.5.7. A 180° rotor model is shown in figure 3.17. A two dimensional mesh of the joined stator and rotor is depicted in figure 3.18.

Once all files have been read in the pre-processor will have a complete model. Figure 3.19 shows 180° of a completed model, figure 3.20 a perspective view and 3.21 a close up of the motor.

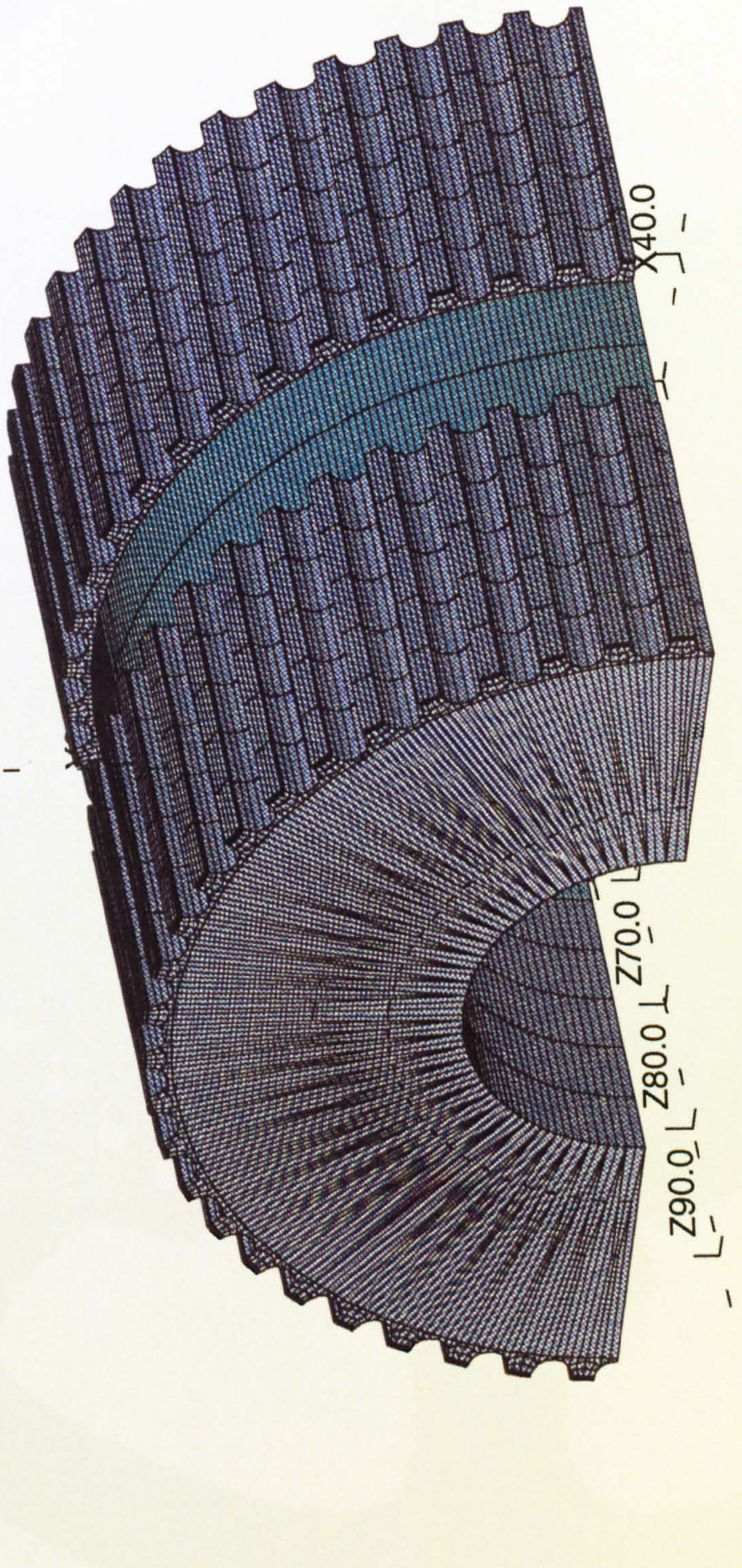


FIGURE 3.17 - Half Rotor Model

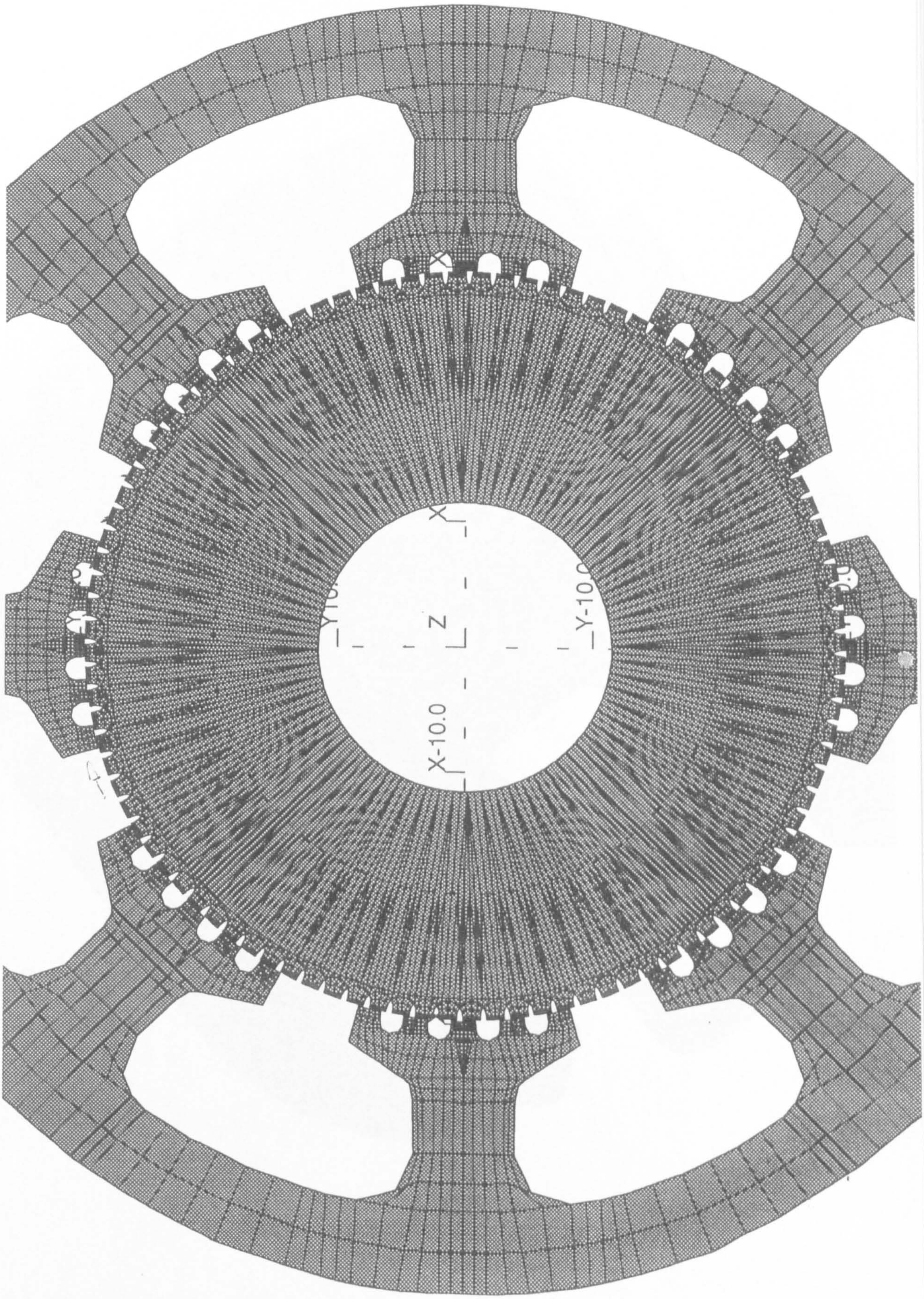


FIGURE 3.18 - 2-D joined Rotor and Stator

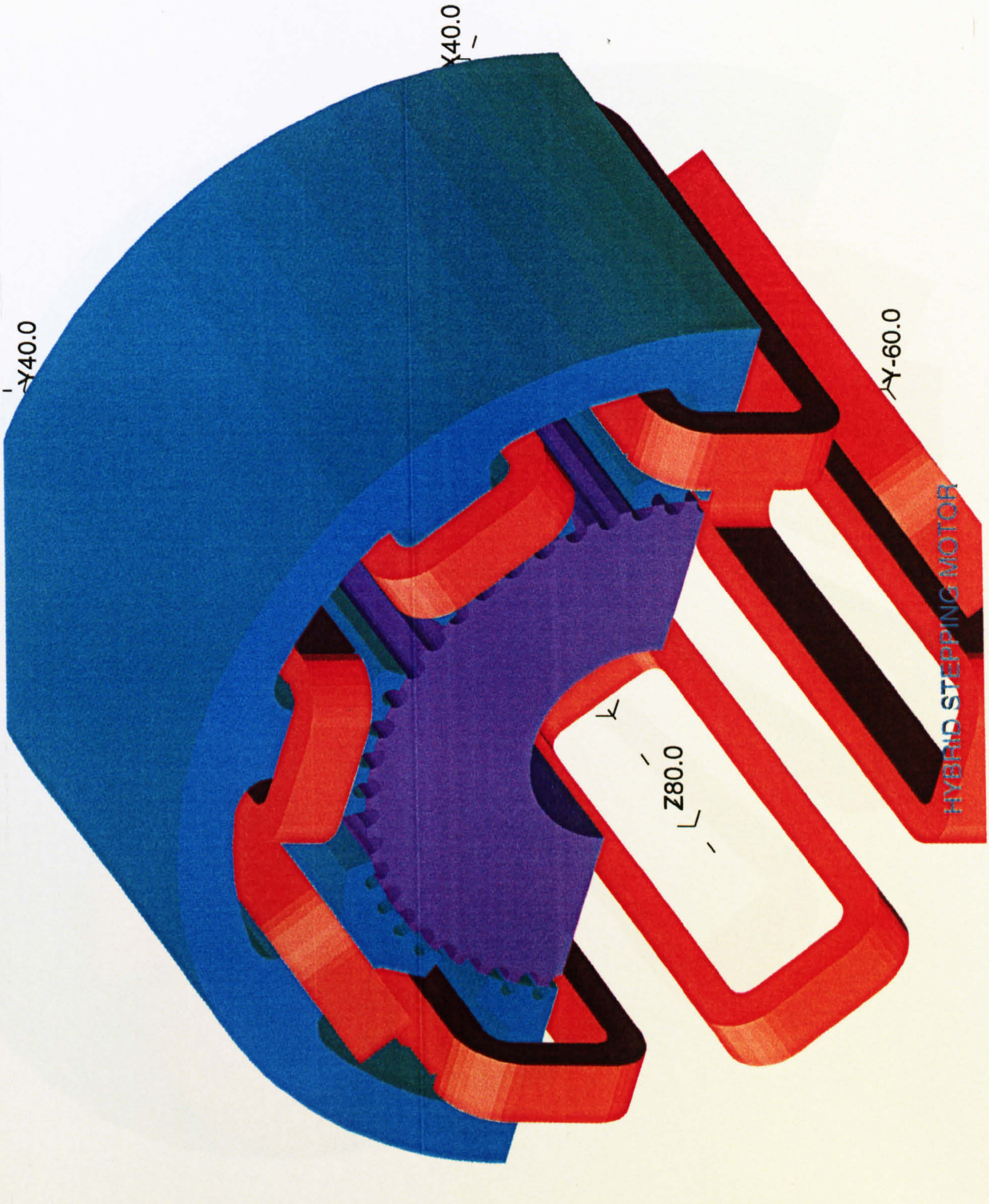


FIGURE 3.19 - Completed Half Model

FIGURE 3.20 - Two phase 2-D (360° by Reflection) Complete Model

FIGURE 3.21 - Close up 3-D Complete Model

3.5.10 Algebraic Model

Before the field equation solver (TOSCA) [15] can be run for the calculation of the solution potential, the user is required to select certain factors that define the type and accuracy of the solution. These include iteration count, convergence tolerance, type of solution (Newton-Raphson / simple up-date), and the way the solver deals with materials (linear / anisotropy).

To solve for the hybrid stepping motor, a non-linear, laminated material solution was used. The convergence tolerance was set at 0.001 with Newton-Raphson solution type. The maximum number of iterations was higher than the default of 15 and was increased to 40. This was because when the iteration count was 15, the numeric solution had sometimes not converged.

3.6 Analysis of the Un-Excited Motor

When no current is in the motor phase windings, the flux circulating throughout the motor is generated solely by the rotor's permanent magnet.

The significant flux paths in the magnet excitation can be classified as a series of loops emanating from one source. Starting from the magnet the flux enters the rotor end-cap. The inner flux paths travel the longest lengths to the outer limits of the end-caps, whereas the outer flux vectors in the magnet travel the shortest distances. As the flux takes the most permeable path, the majority of flux crosses the air-gap closest to the magnet, hence avoiding crossing the laminations. This is demonstrated by the low value of flux density

on the outer faces of the rotor end-caps. The flux will travel mainly across the air-gap at an aligned position. This position is quickly saturated, thus flux will take the next permeable option, flowing across the semi-aligned teeth. Some of the flux will cross the air-gap at the un-aligned position, but mainly towards the centre of the rotor, where the path reluctance is comparable, to travelling through the laminations. Once in the stator, the flux on the outer edges is confined to travelling up the laminations towards the back iron. It then gradually flows in the Z-direction to enter the rotor again at the appropriate aligned, mid or un-aligned position. This is again dependent on the saturation of the stator teeth and the MMF dropped by travelling across the laminations.

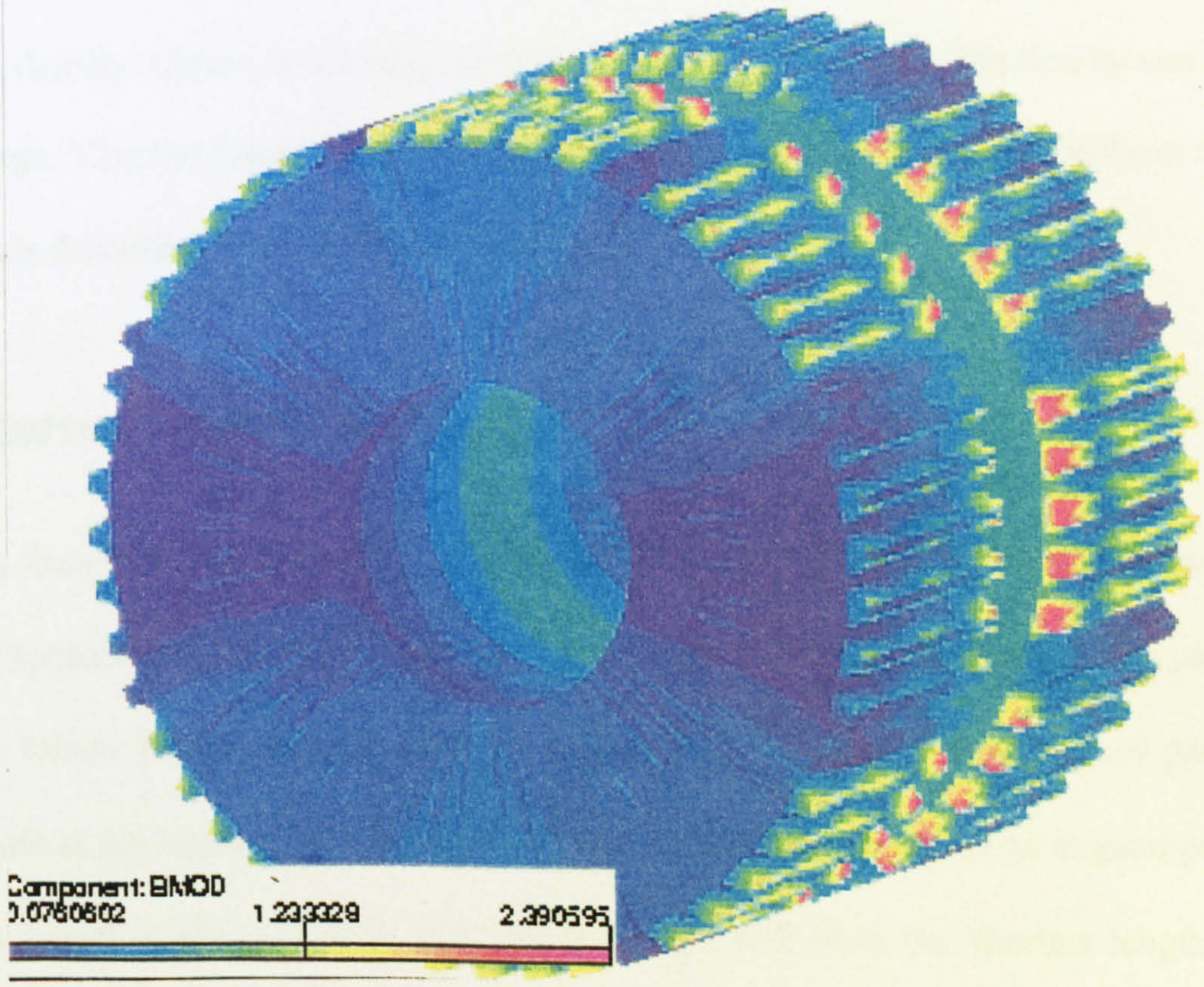


FIGURE 3.22 - Rotor Excited by Magnet Flux

Figure 3.22 is a three dimensional representation of the rotor showing distribution of flux density. Even though the flux is generated solely from the magnet, comparable high areas of flux density can be seen on the inner edges of the rotor end-caps. The highest areas are found on the teeth edges that are most aligned with the stator pole castellations. The lowest areas on the teeth are found where the teeth are totally unaligned and on the middle sections of the end-cap. Fringing is commonly found on the edges of the teeth. The semi-aligned teeth reveal a balanced flux flow between them and this is mirrored around the magnet interface. The section of rotor located between the stator pole castellations exhibit little leakage flux paths.

The centre of the rotor end-caps faces on the front and back of the motor show the lowest areas of flux density. Closer to the rotor end-cap /magnet interface, the flux density can be relatively large. This has been found to be because of the way flux flows internally in the rotor, which is described in the following paragraph.

3.6.1 Internal Rotor Flux Paths

Considering lines of flux are a useful way of visualising the flux distribution in the motor though it is appreciated that they do not have a physical meaning. The internal flux paths in the rotor follow helical shaped routes. The figure of 3.23 shows three typical paths. Route A starts at the inside edge of a rotor end-cap, on the centre tooth of an aligned pole. It has been found in the 3-D FEA that this flux path will have the shortest length. It therefore needs to reach a point close to the outer diameter of the permanent magnet. The flux path needs to obtain an equal flux length each side of the magnet so the path lengths

in either end-cap are balanced. In the magnet itself the flux line is confined to a straight line of the magnet orientation.

In path *B*, the flux path starts on the same tooth as path A, but on the outer edge of the end-cap. It consequently has a longer Z-direction distance to travel. Its position of contact with the magnet face is much nearer the centre. The longer the Z-direction distance the closer to the centre of the magnet the flux path will enter. If a flux vector analysis was taken half-way into a rotor end-cap on the X-Y plane, flux paths closer to the inner diameter would be highly Z-directionally biased.

Path C starts on the centre of a tooth which is aligned with a side end tooth on the stator. In this case the flux path needs to meet the magnet interface (X-Y) at approximately the centre. The flux path will flow to the equivalent tooth on the other end-cap. The flux will always flow to the near side corresponding air-gap crossing on the other end-cap. This is true of all flux paths.

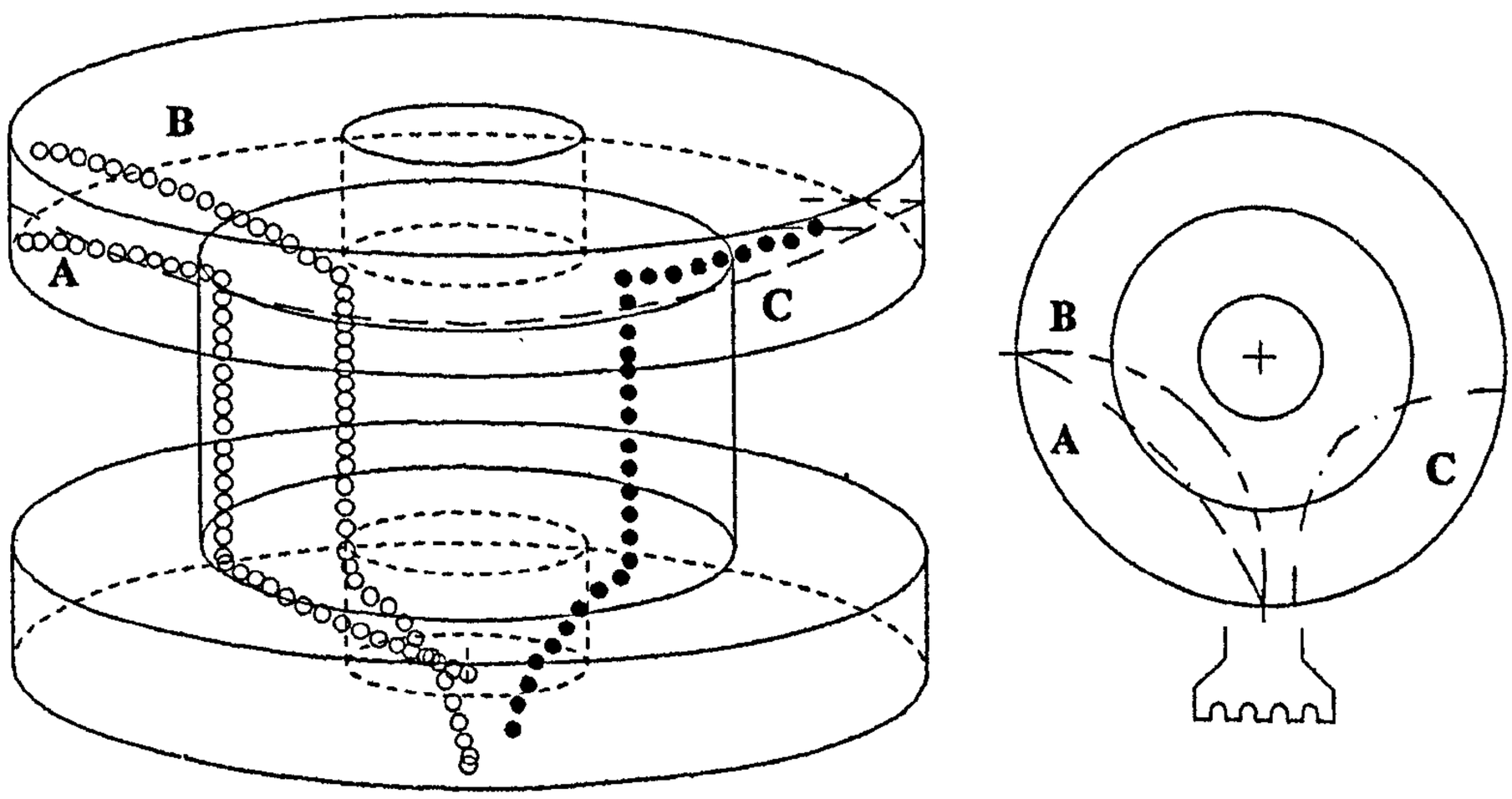


FIGURE 3.23 - Magnet Flux Paths

3.6.2 Analysis of the Air-gap Region

Figure 3.24 shows a close up of an air-gap region. With magnet excitation alone the flux density reaches a relatively high value only on the tips of the teeth. The flux is constrained within the teeth and flows up through the stator pole. The majority of the flux flows into the corresponding aligned stator pole and the two poles either side. However a large proportion of the flux does flow through the middle aligned pole. On the back centre pole (which is unaligned) a significant amount of flux leaks back into the rotor. This indicates that the un-aligned path is still relatively permeable because the laminated stator has a significant reluctance in the axial direction.

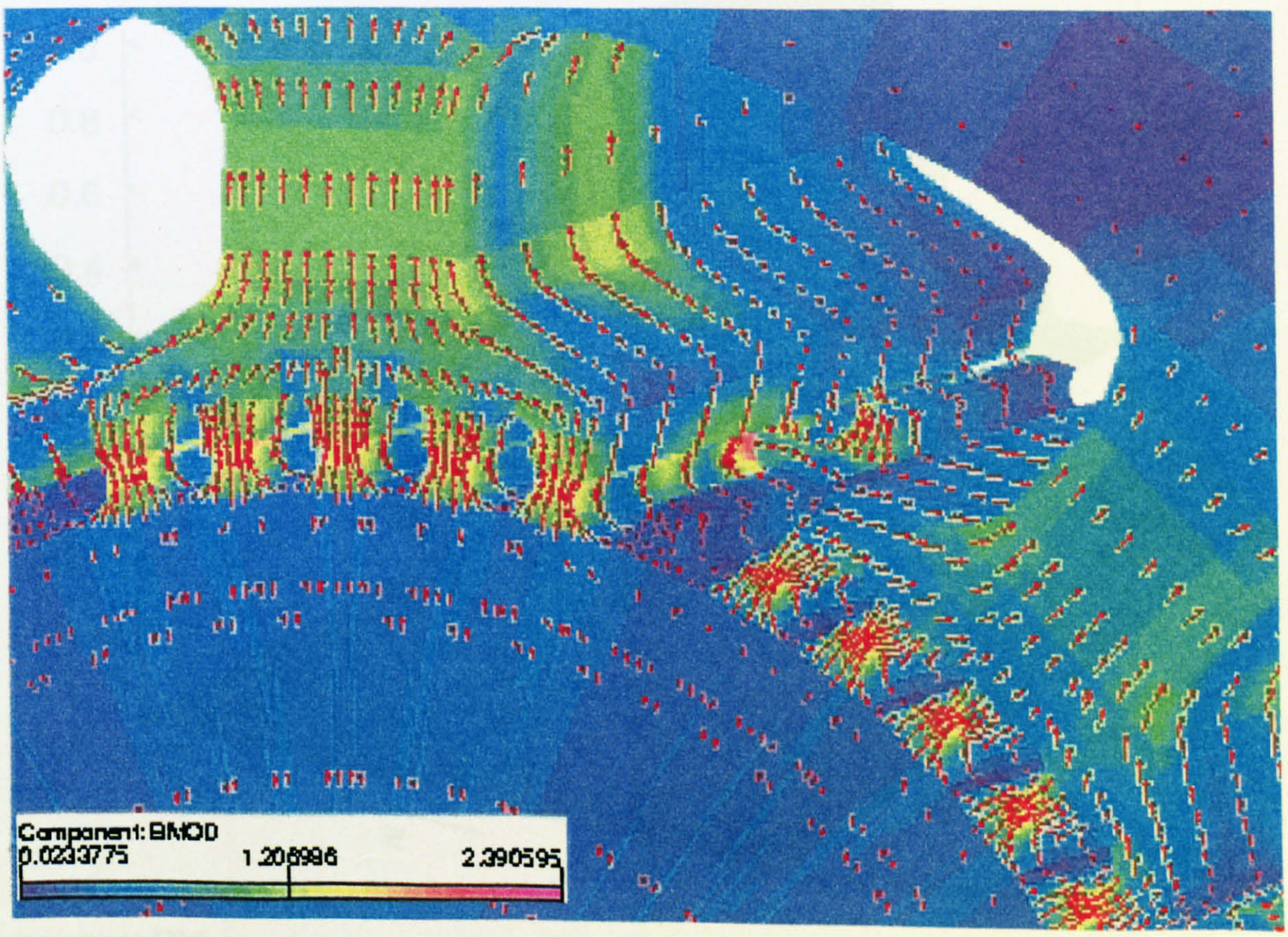


FIGURE 3.24 - Air-gap Crossing

To illustrate the flux distribution across the aligned stator refer to the graph in figure 3.25. This graph represents a Y- plane (constant height), where the Z-axis runs from points 1 to 4, and the X-axis is from 1 to 2. The Y-axis height is half way up the stator pole. Here a near sinusoidal distribution in terms of magnitude is illustrated. The flux density is highest on the magnet side of the rotor end-cap length. The polarity quickly reverses at the centre of the magnet. A fringing step is found at the outside edge of the stator, where the steel meets the air boundary. On other FE studies the smoothness of the distribution, and height of this outer step is highly affected by the packing factor of the steel.

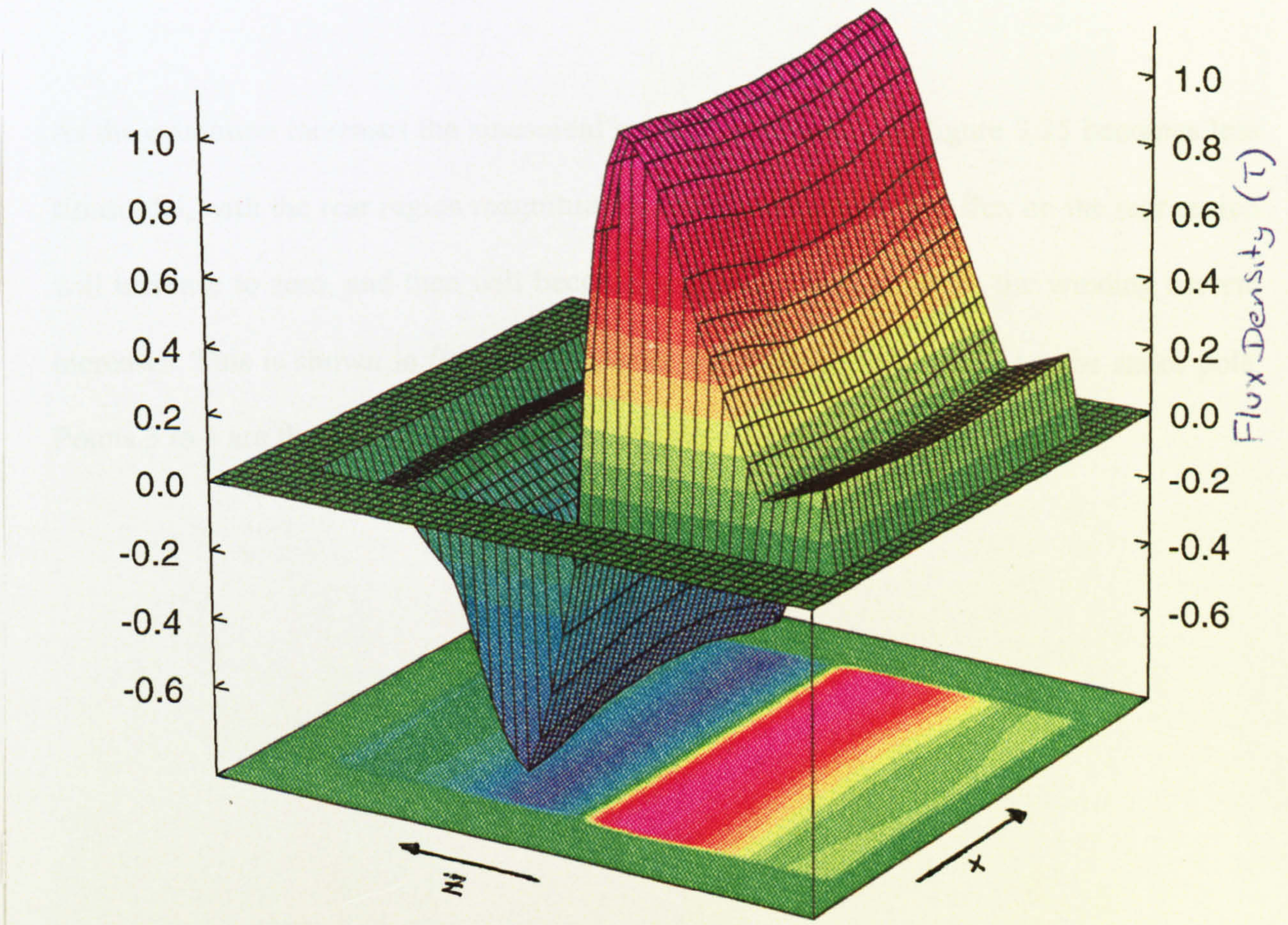


FIGURE 3.25 -Plot of Flux Distribution in Unexcited Stator Pole

3.7 Single Phase Analysis

At low excitation the single phase flux distribution is essentially similar to that of the magnet only analysis. However as the current in the phase is increased, certain characteristics of single phase flux distribution emerge. Figure 3.26 shows a close up on a stator/ rotor section. Here flux is flowing from the magnet into the front end-cap and is pulled into the aligned teeth by a low current excited winding. The phase winding is not shown for clarity. Flux flowing directly back into the un-aligned position at the back is now opposed by the winding polarity, hence more flux will flow upwards toward the back iron.

As the excitation increases the sinusoidal magnet distribution of figure 3.25 becomes less sinusoidal, with the rear region magnitude reducing. Eventually the flux on the rear region will increase to zero, and then will become positive in magnitude as the winding current increases. This is shown in figure 3.27. Again the Y-plane is half way up the stator pole. Points 3 to 4 are the X-axis, and 3 to 2 the Z-axis.

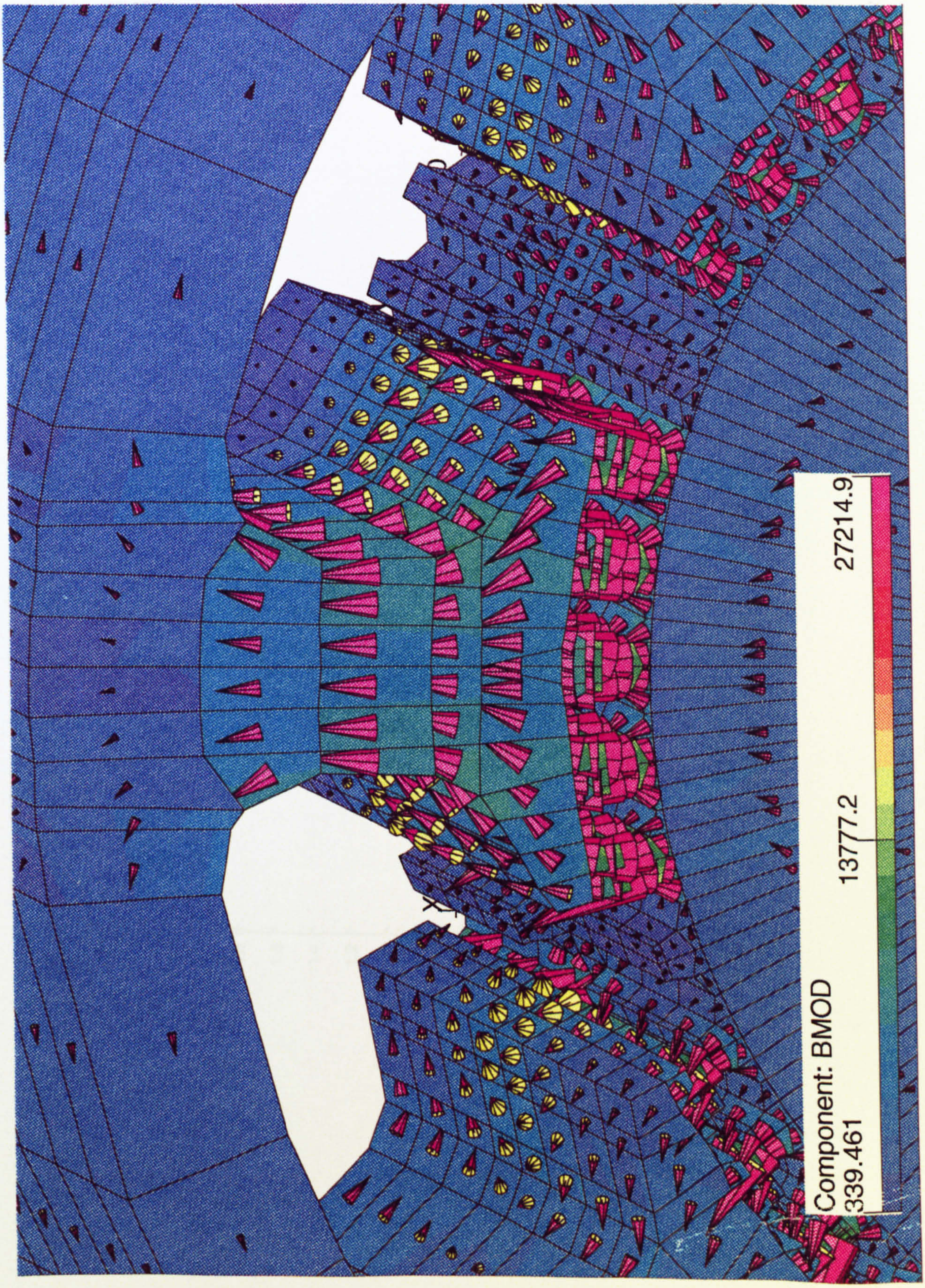


FIGURE 3.26 - Close up of Rotor Teeth and Stator Pole

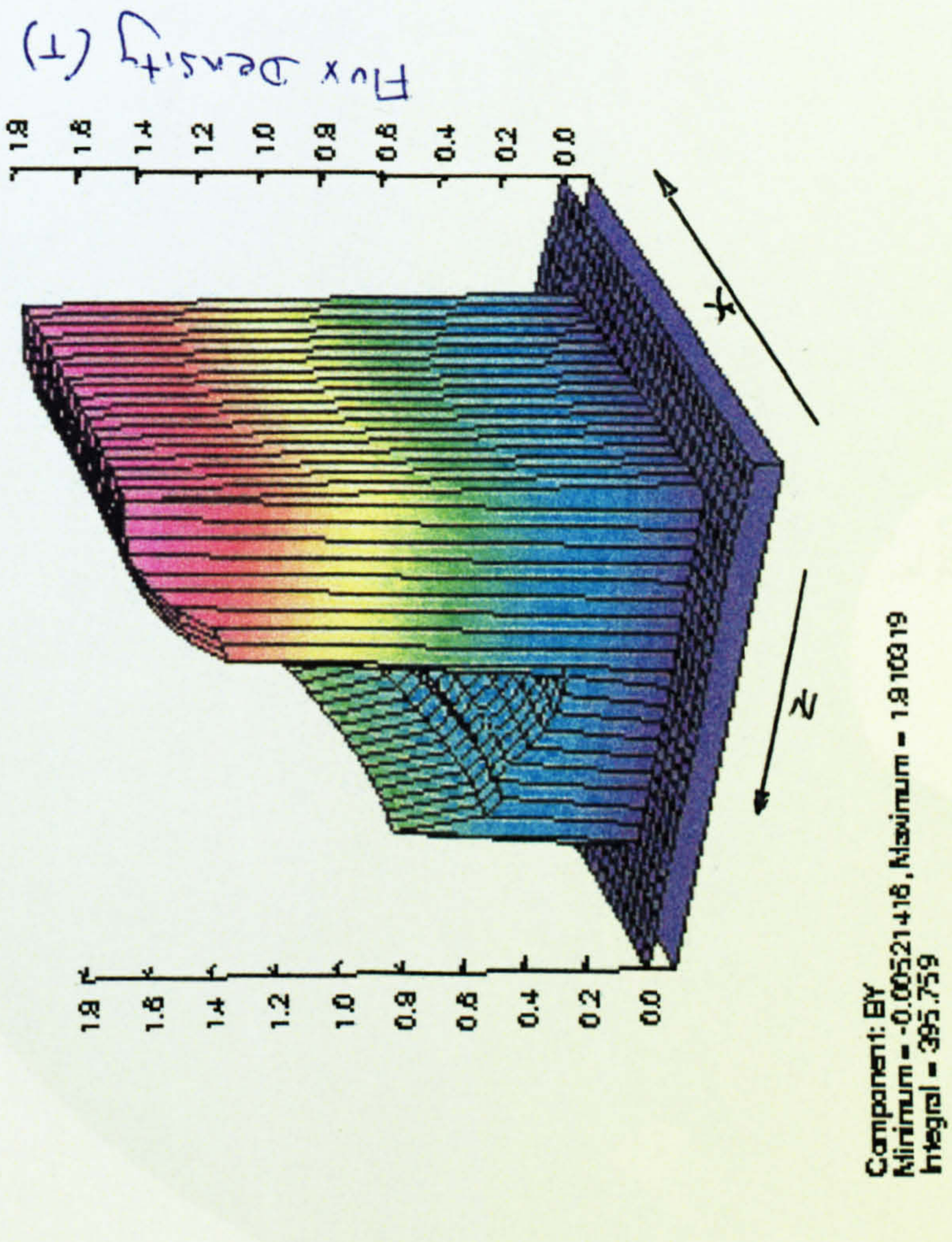


FIGURE 3.27 - Flux distribution in an Excited Pole

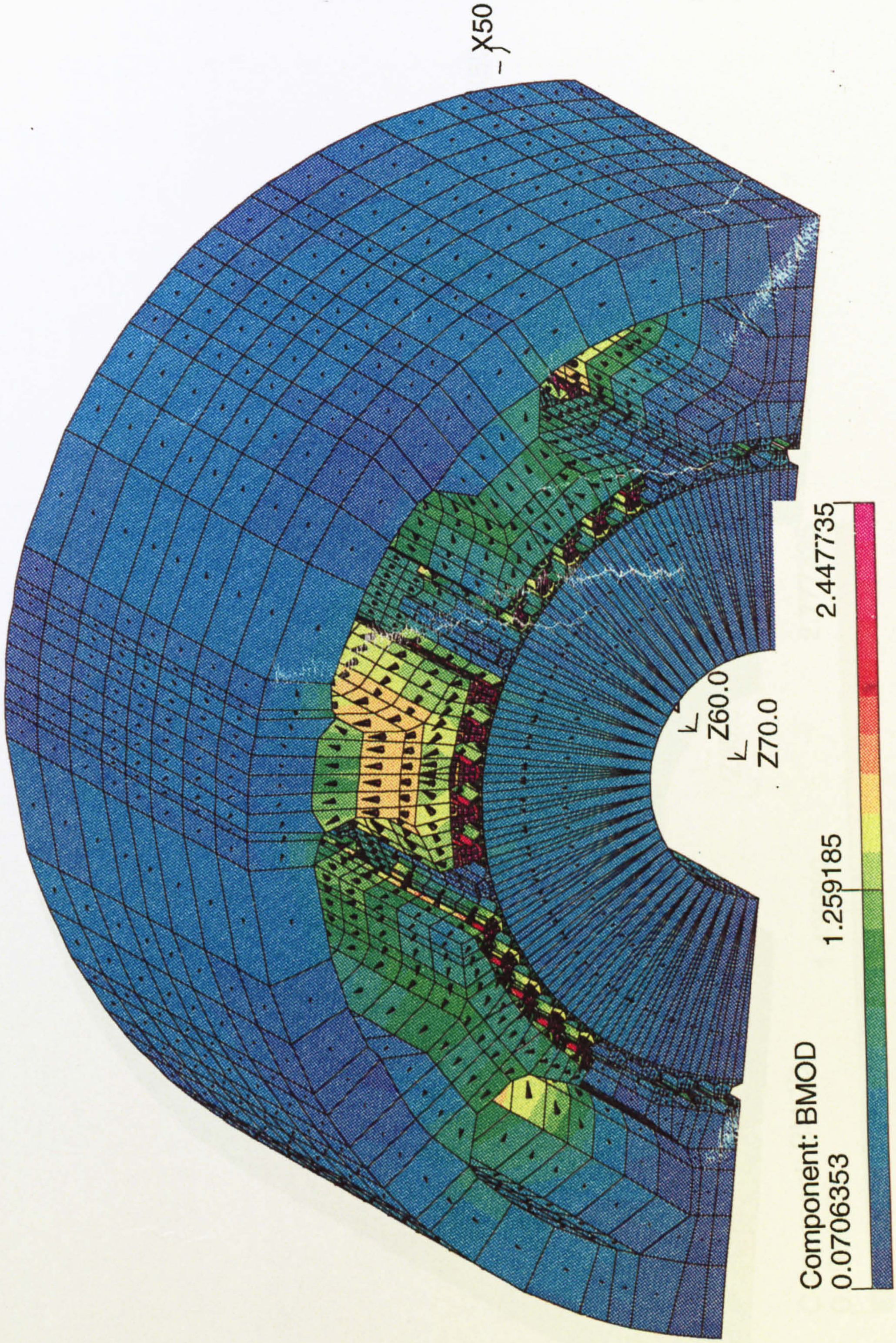


FIGURE 3.28 - Complete Flux Distribution

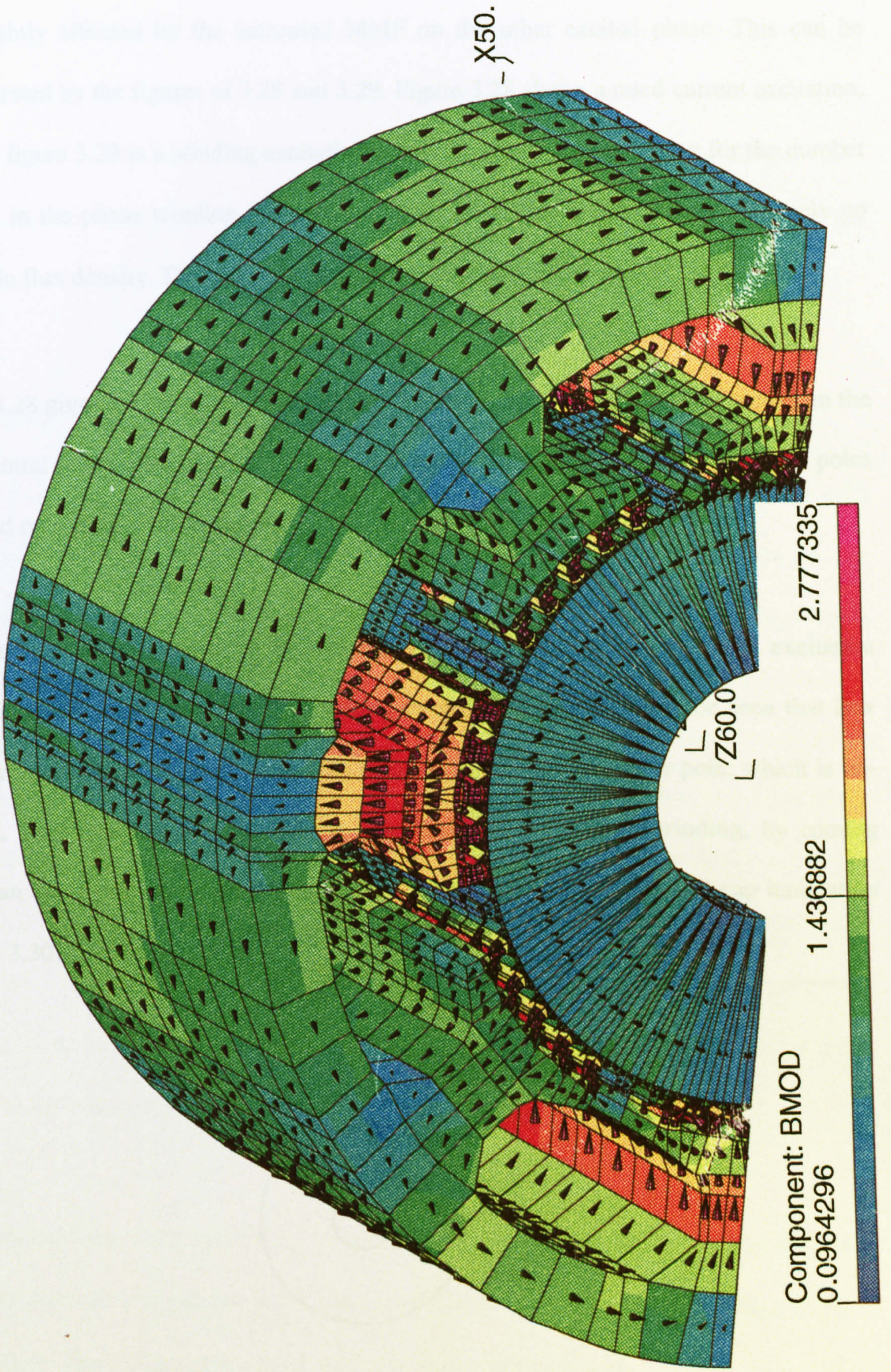


FIGURE 3.29 - Flux distribution with High Excitation

The stator pole that does not have any excitation from the windings directly placed on it is only slightly affected by the increased MMF on the other excited phase. This can be demonstrated by the figures of 3.28 and 3.29. Figure 3.28 shows a rated current excitation, whereas figure 3.29 is a winding excitation above the typical current rating for the number of turns in the phase winding. The colour of the two middle poles shows relatively no change in flux density. This suggests low mutual coupling in these poles.

Figure 3.28 gives a view of the flux paths in the motor. A general path that starts from the front central pole on the aligned teeth, travels to the rear aligned teeth on the stator poles on found on the sides of the figure.

Breakdown of the flux flowing through the magnet can be seen in the high excitation model in figure 3.28. Flux avoids travelling through the magnet. It can be seen that flux flows from one front aligned stator pole to the front of the next stator pole, which is un-aligned. This flux has directly arrived from the opposite polarity winding, by coming across an un-aligned tooth air-gap and then proceeding over the rotor end-cap lamination (Figure 3.30).

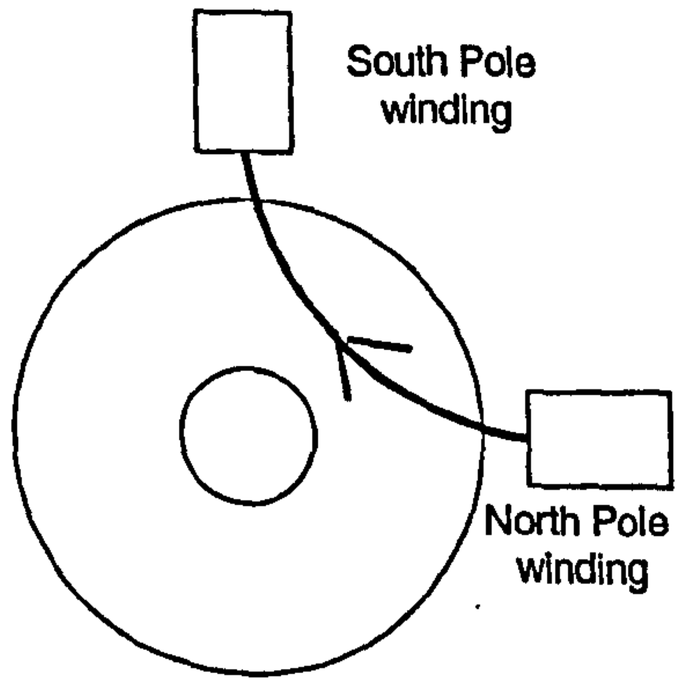


FIGURE 3.30 - Breakdown of Magnet Path

This effect is a contribution of the fact that the winding excitation pull becomes much stronger than that of the magnet. As the MMF of the winding increases the corresponding 'negative' MMF produced by the magnet increases also. At the same time the amount of flux flowing through the magnet will decrease. This corresponds with the B-H characteristics of the magnet, which shows that for increasing 'negative' field strength the flux density will fall.

3.8 Two Phase Analysis

Figure 3.31 shows the flux distribution for two phase excitation. Major flux loops are seen operating between opposite winding polarities. The front central aligned pole has an excitation polarity differing to that of its left hand side neighbour and flux will flow to this left hand pole. As this left hand pole has an equal steel permanence on the front and back the influencing factor is the operation of the magnet. This would act to direct the flux into the back part of the stator pole. However it can be clearly observed that the front of the stator pole has a significant amount of flux flowing directly into the front of the rotor. The MMF drop by flux travelling across the laminations in the Z-direction in the stator core is significant. This suggests that the MMF dropped by travelling across the laminations is comparable to the MMF produced by the magnet. The overall distribution can be again viewed in the sense of flux density distribution in figure 3.32.

The corresponding rotor teeth view is figure 3.33. The teeth that are close to the stator poles that have differing polarities of excitation to that of the magnet exhibit high levels of flux density. The bottom inner stator teeth are shown in figure 3.34.

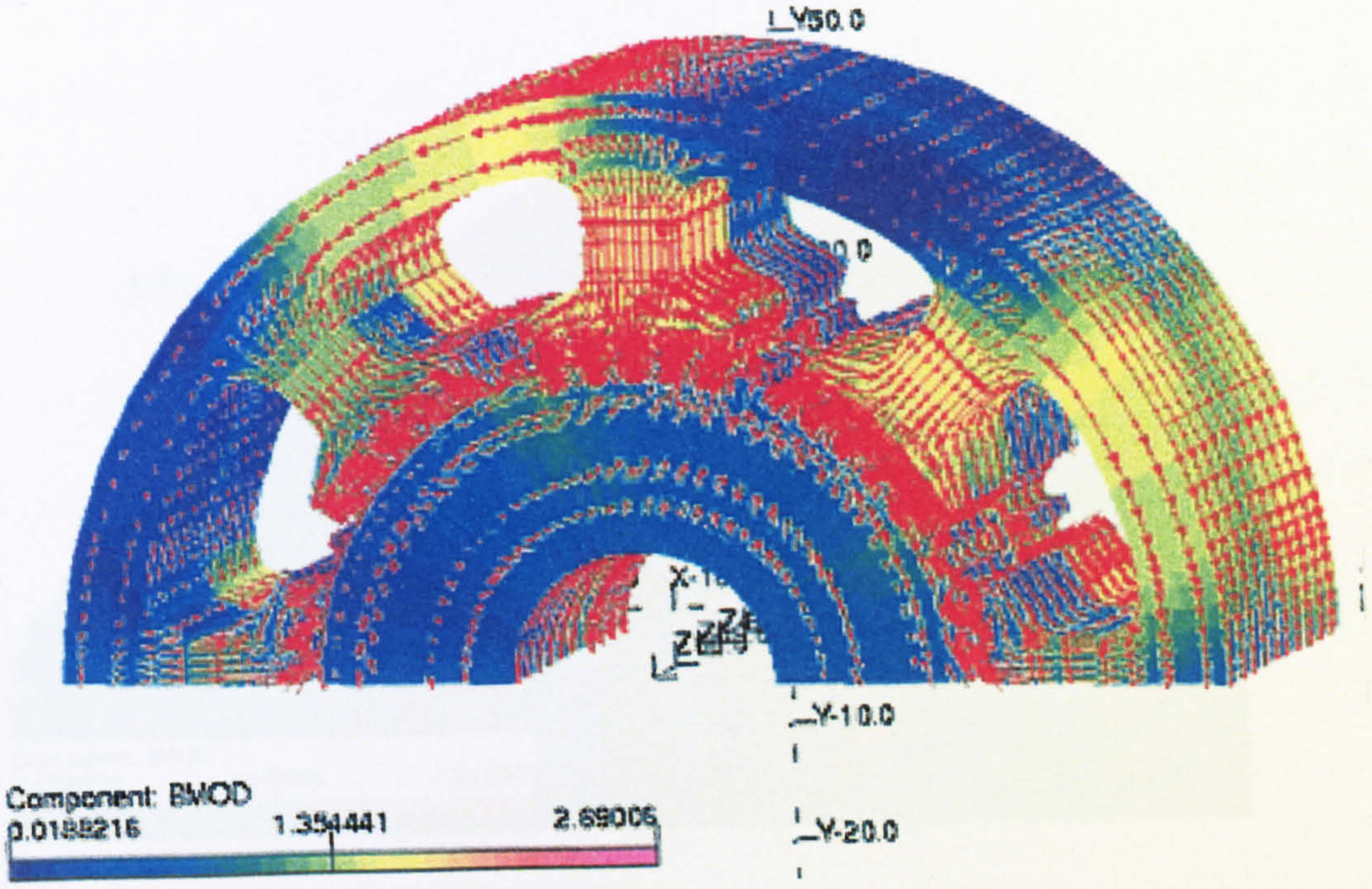


FIGURE 3.31 - Overall Flux distribution in Two Phase (180°)

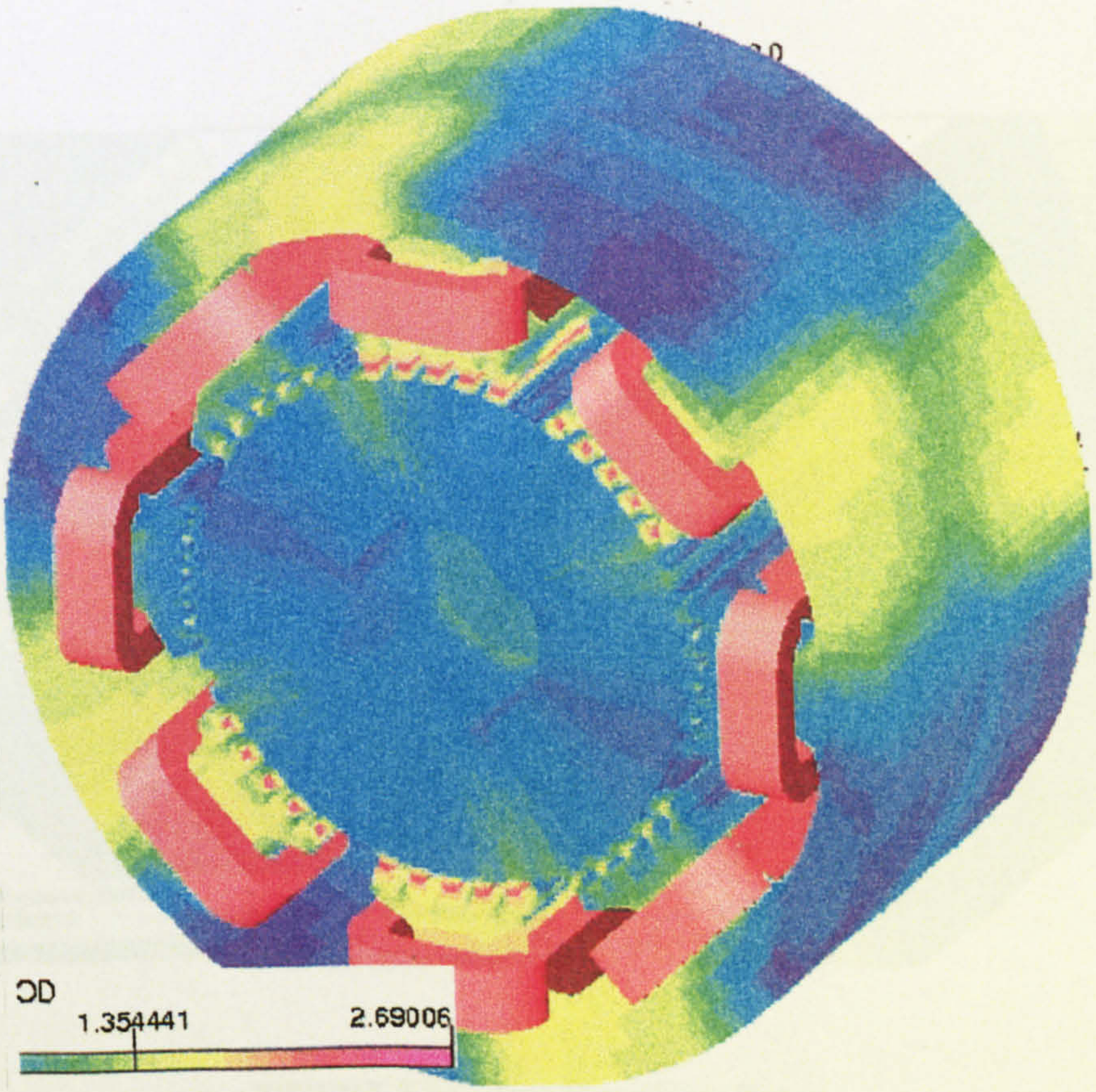


FIGURE 3.32 - Overall Flux distribution in Two Phase (360°)

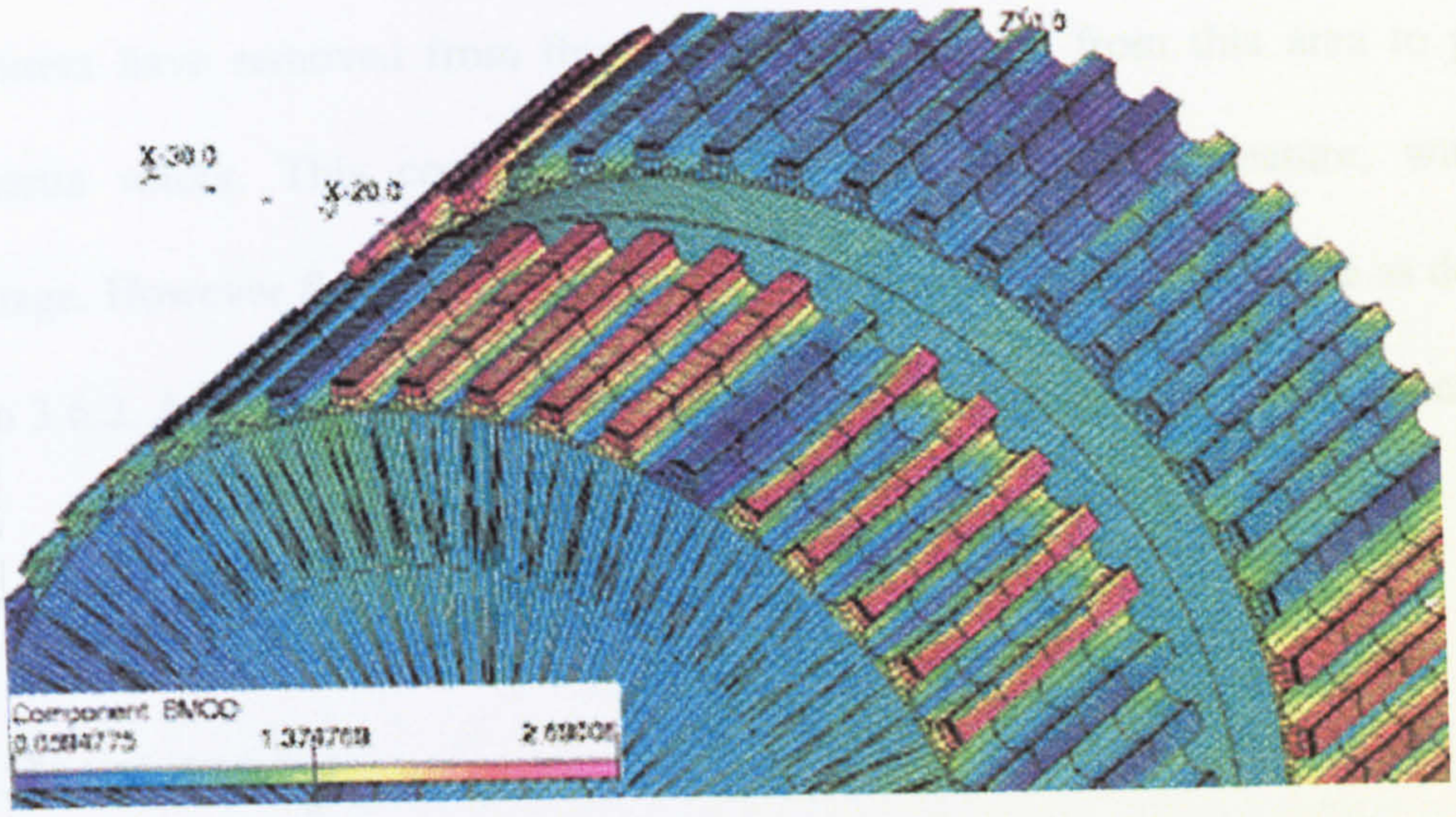


FIGURE 3.33 - Two Phase Rotor Teeth

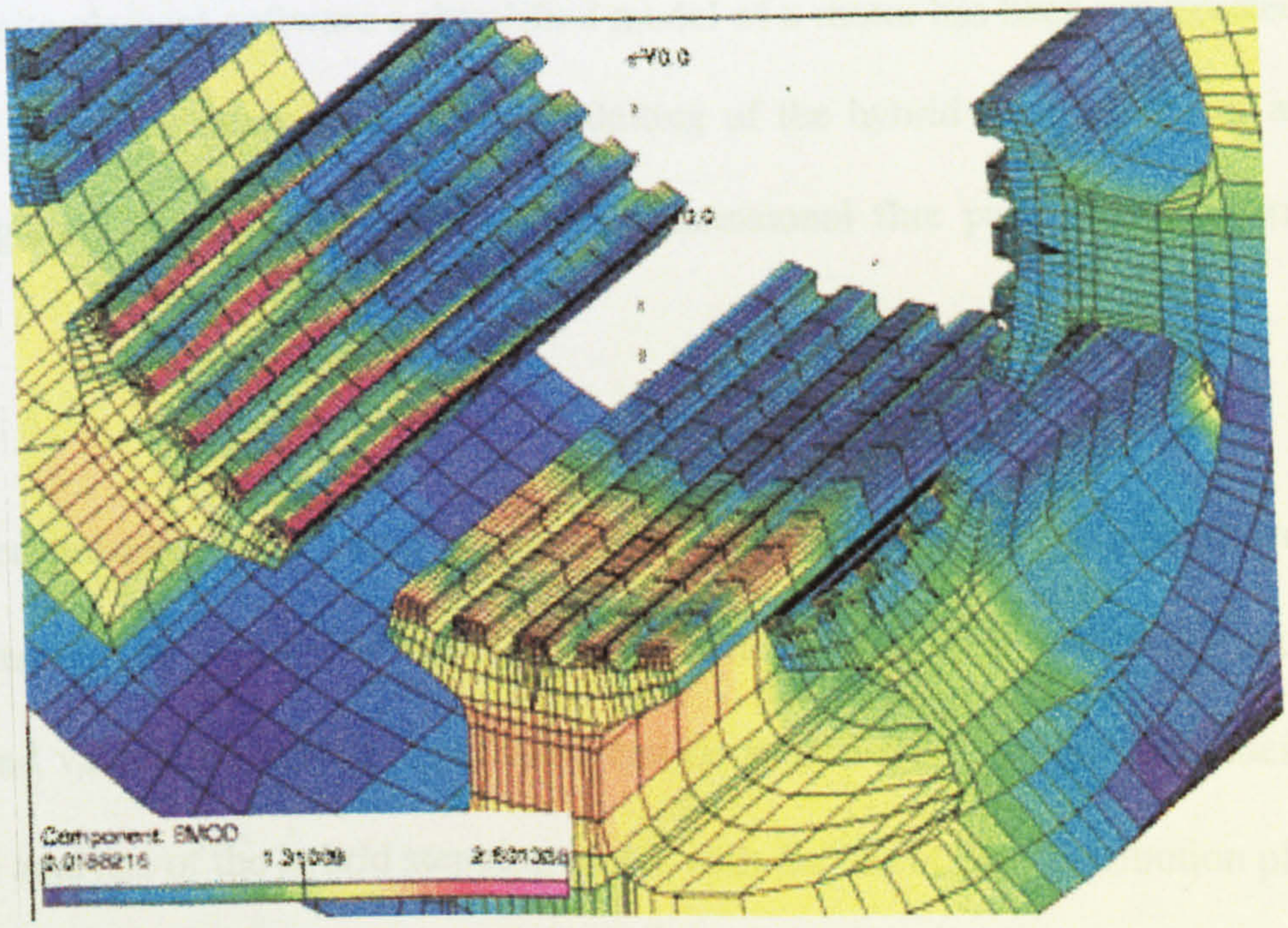


FIGURE 3.34 - Two Phase Inner Stator

3.9 Rotor End-Cap Core

In all the analyses, the central core of the rotor end-caps that are on the outside of the motor (z-axis) typically do not have a high value of flux density. Recently some manufacturers have removed from this a cylindrical section from this area to produce lower inertia rotors. This concept appears to be a worthwhile feature, with little disadvantage. However flux density does increase nearer the magnet interface as described in section 3.6.2. A better solution would be to remove a cone, with the pinnacle nearest the magnet.

3.10 Summary

This chapter forms a tutorial of techniques for modelling and analysing hybrid stepping motors using 3-D finite element software packages. A typical finite element package has been described. To understand the concepts for modelling the hybrid stepping motors with such 3-D finite element software a simplified model of a motor has been constructed. This model paid particular attention to basic modelling of the hybrid stepping motor and has allowed a greater understanding of the three dimensional flux paths in the hybrid type motor.

The chapter proceeds with using the concepts developed for the simplified motor to construct models for a practical hybrid stepper motor. It shows how different excitation schemes and values, for different positions can be solved. The chapter concludes with a qualitative analysis of the hybrid stepping motor with the use of flux distribution plots,

which has allowed much expanded comprehension of the fundamental elements and operation of the motor.

CHAPTER 4

DEVELOPING A STATIC MODEL OF THE HYBRID STEPPING MOTOR FROM 3-D FINITE ELEMENT ANALYSIS

Three dimensional analysis can provide a high accuracy solution for a hybrid stepping motor problem, at the expense of a long computing time. However it would be beneficial if a model was devised to achieve an accurate first solution or an indication of a new design's capability without resorting to the expense of three dimensional finite element analysis (3-D FEA) computing.

4.1 Three Dimensional Analysis Modelling

Three dimensional finite element analysis gives a complete and informative insight into the electromagnetic behaviour of a motor. The software can deal with the highly complex non-linear representation of magnetic steel properties, revealing non-linear saturation effects in the motor. It has the benefits of taking anisotropy effects of the laminations into account when calculating a solution. This is important as the operation of the hybrid

stepping motor depends on flux crossing the lamination in an axial direction perpendicular to the lamination plane in the back iron of the motor.

Three dimensional FEA allows easy manipulation of the different materials found in a motor. This allows differing grades of steel and packing factors to be selected for particular areas in the motor. Magnet polarity orientation can be defined in any three directions. End core and axial fringing effects are three dimensional effects, which increase in influence when the machine has a small axial length [49]. These can only be viewed in a 3-D analysis. Torque calculations may be performed by using the virtual work method, or the use of Maxwell stress analysis [48].

4.2 HyStep PC Modelling

HyStep is a computer modelling and simulation program developed during this project, for hybrid stepper motors written in the Borland™ C++ language [50]. It is used to predict the motor performance under static operating conditions and allows the motor's operation under dynamic situations to be solved. In order to generate a modelling technique for HyStep static analysis using 3-D FEA was performed to identify the key electromagnetic parameters. The hybrid stepper motor was split into sections (regions), that could then be modelled by different methods and subsequently integrated, producing a simple and accurate representation of the hybrid stepping motor. The basic structure of HyStep is shown in the flow chart of figure 4.1. The user can input motor material data ranging from B-H characteristics to tooth geometry using a selection of input menus. Options for

calculation methods are also required, as HyStep may use either look-up tables or curve fitting techniques.

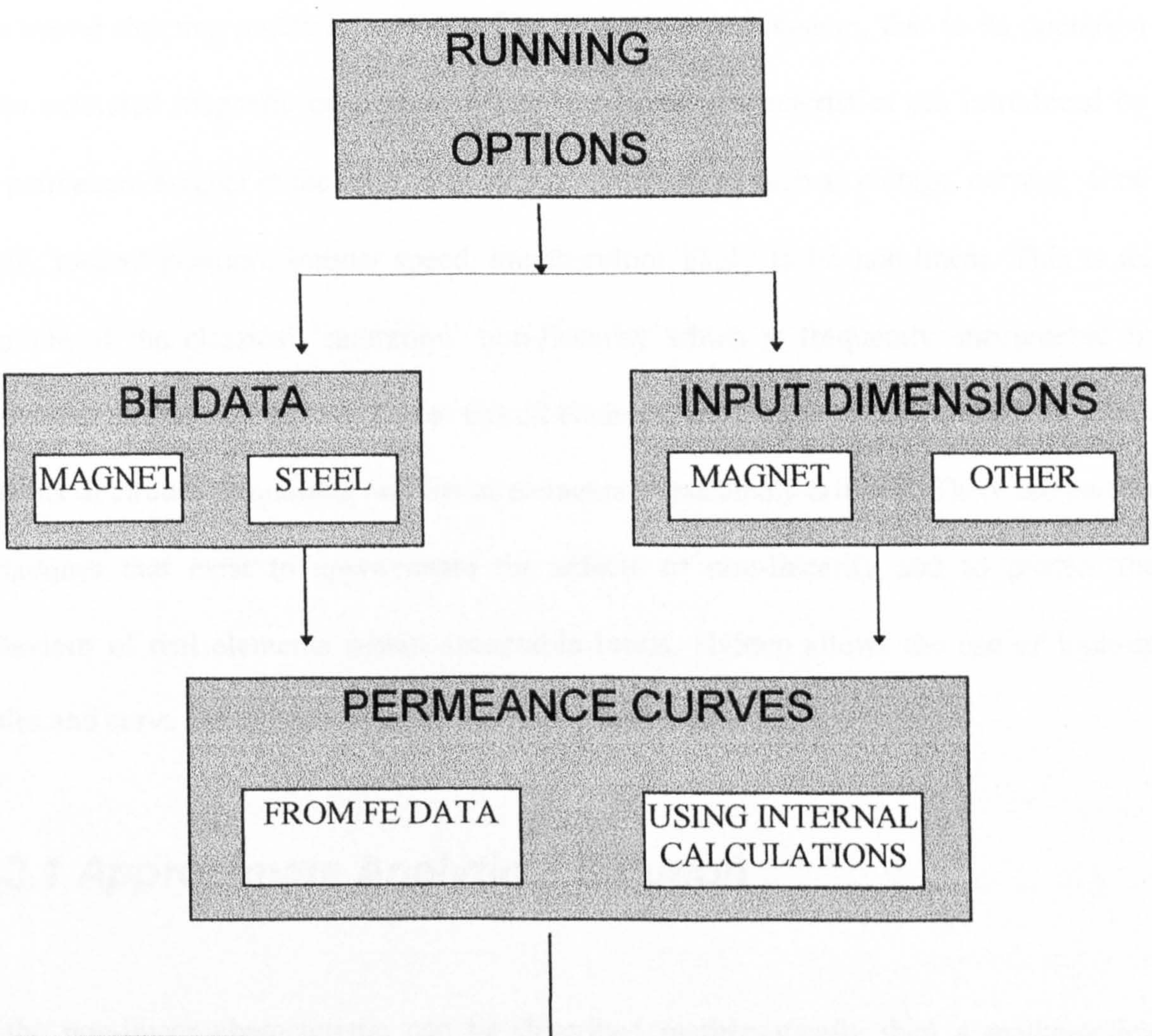


FIGURE 4.1 - Basic Overview of HyStep

4.3 B-H Characteristics

The hybrid stepping motor is considered to be a non-linear system, due to its operation under saturated magnetic conditions. Other non-linear characteristics are introduced by the permanent magnet in the rotor. The output relationships such as voltage/ current, flux/ MMF, torque/ position, torque/ speed, are therefore likely to be non-linear. This is an example of the classical 'saturation' non-linearity which is frequently encountered in many branches of engineering. Linear circuit elements are relatively easy to analyse. Exact analysis of circuits containing non-linear elements is extremely difficult. There are certain techniques that exist to approximate the effects of non-linearity and to predict the behaviour of real elements within acceptable limits. HyStep allows the use of look-up tables and curve fitting techniques to model the B-H characteristics.

4.3.1 Approximate Analytical Solution

If the non-linear characteristic can be described mathematically then a mathematical equation may be used to obtain a complete or approximate solution. For example the power series, may be used to describe non-linear characteristics;

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots\ldots \qquad (4.1)$$

If the first two terms are only considered, and powers of x higher than 1 are neglected, then this is a linear approximation of the non-linear characteristic. The higher the power, which is included, the more accurate the solution may become, but the analysis becomes

more difficult. There are approximations that are better suited to modelling B-H characteristics. These include Jiles-Atherton [51] and Hill type rational functions [52,53].

Modelling of the non-linear magnetic characteristics of steel using a single equation can offer many benefits for motor design and analysis packages. Design and analysis tools often use a B-H look-up table for circuits and devices, employing magnetic materials in their construction. Many curve fitting equations have been used and in particular the well known Jiles-Atherton model for the B-H magnetisation curve given in equation 4.2 may be used [51].

$$B = M\left(\coth(H/\alpha) - (\alpha/H)\right), \qquad (4.2)$$

where M and α are parameters.

This has been employed in some commercial simulation packages [54]. However, it has been found that the modern grades of Silicon steel used in electrical machines have a very sharp 'knee' in their B-H magnetisation curves and this is not always modelled accurately by the Jiles-Atherton expression. Furthermore, the Jiles-Atherton expression can be inaccurate at very low values of flux density.

4.3.2 Example of Modelling B-H curves [55,56[1] [2]]

New algebraic equations for fitting a B-H curve have been devised. These equations give excellent accuracy in the linear (low flux) region, follow the 'knee' and fit the saturation part of the curve with a very low degree of error. The new equations have been compared to the Jiles-Atherton equation and with the use of look-up tables and have fitted well [51].

There are many cases where the analysis for some items would be best represented by a non-linear model. These models are defined as those which have non-linear dependencies on their parameters. The Hill function can be suited to representing the B-H curve for magnetic steel. The Hill function is given by;

$$B = \frac{\alpha H^n}{\beta + H^n}, \qquad (4.3)$$

α, β, and n are real numbers.

Non-linear regression is based on determining the values of the parameters that minimise the sum of the squares of the residual errors. The solution must follow an iterative procedure for the non-linear case. Successful solutions are often highly dependent on good initial guesses for the parameters.

---

[1] Presented at 1996 IEEE Conference on Magnetics, Okyama Japan.

[2] To be Published in a future Journal of Facsimile Applications.

A B-H curve of 30 points would require in general an array of 30 times 2 elements for a curve of 30 discrete points. If this could be reduced to five or fewer parameters a dramatic reduction in memory[1] space would result. The new equations that have been found to accurately represent B-H curves of Silicon steel are of the form:

$$B = \frac{\alpha H^n + \kappa H^{n+1}}{\beta + H^n + \varepsilon H^{n+1}} \quad , \qquad (4.4)$$

$$B = \frac{\alpha H^n + \kappa H^{n+1}}{\beta + H^n} \quad , \qquad (4.5)$$

where α, β, ε, κ, and n are real numbers.

Equations 4.4 and 4.5, are examples of rational functions that were devised to model the B-H curves of magnetic steel[3] typically found in electrical machines and have been implemented into the HyStep programme. Figures 4.2 and 4.3, show typical B-H curves commonly used. Unlike traditional polynomial fits and interpolation they do not exhibit instability, and can intelligently interpolate past the last point of data.

[3]A 'magnetic steel' is defined by British Standards BS EN10106:1996.

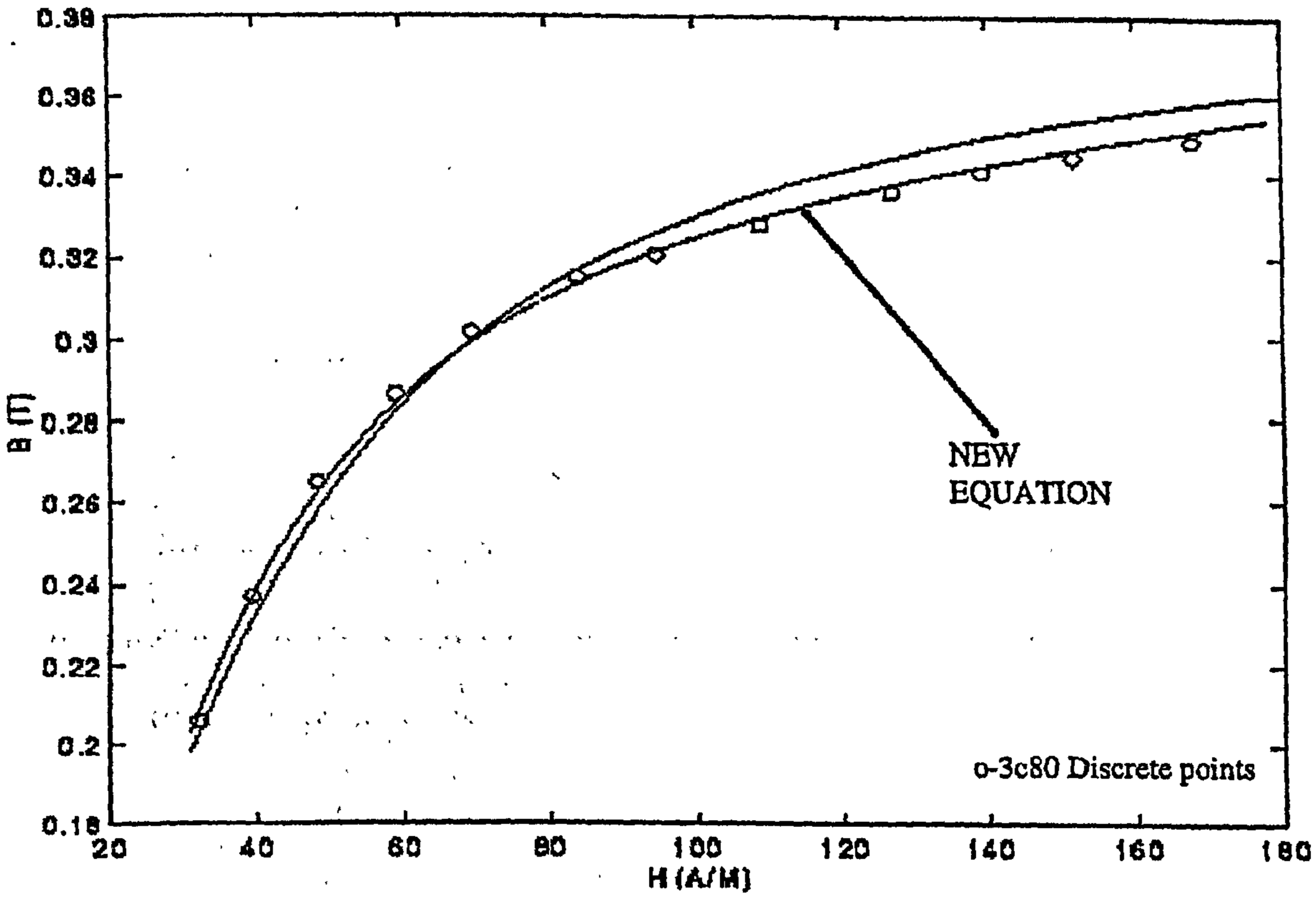


FIGURE 4.2 - Knee of fitted B-H curve to real data points

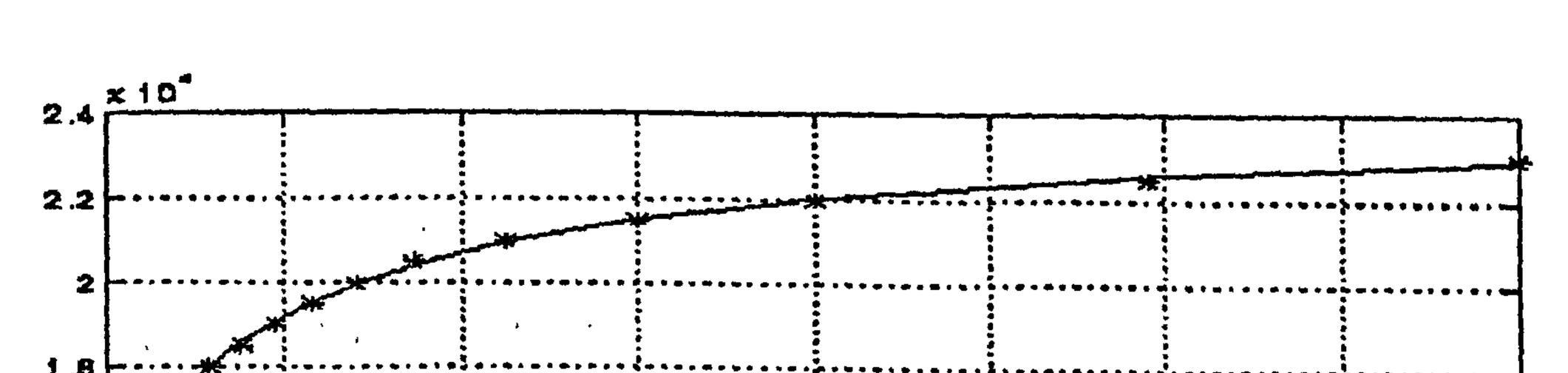


Figure 4.3 - Fitted B-H Curve to Real Data Points by New Equation

4.4 Flux-Linkage Characteristics

The modelling of the hybrid stepping motor requires the production of flux-linkage to current characteristics at different positions. Figure 4.4 illustrates 3 curves on a flux-linkage diagram. These curves are known as the aligned, mid, and un-aligned curves respectively. To produce the aligned curve a stator pole that has its teeth aligned with the front rotor end-cap's teeth need to be analysed. The rear rotor end-cap's teeth will be un-aligned with the same stator pole teeth. Without any current the flux linking the phase winding will be that produced by the magnet. This is represented as the non zero value of flux-linkage at no current on the aligned curve of figure 4.4

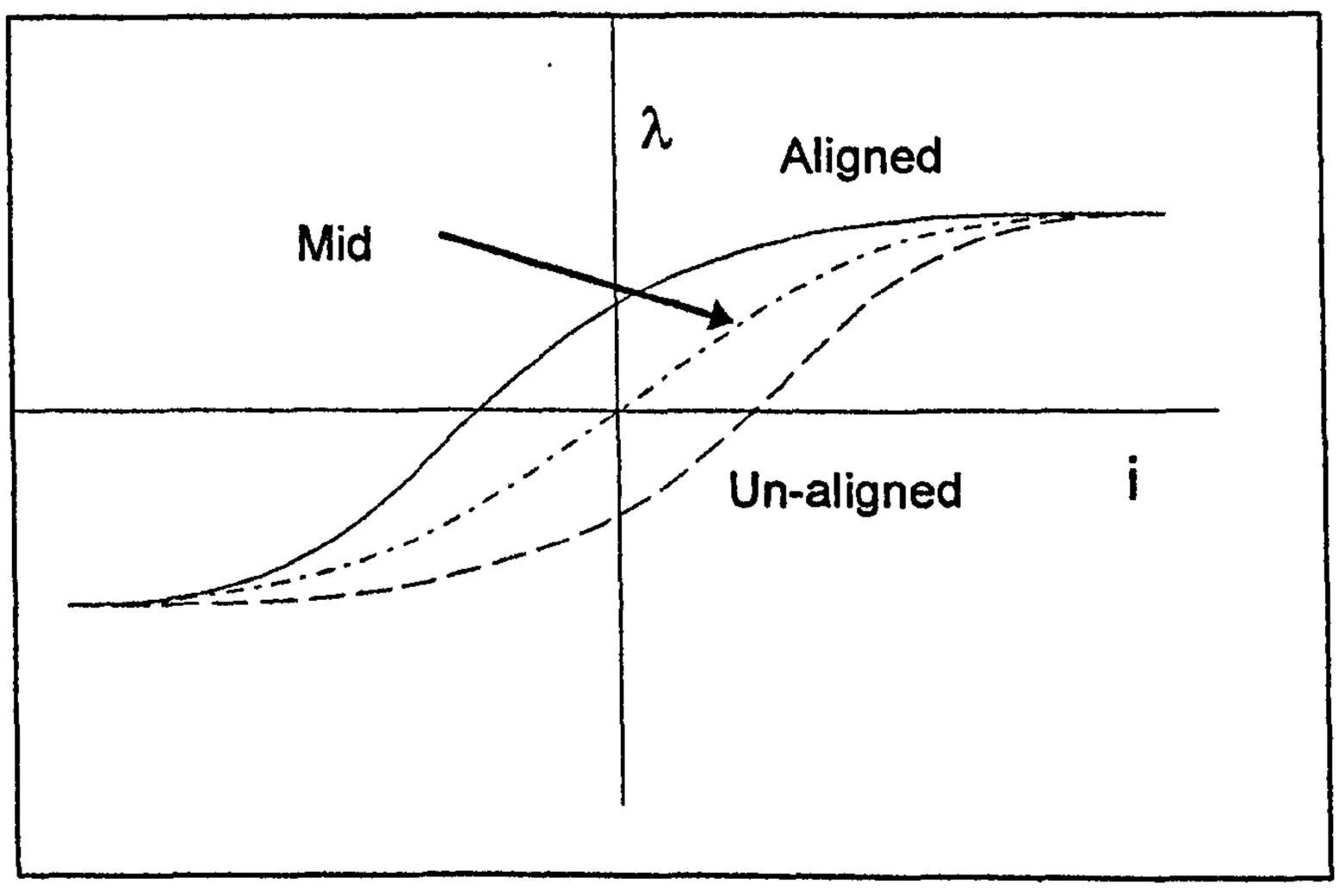


FIGURE 4.4 - Flux-Linkage Curves

To produce the mid curve of figure 4.4, the area of analysis would be the stator pole next to the one used in the aligned curve analysis. This stator pole's teeth will be half aligned with the front and rear rotor end-cap's teeth, but displaced by half a tooth pitch. Finally the un-aligned curve of figure 4.4, is found to be a stator pole with its teeth aligned with

the rear rotor end-cap's teeth, whereas the front rotor end-cap's teeth are now un-aligned. The un-aligned curve is practically the same as the aligned curve with the polarity of the magnet reversed. The same curve can be achieved from the aligned curve by reflecting its values in the Y-axis and then the X-axis of the graph, figure 4.5.

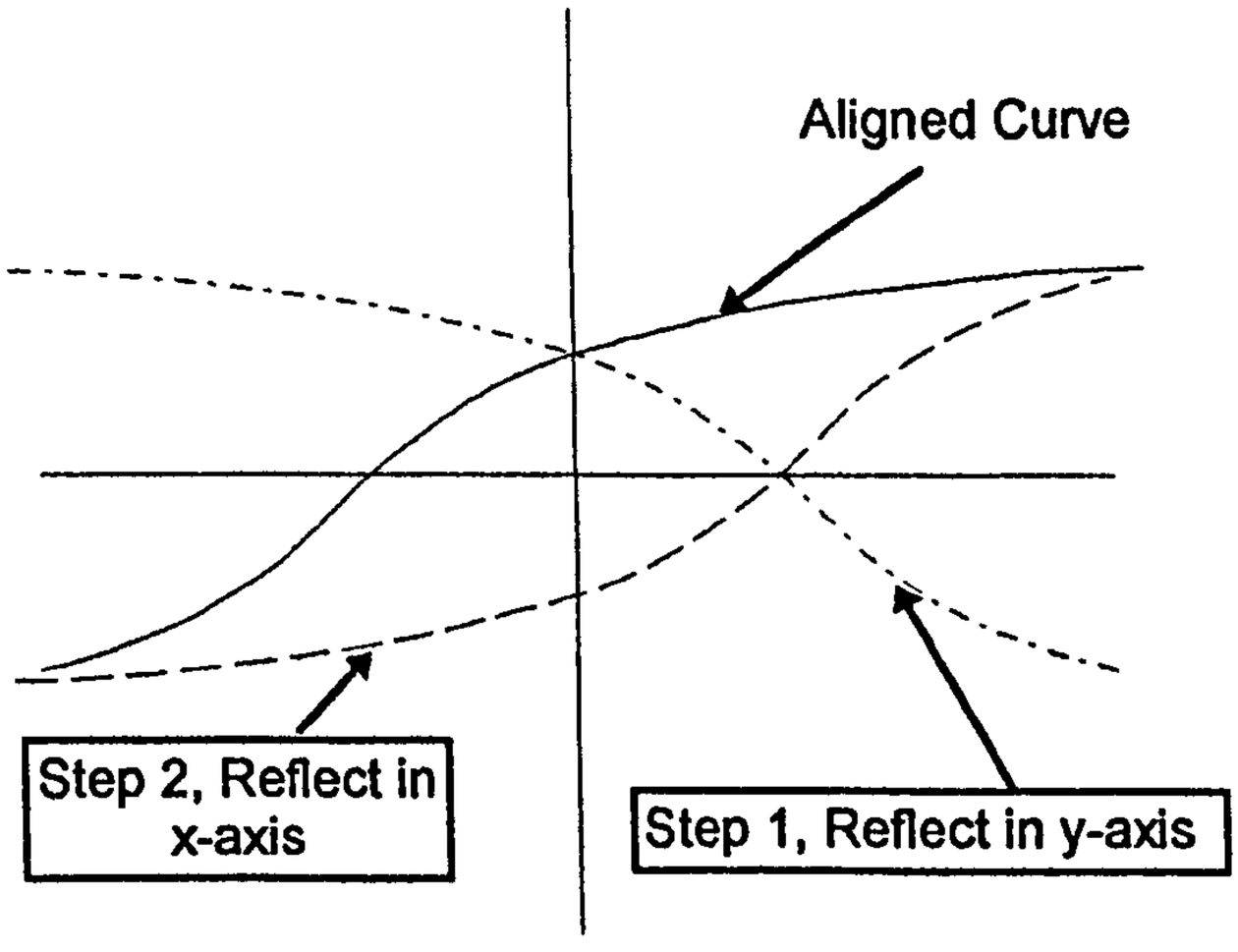


FIGURE 4.5 - Obtaining Un-aligned Flux-Linkage Curve

4.5 Determining Permeances of Air-Gap Regions using Two Dimensional FEA

A detailed description of the hybrid stepping motor operation was presented in chapter 1. In the previous chapters it was demonstrated that the motor operation relies on the generation of complex three dimensional (i.e. axial and radial) electromagnetic flux paths. Therefore, two dimensional (2-D) FEA cannot provide a complete solution for the hybrid stepping motor, as only the axial or radial plane can be analysed at any one time.

However some sections (or regions) in the hybrid stepping motor can be modelled to a good degree of accuracy by using two dimensional methods. In particular the tooth /air-gap regions have a majority of their flux travelling in the X-Y direction. The design of the tooth/ slot air-gap region strongly affects the static torque characteristics. It also affects the dynamic characteristics as it controls the inductance of each phase.

To make the curves of figure 4.4 from a two dimensional analysis, three situations need to be analysed. These are an aligned rotor tooth position, an un-aligned rotor tooth position and a half aligned (mid position) rotor tooth position. Two of these are shown in figure 4.6. From FEA solutions of these two dimensional tooth structures the complete flux-linkage to current characteristics of each position can be calculated. For 2-D FE analysis two excellent pieces of work were undertaken by Huard [45,46].

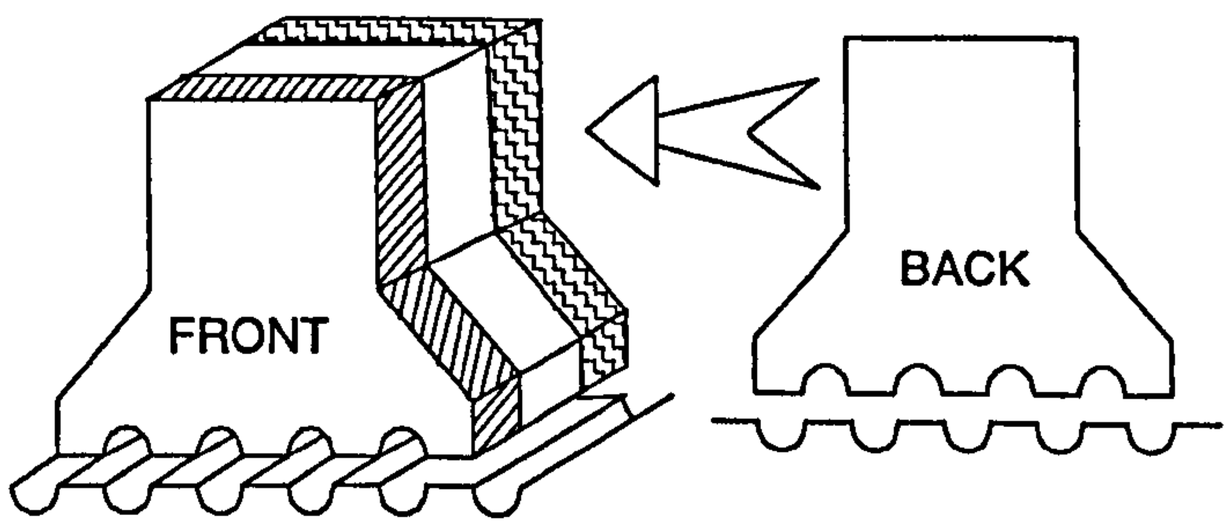


FIGURE 4.6 - Front Aligned Position with Back Un-aligned.

High accuracy can be achieved by using 2-D FEA on the tooth regions. It is beneficial to construct the pole/ teeth profile on a computer aided drawing package and import the outline as a DXF or similar file into the FEM pre-processor. This will allow the profile to

be seen as a series of construction lines and can then be easily adapted into a mesh for solving.

Figure 4.7 shows the rotor aligned with the stator pole. To solve the resulting mesh, and to obtain a useful electromagnetic distribution calculation as in figures 4.8A (half aligned), 4.8B (Aligned), and 4.8C (un-aligned), there are several steps that need to be undertaken.

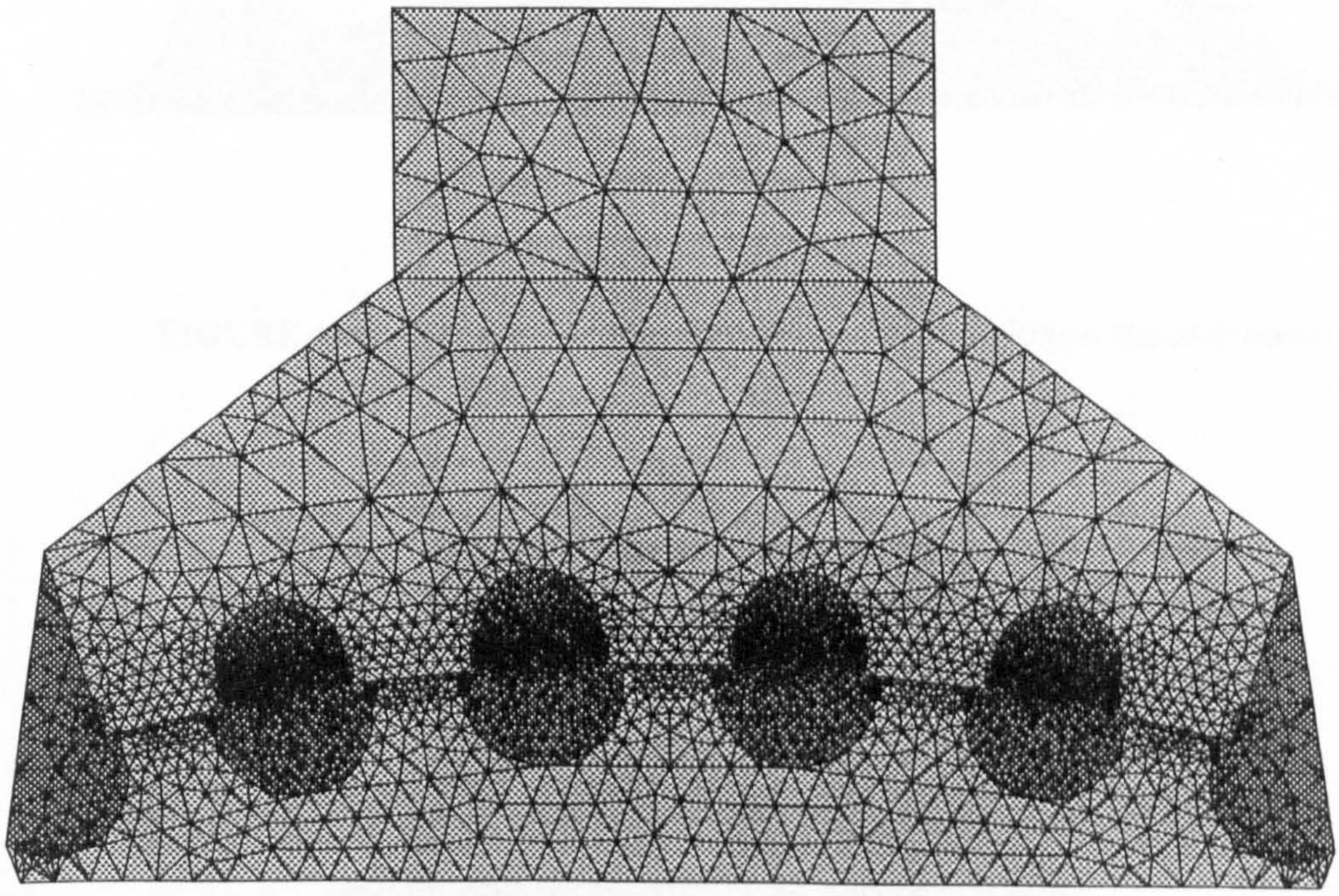


FIGURE 4.7 - Aligned Rotor 2-D Teeth Mesh

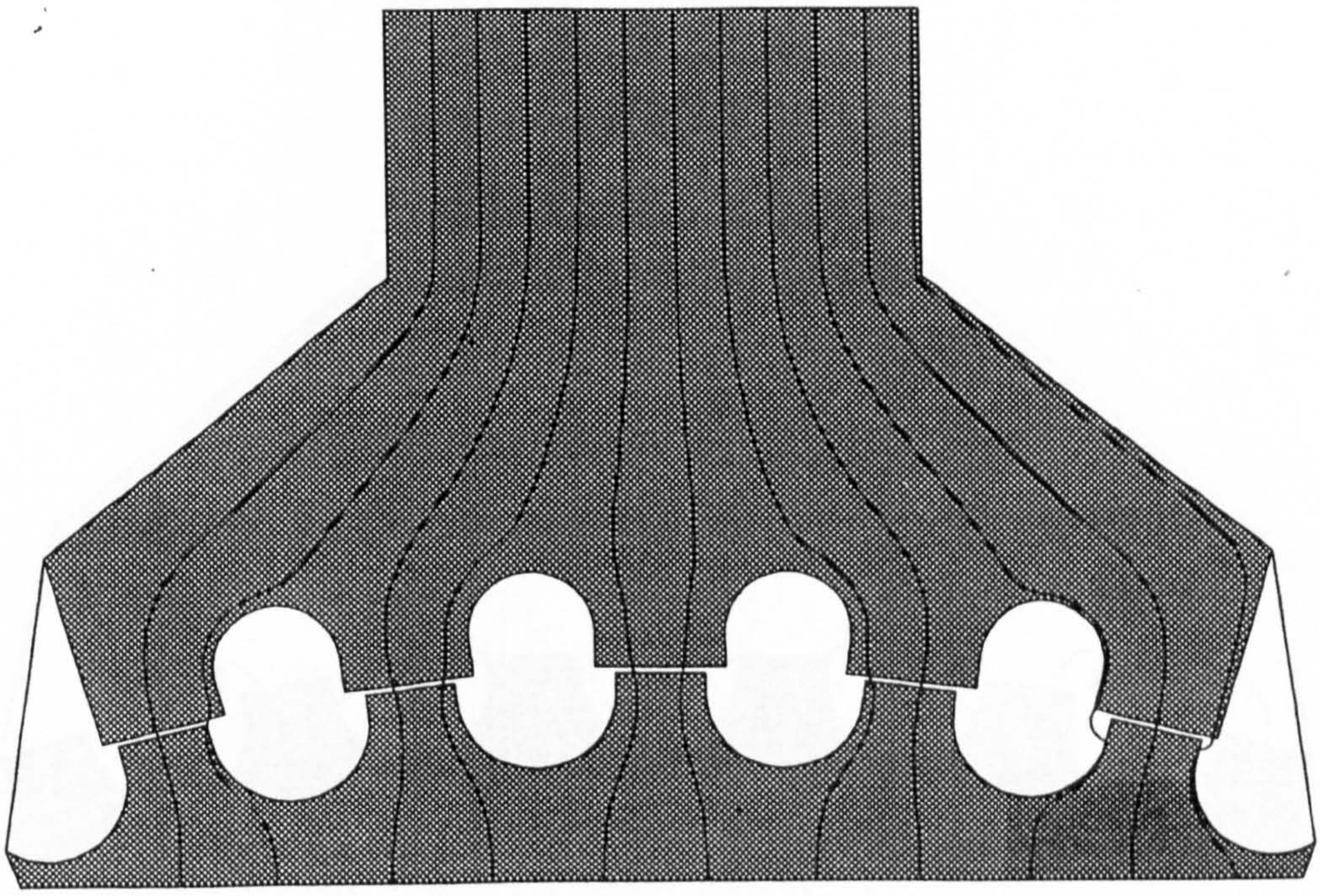


FIGURE 4.8A - Typical Flux Distribution of an Aligned Rotor/ Stator Position

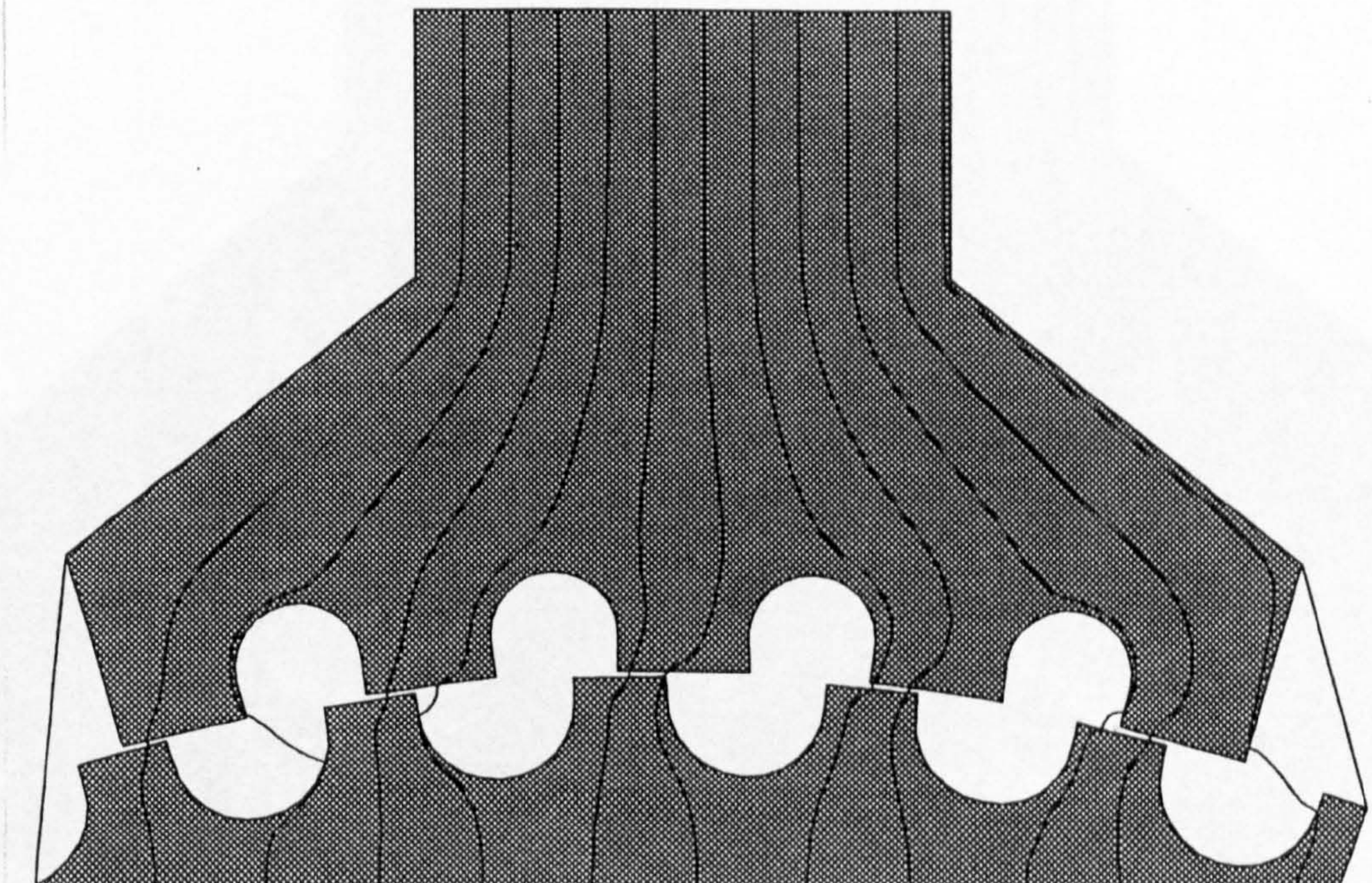


FIGURE 4.8B - Typical Flux Distribution of a Mid Rotor/ Stator Position

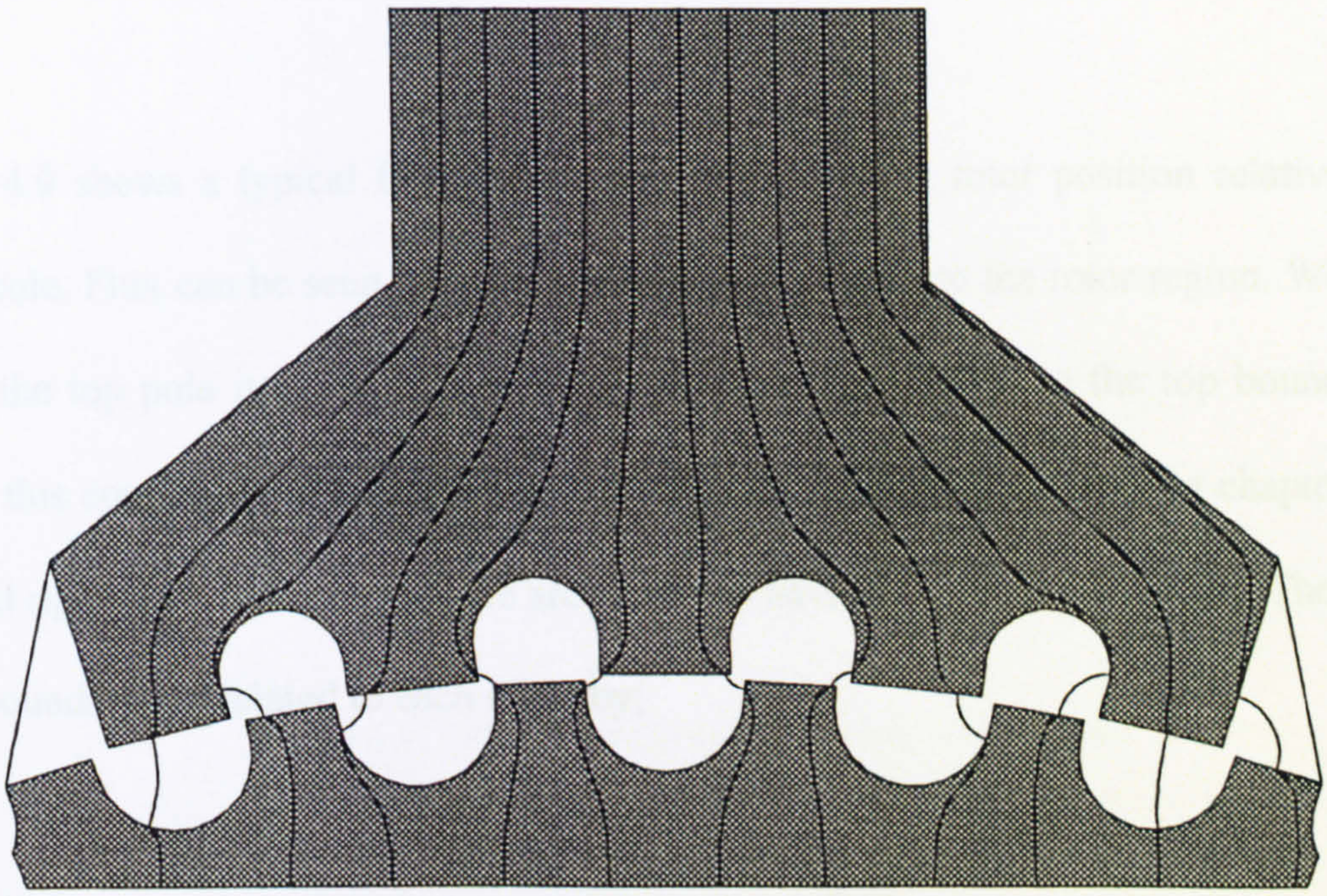


FIGURE 4.8C - Typical Flux Distribution of a un-aligned Rotor/ Stator Position

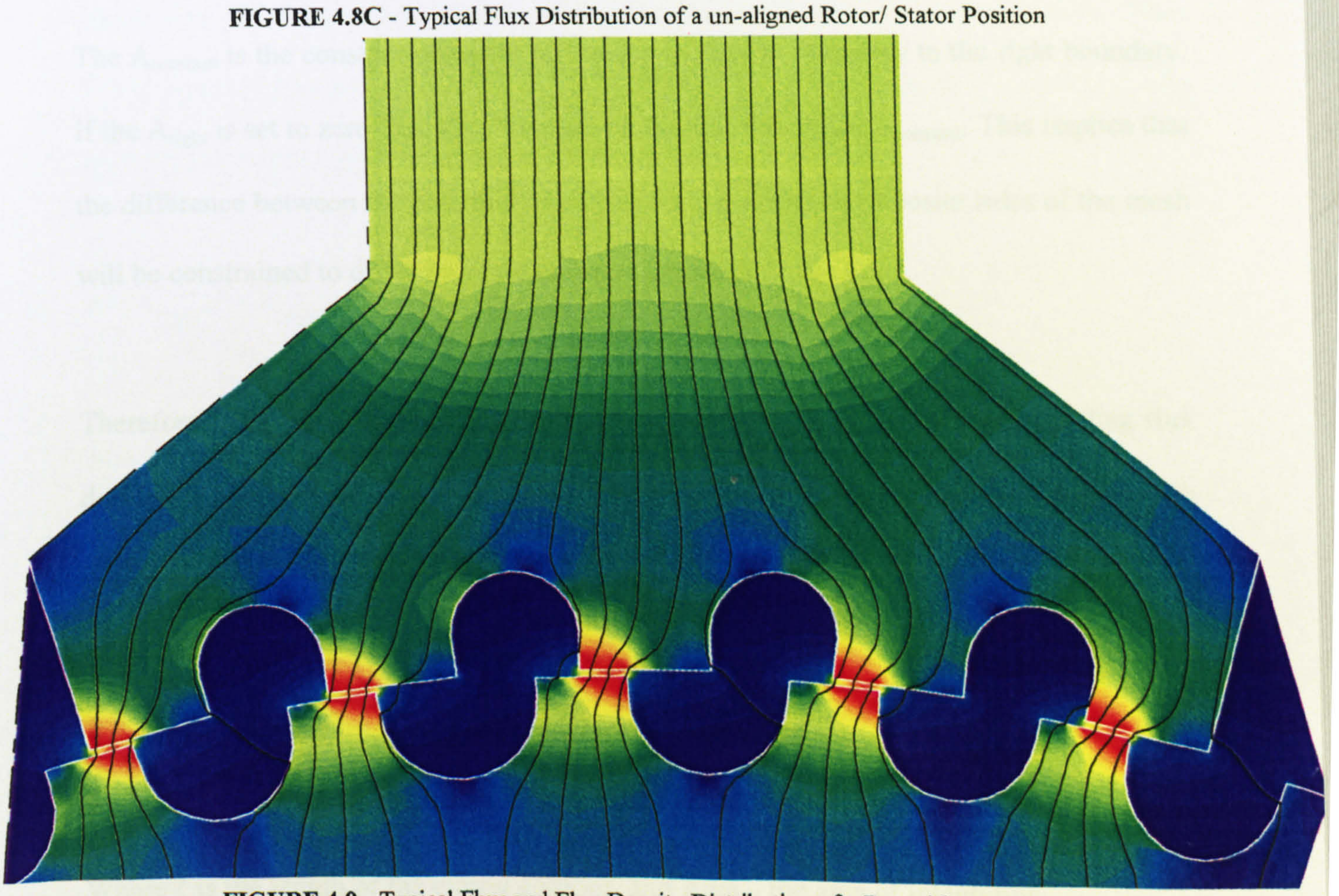


FIGURE 4.9 - Typical Flux and Flux Density Distribution of a Rotor/ Stator Position

4.5.1 Method of Calculating Reluctance [45]

Figure 4.9 shows a typical flux distribution in an aligned rotor position relative to the stator pole. Flux can be seen to travel from the top down into the rotor region. When flux enters the top pole it is assumed to be travelling perpendicular to the top boundary. To obtain this condition a Neumann type boundary is required as reported in chapter 3. The left and right sides of the stator pole are forced to having a differing potential. The left and right boundary are related to each other by;

$$A_{left} = A_{right} + A_{constant}. \qquad (4.6)$$

The $A_{constant}$ is the constant potential difference of the left boundary to the right boundary. If the A_{right} is set to zero then the Potential on the left will equal $A_{constant}$. This implies that the difference between the potential at corresponding nodes on opposite sides of the mesh will be constrained to differ by some constant amount.

Therefore for a particular solution we need to calculate $A_{constant}$ for a corresponding flux density. To understand the calculation of the potential constant, equation 4.7 is used with reference to figure 4.10 (in vector form).

$$\oint_C Pot \cdot d\vec{\ell} = \int \vec{B} \bullet d\vec{A}rea_{int} = \phi, \qquad (4.7)$$

Where ℓ is the length of the closed contour, and $Area_{int}$ the area of integration.

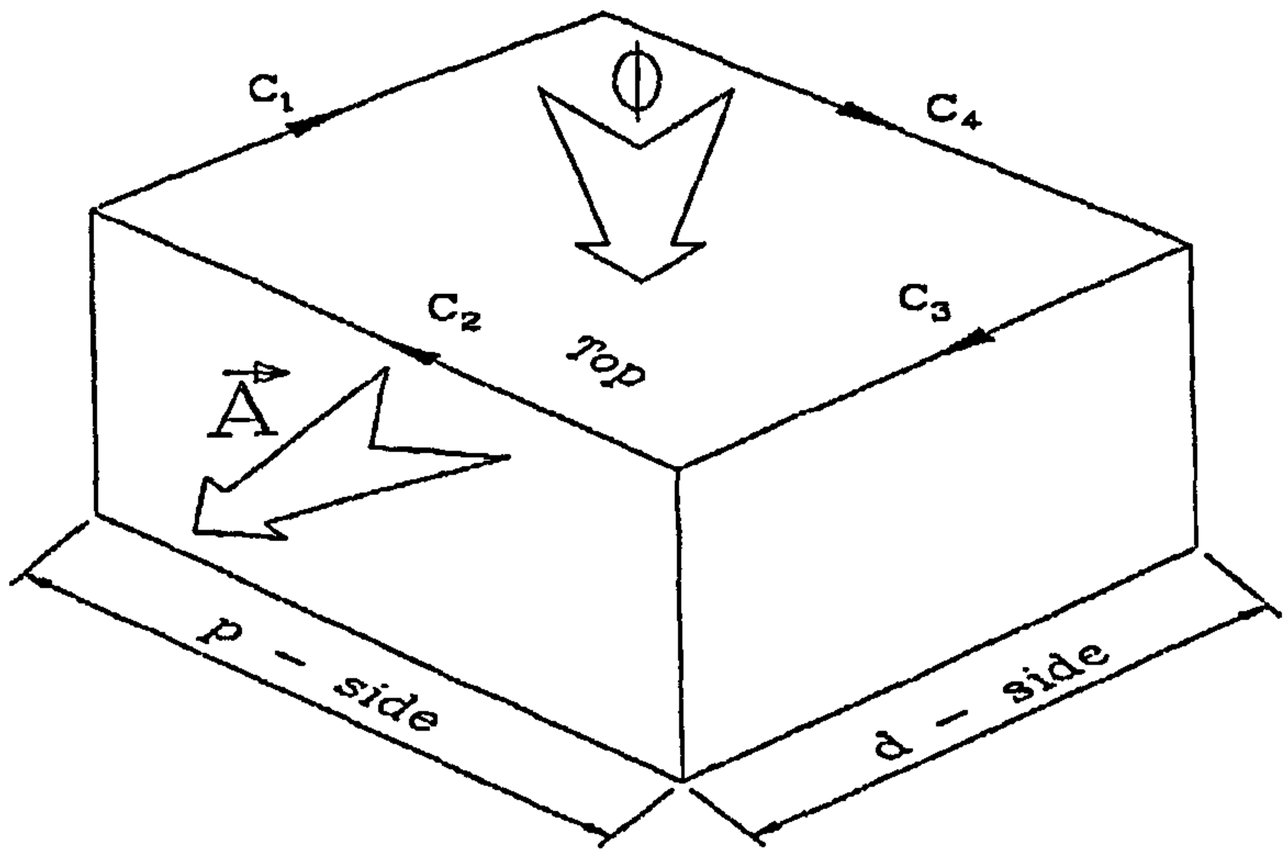


Figure 4.10 - Reluctance Calculation from Vector Potential

This states that the closed contour integral of the vector potential field is equal to the flux enclosed by the closed contour. The contour integration is indicated by path C. Since the problem is two dimensional the potential will be perpendicular to the front surface. Thus, it will be perpendicular to contours C_2 and C_4, and parallel to contours C_1 and C_3. The result of the contour integration is given by equation 4.10.

$$\phi = \mathrm{A}_{\mathrm{left}} d - \mathrm{A}_{\mathrm{right}} d = \mathrm{A}_{\mathrm{constant}} d\,, \qquad (4.10)$$

where d is the depth of mesh, and equal in length to C_1 and C_3.

In terms of Flux density B we can assign an average flux density at the top of the block to be;

$$B_{ave\ top} = \frac{\phi}{A} = \frac{\mathrm{A}_{\mathrm{constant}} d}{p \times d} = \frac{\mathrm{A}_{\mathrm{constant}}}{p}\,, \qquad (4.11)$$

where $B_{ave\ top}$ is the average flux density,

and p the front face length which is equal to C_2 and C_4.

This allows a mesh to obtain a flux field without any coil or magnet excitation in the mesh.

The mesh now can be solved. Quadratic or linear elements may be used in the solution. Quadratic elements will give a higher accuracy, due to their solution nature at the expense of increased computational time [48].

The general equation to calculate reluctance is shown in equation 4.12. This equation is not easily implemented when the problem is of a complex shape. Instead we make use of the linearised energy calculation that is readily available from FEM software packages[44].

The energy contained in a solution is equal to flux flowing through the mesh squared multiplied by the reluctance, i.e.

$$E_L = \frac{1}{2}\phi^2 \Re_L , \qquad (4.12)$$

where E_L and $\mathcal{R}_L$ are the linearised mesh energy per unit length, and mesh reluctance per unit length respectively.

The mesh reluctance can be calculated by rearranging equation 4.12 in the form of

$$\Re_L = \frac{2E_L}{\phi^2}. \qquad (4.13)$$

The linearised energy is calculated in the finite element program by;

$$E_L = \sum_e \frac{1}{2}\frac{B_e^2}{\mu_e} dA_e, \quad (4.14)$$

where B_e is the flux density in element e, μ_e is the operating point permeability of element e, and A_e is the area of element e.

The linearised energy concept is used to calculate reluctance will be per unit length, therefore the previous equation does not need to include the length of the element, i.e. ℓ_e. After obtaining the linearised energy, the reluctance per unit length for a particular flux level can be calculated by using equation 4.14. From the solutions, a table of MMF/flux relations can be generated, this should be scaled by the appropriate length of the stack under observation.

4.6 Isolated Teeth Analysis

In many papers on analyses of flux tube frames appearing on the teeth for rotor and stator poles, the individual teeth are assumed to operate as identical components. This implies that other teeth do not interact, thus assuming that the teeth can be summed together in parallel as a set of lumped equal teeth [45]. Finite element studies, recognise that the end

tooth is not equal to a mid tooth and therefore make allowances for its differing characteristics, such as end tooth side flux lines. Problems arise when the modeller assumes that the stator tooth components act as a complete stator pole. One does not know how flux will distribute once entering the top of the summed teeth components. Though some will argue that allowing analysis of individual teeth will give a higher degree of accuracy of individual components, personal studies have shown the accuracy is lost because of the unknown interaction between the teeth and stator poles.

Above the teeth is a region that is ignored in many analyses. This area allows the flux to flow into a particular tooth. It has critical importance in the analysis of the teeth/air-gap region as a whole as it accounts for correct flux distribution.

The stator pole is disregarded in some analysis where the modeller argues that unless the pole is thin the pole will not affect the results. Some papers will give two answers to an analysis, showing the discrepancies to be due to the stator pole saturation [33,57]. In the majority of cases the stator pole is shown to have a relatively high flux density concentration in relation to the rest of the motor, especially in the aligned teeth position. It is difficult to say when the pole width will affect the performance of the motor. When the pole width is smaller than the combined width of the teeth, the motor's performance may be affected by losses associated with higher than necessary flux densities in the pole. These will affect the MMF drop in the motor which is magnified with decreasing pole width [57]. Coil area is also dependent on pole width and as coil area is directly related to coil MMF, it would seem sensible to include the stator pole in all examples.

In the work involved in this thesis it has been found that the overall analysis of this region can be achieved accurately by taking both the pole and teeth, plus a small area of the rotor into account. This method can be difficult and tedious to calculate. By using 2-D FEM, a study on a particular design of tooth profile has allowed a highly accurate and in-depth prediction of the flux paths in the aligned, half aligned and unaligned rotor position regions. These have been successfully modelled by elliptical methods [58,59]. This method has the advantage of being relatively simple to input dimensions and has a fast solution time. These methods are described in the following sections.

4.7 Determining Permeances Algebraically

It would be beneficial to obtain permeance relationship with the use of finite element analysis to achieve greater accuracy. This however may be time consuming. If unusual designs of teeth are to be modelled it would be essential to obtain these relationships from a finite element package. If finite element software is not used then HyStep may alternatively calculate the permeance data by predicting the paths of flux lines internally. Notable work in this area has been carried out by several authors [6,31,57].

4.7.1 Permeances Calculation by Flux Lines Methods[66,67[4] [5]]

As the flux density in the air-gap can be relatively large and the air permeability is low, a large MMF drop is found in the air-gap region itself. Methods have been proposed which calculate the reluctance of the gaps by the means of flux tubes. These tubes are sections whose permeances are calculated and summed for an overall permeance. The crudest method assumes that the air-gap is the only region worth analysing. However it can be shown that the steel teeth surrounding the air-gap operate at high flux densities and therefore become like air as their permeability decreases. It is worth noting that many air-gap analysis methods that use flux tubes do so by predicting the tubes as a series of circular and straight lines [33]. This is inaccurate for the hybrid stepper motor as the tubes tend to follow more elliptical paths, as described in the following section.

The saturated, non-linear steel lamination characteristics described in section 4.3 present difficulties in producing analytical methods for prediction of the flux-linkage at varying rotor positions. Corda and Stephenson produced work for a similar problem in the switched reluctance motor [60]. This was later improved by Materu and Krishan [34] with intermediate positions.

---

[4] Partly taken form a paper presented at IEEE conference on Magnetics Okyama japan 1996

[5] Partly taken from a transactions paper published by the IEEE Magnetics society

A new analytical method for estimating the minimum, maximum, and intermediate positions of both the hybrid stepping motor and other salient pole structures has been developed. The proposed method uses a combination of ellipses to predict the behaviour of the flux lines in all position. The elliptical curve closely matches the actual flux line path.

The reluctance $\Re$ of flux tube in the steel or air-gap can be calculated by a combination of its permeability and physical size,

$$\Re = \frac{\ell}{\mu \times A}, \qquad (4.15)$$

where ℓ is the length,

A the area of the flux tube,

and μ the permeability.

If the path length follows an ellipse, such as in figure 4.11, the arc of the ellipse relates to the length $\ell_{ellipse}$. The governing equation for an ellipse is,

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1. \qquad (4.15)$$

The length ℓ_{ellipse} of an elliptical path can be calculated by taking the average of either side of the flux tube under analysis, for each boundary of the flux tube the length is,

$$\ell_{\text{ellipse}} = \int_{x_1}^{x_2} [1+\left(\frac{dy}{dx}\right)^2]^{1/2} dx, \qquad (4.16)$$

Where x_1 and x_2 are the start and end positions respectively.

Equation 4.16, may be calculated using Simpson's rule that states that an integral $\int f(x)dx$ can be approximated as;

$$\frac{d}{3}(y_0 + y_{2n} + 4\sum_{r=1}^{2n} y_{2r-1} + 2\sum_{r=1}^{2n-2} y_{2r}), \qquad (4.17)$$

where $y_0 \ldots y_n$ are the co-ordinates which divide the area under *f(x)* into strips of equal width *d*. The reluctance of the flux tube can now be calculated using equation 4.15.

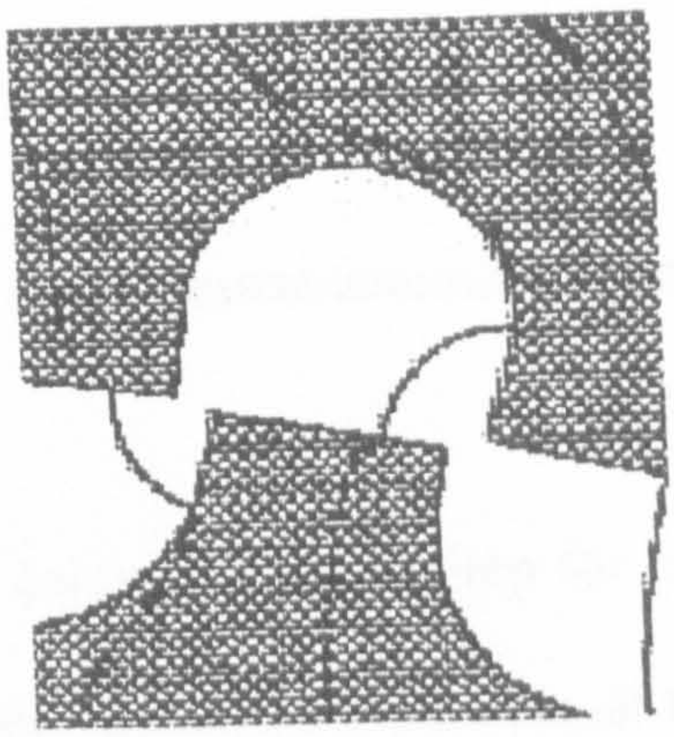


FIGURE 4.11 - Example of elliptical flux tube

4.7.2 Flux versus MMF Relationships

The flux versus MMF relationship are easily determined from 2-D FEA results as a reluctance is calculated at a set value of flux ϕ. The MMF drop in the 2-D profile can be calculated by

$$MMF = \phi\Re, \qquad (4.18).$$

However to calculate the flux versus MMF relationship using the reluctances of the sections calculated by methods described in the previous section. Each will need to be summed with its neighbours to produce a complete flux-linkage versus MMF relationship for each motor position. MMF may be calculated

$$MMF = Ni = \phi\Re, \qquad (4.19)$$

where N is the number of coil turns and ϕ the flux flowing in the path. The inductance can be determined by,

$$\lambda = N\phi = li, \qquad (4.20)$$

where λ is the flux-linkage and l is the instantaneous inductance

The flux versus MMF profiles calculated by HyStep for each position are dictated by the size of the air-gap and the magnetisation (B-H) curves of the steel used in the circuit. The magnetic circuits of each position are divided into sections of different lengths and areas. The sections are given an average area that is assumed constant over that particular area. A simplified example is given in figure 4.12.

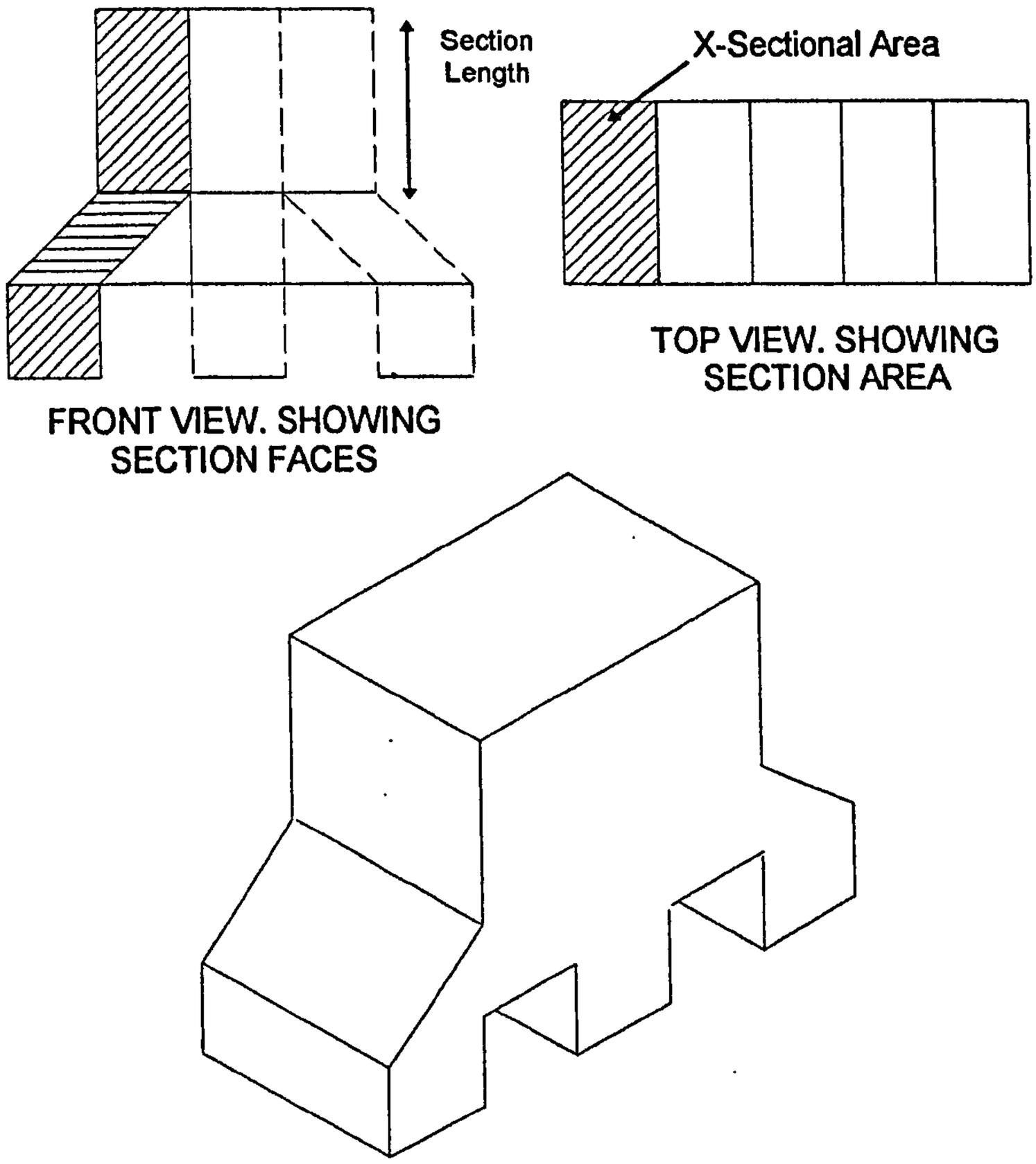


FIGURE 4.12 - Sections in a Simple Tooth Geometry

For a maximum flux density, B_{max}, in the steel, the maximum flux, ϕ_{max}, calculated in any section is limited by,

$$\phi_{max} = B_{\max} \times A_{\text{small sect}} \quad , \qquad (4.21)$$

where $A_{small\ sect}$ is the section with the smallest cross-sectional area.

From this equation it can easily be determined that the flux density of each section is found by,

$$B = \frac{\phi}{A} \quad . \qquad (4.22)$$

The MMF drop in each section is then determined using the B-H curve of the steel and the length of the section. For a given flux, the flux density for the section is given by equation 4.21. From the B-H curve of steel, the field strength H of the steel can be found. The MMF drop in these sections is then calculated by multiplying the section length ($\ell_{section}$) by field strength, equation 4.23.

$$MMF = H \times \ell. \quad . \qquad (4.23)$$

For air sections the relationship is linear between flux density and field strength, equation 4.24.

$$B_{air} = \mu_0 \times H_{air} \quad , \qquad (4.24)$$

where μ_0 is the permeability of air.

The sections which are in series have their MMF summed to create a total MMF for a given flux equation 4.25.

$$MMF_{series\ total}(\Phi) = \sum MMF_{series\ sections} \quad . \qquad (4.25)$$

Now there is a set of parallel ϕ versus MMF solutions. To find the total ϕ versus MMF relationship of the position, the fluxes of each parallel section need to be summed, with the series sections holding the same value of MMF.

$$\Phi_{\text{total}}(MMF) = \sum_{i=1}^{n} \phi_{\text{path i}}(MMF), \qquad (4.26)$$

where n = number of parallel paths, and $\phi_{\text{path i}}$ is the value of flux in the series path i. From this equation a flux versus MMF table can be created for each position under analysis. The complete process is summarised in the flow chart of figure 4.13.

4.8 Tabulated Data or Continuous Function Form

HyStep has two methods of storing and manipulating data. The first is the use of tabulated data or look-up tables, the second is the use of curve fitting techniques. In general the software will compute with the method chosen for B-H curve input, but it is possible to define which stages of HyStep use curve fitting and which use look-up tables. For example having the B-H tables as look-up table and the rest of the programs solutions controlled by curve fitting. This is advantageous in a situation where an unusual B-H curve relation is used or in a situation where the curve fit will not converge or the fit error estimate is too high. Curve fitting methods are extremely useful when the user requires results that are between or past known data points. The two methods are reviewed in table 4.1.

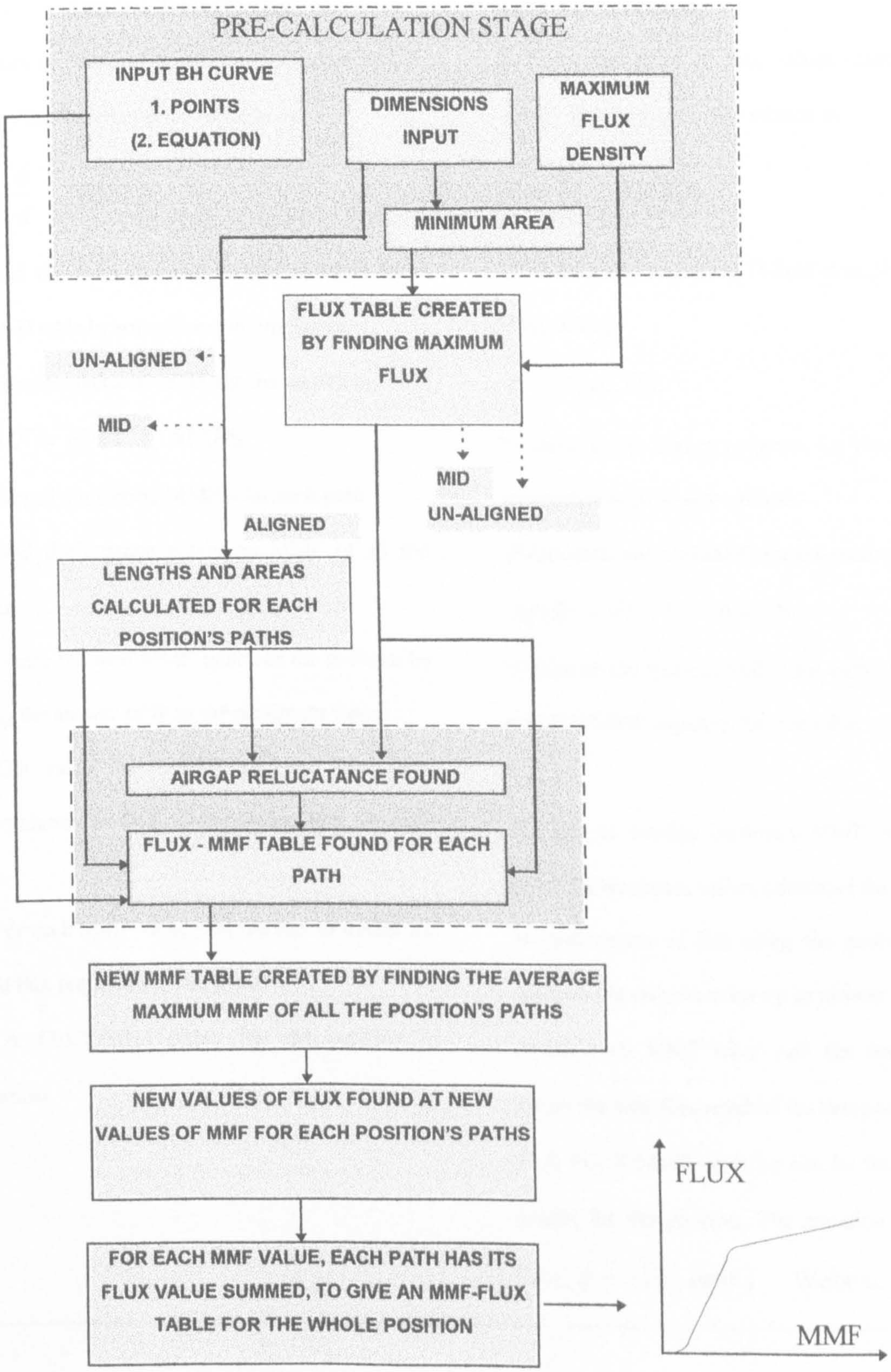


FIGURE 4.13 - Finding Flux /MMF table for each Position

Standard Tables Method.	Curve Fitting Method.
① From the table of flux values, calculate B-flux density for the appropriate section by: $B = \frac{\phi}{A}$ TESLA.	① From the table of flux values, calculate B-flux density for the appropriate section by: $B = \frac{\phi}{A}$ TESLA.
② Find the corresponding H-field strength from the B-H table by using linear interpolation.	② Find the corresponding H-field strength from the B-H equation; $B = f(a_i, H)$. by using a root finding program, i.e. Newton Raphson interpolation or Secant method.
③ Calculate value of MMF for the section by: $MMF = H \times \ell$ AMPS.	③ Calculate value of MMF for the section by: $MMF = H \times \ell$ AMPS.
④ Sum all the section MMF's for each path.	④ Sum all the section MMF's for each path and obtain a FLUX/MMF equation for that table.
⑤ Find the average maximum MMF of all the paths.	⑤ Find the average maximum MMF of all the paths using the maximum values calculated for each path.
⑥ Create the final MMF table for the position by using the answer of ⑤ as the maximum value.	⑥ Find values of flux using the paths FLUX/MMF equations in discrete value up to answer found in (5.
⑦ Get value from MMF table and use linear interpolation to find corresponding flux for each path.	⑦ For each MMF value sum the resulting flux to obtain the total flux required for that position.
⑧ For each MMF value sum the flux to obtain the total flux required for that position.	⑧ A FLUX/MMF equation can be fitted to the new points, for the position. The equation will be of the form; $\phi = f(a_i, mmf)$ Webers.
⑨ A FLUX/MMF table for the position is obtained.	

Table 4.1 - Finding MMF for Paths

4.9 Three Dimensional Effects Comparisons.

Can the tooth /air-gap regions be modelled to a high enough degree of accuracy using a 2-D analysis. Figure 4.14 is an analysis for an aligned curve taken inside the rotor end-cap. The left hand side is the rear end-cap (un-aligned teeth), and the right hand side is the front end-cap (the aligned teeth). The B_y (flux density in the Y direction) is clearly the largest throughout the front pole. The B_x and B_z components of flux density are relatively small in relation to the B_y.

In the air-gap region which is represented by figure 4.15, the results are similar to the rotor. The peak of B_y is lower as there is a spreading of flux over the complete rotor. The B_x and B_z components are still relatively small. The dip in B_y occurs over the magnet where the air-gap length increases.

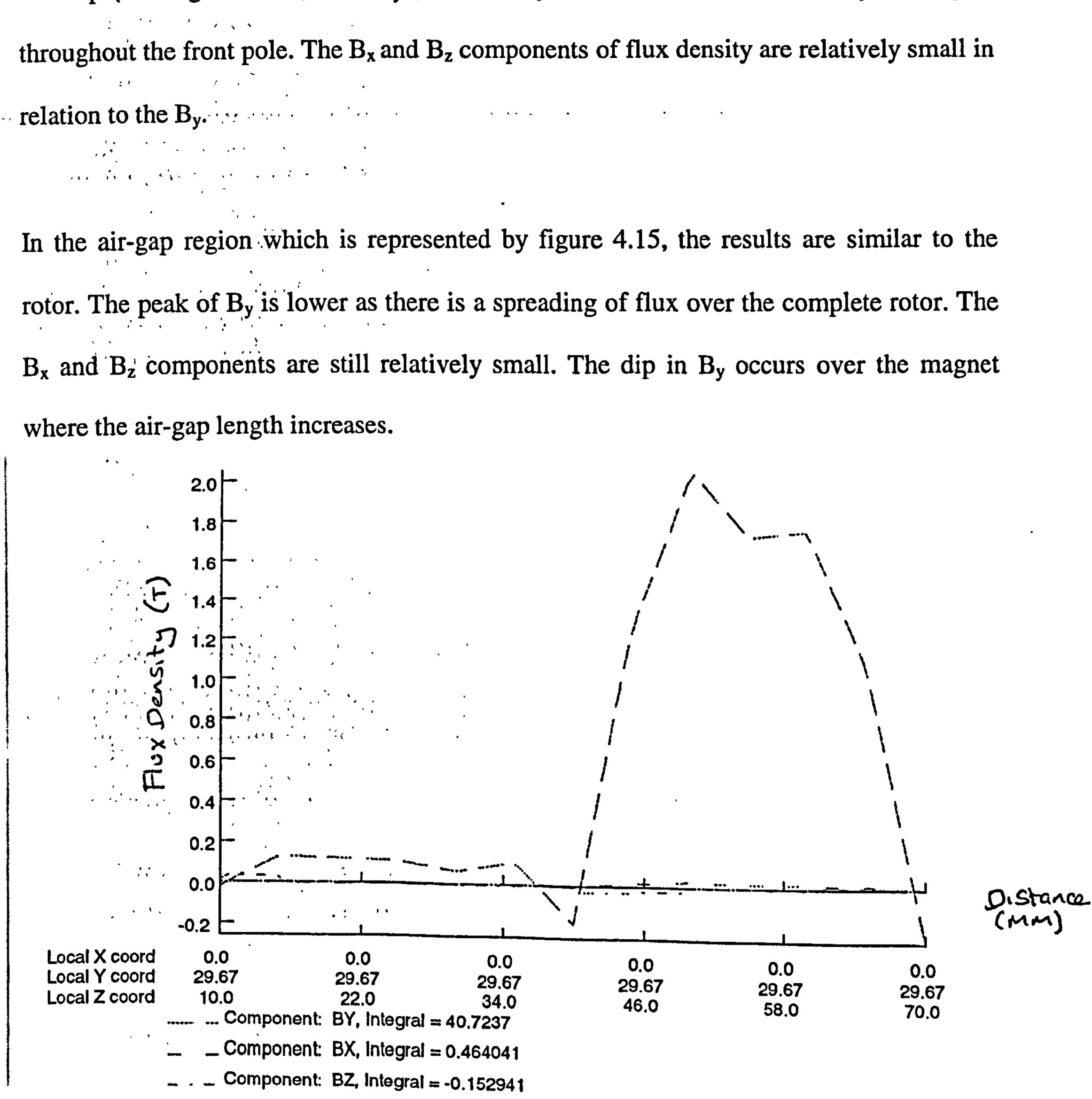


FIGURE 4.14 - Bx, By, and Bz Plot for Rotor

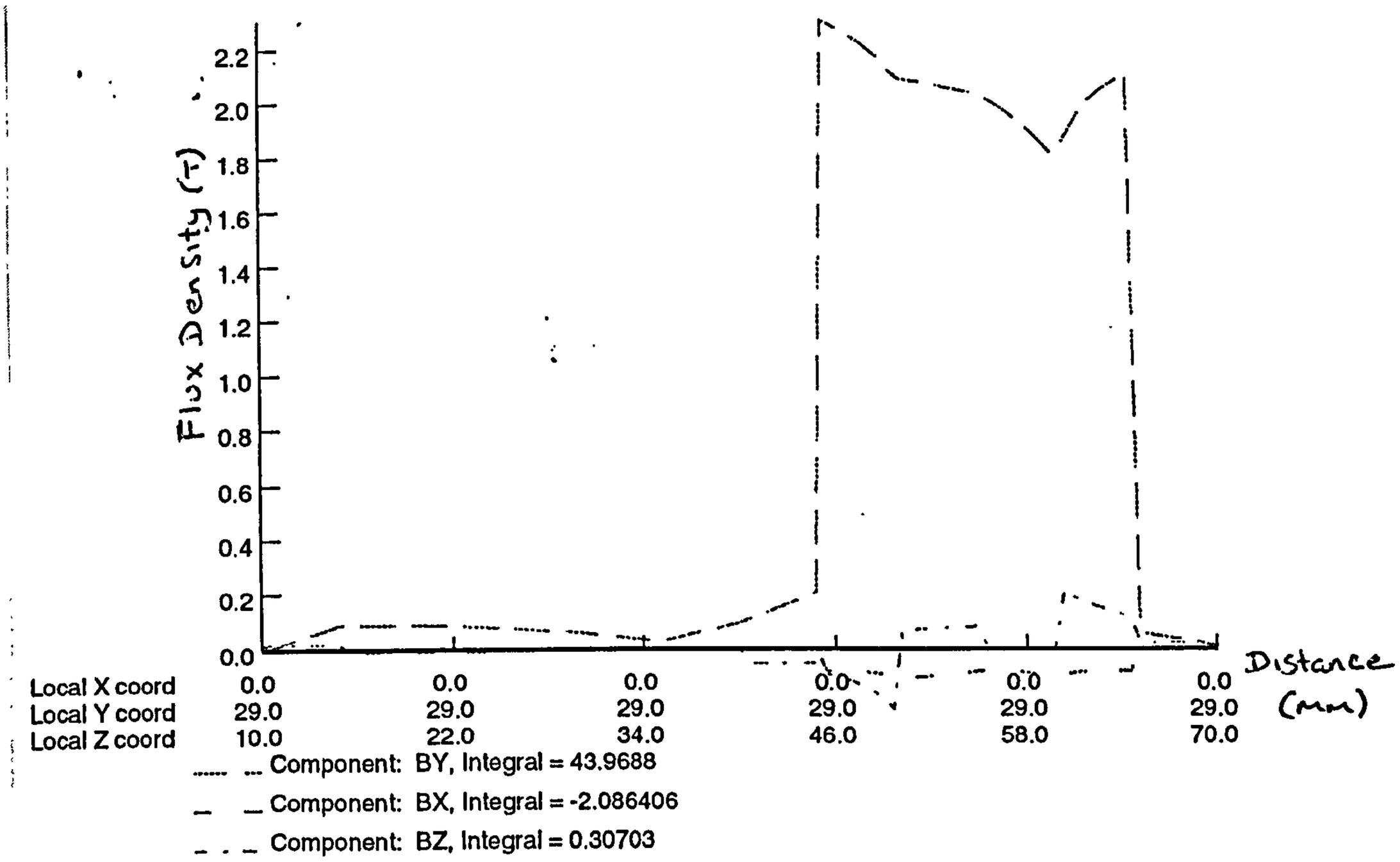


FIGURE 4.15 - Bx, By, and Bz Plot for Air-Gap

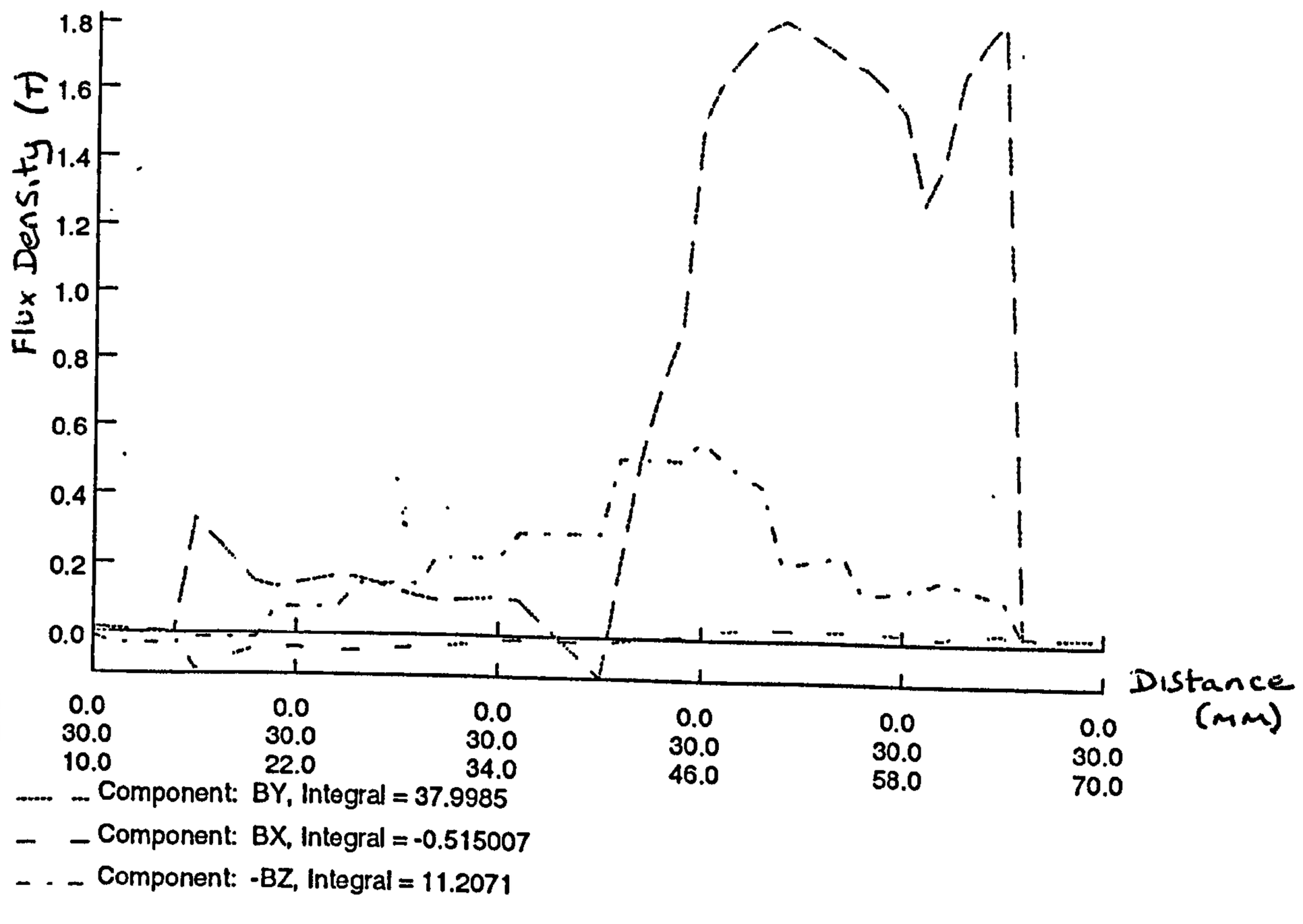


FIGURE 4.16 - Bx, By, and Bz Plot for Stator

The flux density distribution half way up the stator pole (figure 4.16) shows the X-direction component of flux density (B_x) has remained small. However there is a significant increase in the B_z component as more flux will now flow in the Z-direction in orientation. This is due to flux flowing into the stator back-iron. On the right hand side next to the air boundary, a second peak is produced by the end effect of the coil. The B_y component then falls and then peaks again towards the centre of the stack

Key parts in this analysis to note, are that the rotor iron sides generally confine the flux to the aligned teeth section. In the air-gap the flux does tend to fringe increasing the base area of crossing. In the stator pole the flux shifts into the centre of the stator pole as flux distribution now has a larger amount of flux travelling in the Z-direction. When 2D FEA is undertaken the analysis uses an average flux density, therefore an average area is required. Generally the length of the Z-direction dispersion changes, giving around 15% increase in depth. For a flux tube calculation method the areas can be individually increased for the differing lengths. Therefore in 2-D FEM analysis the height of the stator pole should not be too long.

4.9.1 Magnet

The magnet plays an important role in the hybrid stepper. It generates flux when there is no current excitation into all the regions of the motor. The magnet is a non-linear source, and hence its MMF and flux will change due to loading. For some magnet materials it may be possible to simplify the relationship between MMF and flux to a linear source, but for magnet types like Alnico there is a non-linear relationship between flux and MMF. Generally the magnet flux travels through the parts of the motor with the lowest reluctance

and so the flux is distributed disproportionaly. The flux from the magnet produces a near sinusoidal relationship of flux-linkage to position into a phase winding. The magnet has a direct effect on the torque output of the motor as the flux-linkage of the coil with position, produces the magnet back EMF effect. This back EMF induced into a single phase is estimated by;

$$EMF_{mag} = -N\frac{\mathrm{d}\phi}{\mathrm{d}t} = -\mathrm{N}\frac{\mathrm{d}\phi}{\mathrm{d}\theta}\frac{\mathrm{d}\theta}{\mathrm{d}t} \approx \omega n_r N\Phi_{mag}\, sin(n_r\theta). \qquad (4.27)$$

Where Φ_{mag} is the peak flux generated by the magnet, n_r number of teeth on the rotor.

The torque produced by a phase is;

$$T = -\frac{EMF_{mag}\, i}{\omega} + \tfrac{1}{2} i^2 \frac{\mathrm{d}L}{\mathrm{d}\theta}. \qquad (4.28)$$

The second term which is torque produced by the change of phase inductance, is sometimes considered to be relatively low compared to the magnet contribution. For example when the motor is running under low currents and is not saturated, it has been seen that even for low excitation, saturation occurs on the teeth profiles. Accordingly for general operation of the motor the main source of back EMF is from the magnet, and its magnitude is directly related to the peak flux linkage from the magnet. This would indicate that the stronger the magnet the greater the torque produced. The last statement would again be subject to saturation effects in the iron of the motor.

The holding torque characteristics of the motor can be calculated by treating the motor as two counteracting variable reluctance motors [61]. The co-energy area stored between the points;

$$point_1 = mmf_{coil} + \tfrac{1}{2} mmf_{mag}, \qquad (4.29)$$

$$point_2 = mmf_{coil} - \tfrac{1}{2} mmf_{mag}, \qquad (4.30)$$

(mmf_{coil} *and* mmf_{mag} are the instantaneous MMF's produced by the coil and magnet respectively.)

To obtain the maximum torque for a given coil excitation the magnet needs to be producing an amount of MMF which places its value in the middle of the largest area (figure 4.17).

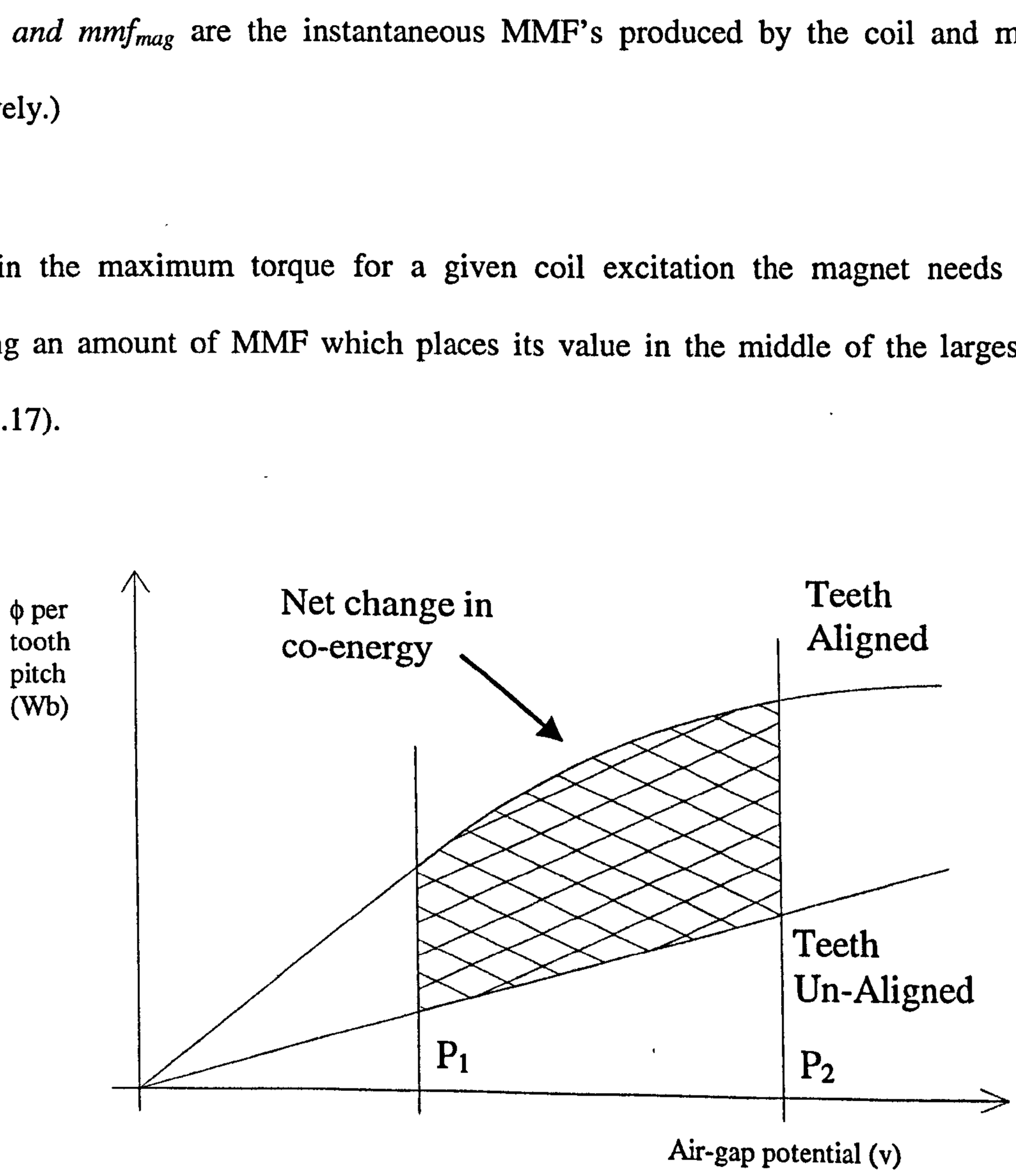


FIGURE 4.17 - Window for Magnet Co-energy

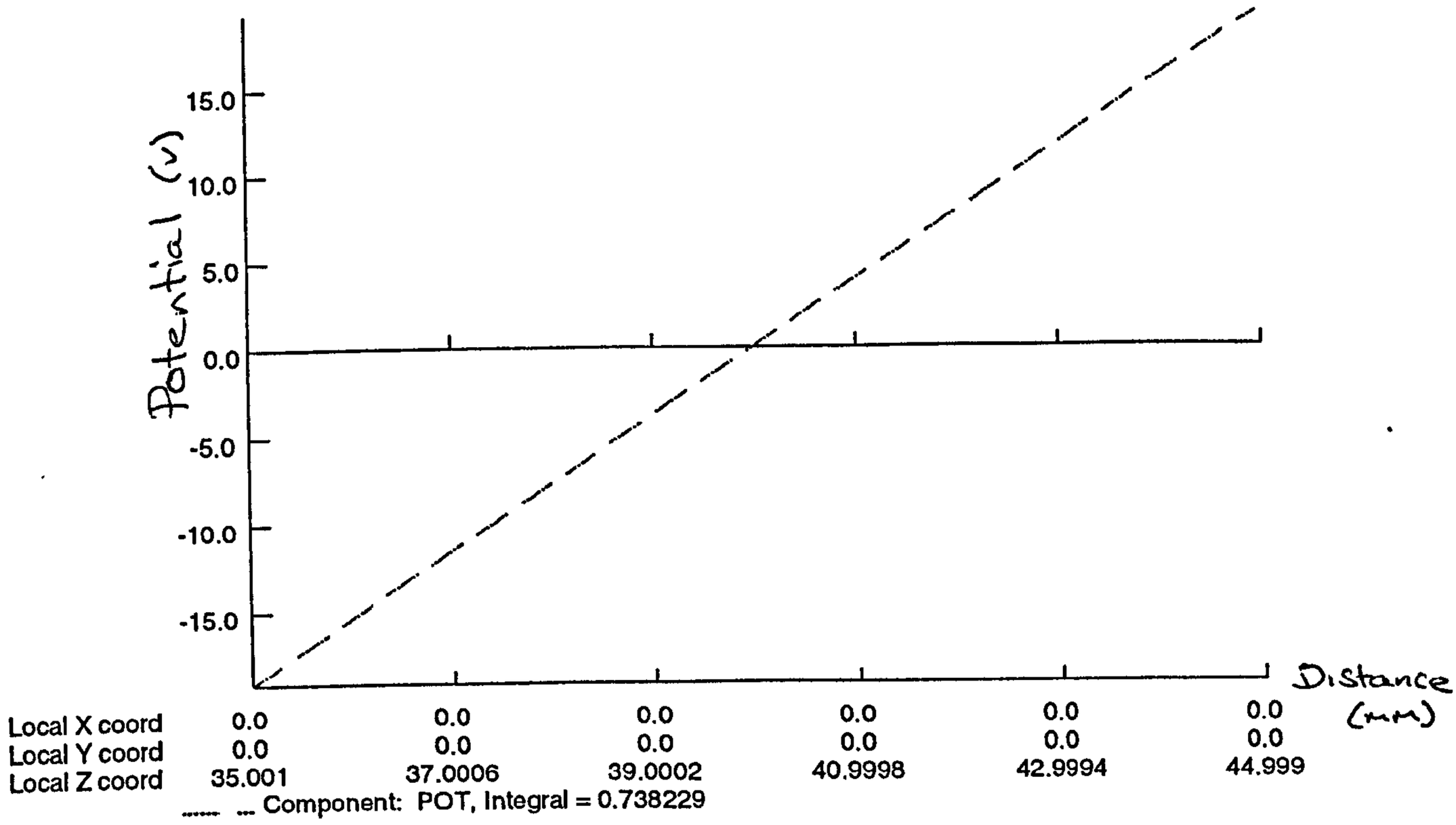


FIGURE 4.18 - Magnet Zero Potential

The magnet has a varying potential along its axial length, with a zero potential in the centre. This is a significant property of the magnet as it will allow symmetrical modelling of the magnet (figure 4.19).

4.9.2 Laminations

In 2-dimensional machines, i.e. where only a radial field distribution exists, laminations resist the flow of eddy currents. Unfortunately the hybrid motor's operation principles require flux to flow perpendicular to the laminations.

4.9.3 Perpendicular Flux

Concentrating on flux which travels parallel to the laminations, the effects of laminations added to the reluctance of the circuit is small (increasing permeability of the circuit). The following equation shows the change of permeability in the same plane of the laminations with the packing factor of the laminations [48].

$$\mu_{plane} = pf \times \mu_{iron} + (1 - pf)\mu_0, \qquad (4.31)$$

where *pf* is the packing factor of the laminations and is defined as the percentage of a lamination stack, which can be considered as solid.

The increased permeability in the same plane as the laminations affects the value of flux by a small percentage and in most cases can be assumed negligible. In some cases the laminations constrain certain areas to behave more two dimensionally. Such effects assist our accuracy when applied to certain motor sections such as the rotor/ stator tooth regions.

4.9.4 Cross lamination Effects

Cross lamination effects are difficult to analyse, as the inter lamination is unlikely to be uniform across a whole lamination stack. The biggest cross lamination effects take place in the stator back iron. This is where the flux is required to cross the lamination. In the stator poles at the front and back of the motor, close to the air-gap, the flux crosses in a mainly Y-directioned manner. As the flux goes up the stator pole it becomes more z-directional.

The same will happen in the rotor stacks. Figure 4.19 gives an understanding of what a lamination stack consists of.

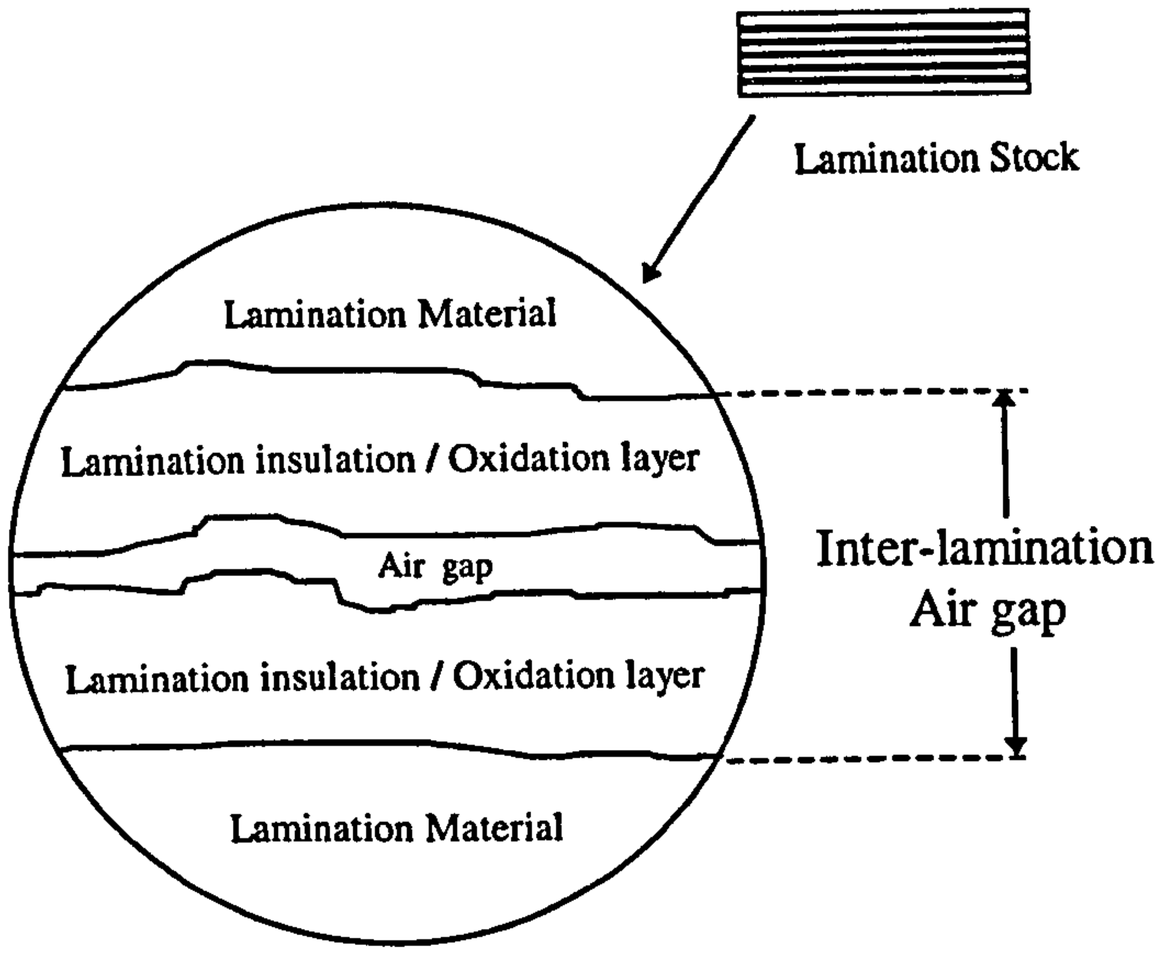


FIGURE 4.19 - Lamination Structure

The lamination can be seen in its most basic form as a steel sheet separated by a non-conductive layer. It has to be accepted that the actual physical structure of the inter lamination air-gap cannot be truly modelled by a uniform packing factor. This is because of the complications introduced from air spaces that lie between laminations and is a result of the laminations' surface roughness. This surface roughness on the lamination surfaces, causes the air space between the laminations to be non-uniform. Hence it is not known if the air-gap is greater or less than the lamination insulation layer. Figure 4.20 shows the lamination insulation thickness as much larger than the physical air thickness. The actual thickness of the air space could be much larger than the insulation thickness. The inter-

lamination air gap is highly reluctant, this layer will affect the performance of the motor because of the increased reluctance.

Many analyses discount the back iron as having negligible reluctance when it is not saturated [33,61,62]. It is true that the back iron has generally a low flux density distribution, but the large volume that may be involved makes it an important factor. If the back iron and inter-lamination effects are seen as a combined issue, it does not take a large value of flux density to cause a detrimental change in motor performance. Flux flowing across laminations will have to cross extra amounts of air and adhesive. The packing factor is used to give an estimate of the change in cross lamination permeability,

$$\mu_{cross} = \frac{\mu_0 \mu_{iron}}{pf \times \mu_0 + (1 - pf)\mu_{iron}} \quad , \qquad (4.32)$$

This generally translates to an increase in reluctance of over 10 times making this a very significant part of the model.

4.9.5 End Effects

Referring back to figure 4.16, which is a plot of the stator poles with an excited phase winding, this shows that flux in the pole appears to have an extra peak nearer the end of the machine(air region). This effect can be accounted for by considering the 3-dimensional effects that arise at the end of the machine stack [49]. The major effect is the end winding flux, that is the flux lines generated from the conductor area that are outside the machine

stack length. These flux lines link the phase winding via the main magnetic circuit. The other end effect is axial fringing where the flux lines travel in the axial direction through the air from the stator to the rotor. The effect is greatest when the teeth are in the aligned position. These effects are generally only accounted for in three dimensional analysis. This effect is also noticeable on the rotor end-caps where the magnet boundary occurs.

4.9.6 Summary of Contributory Factors to the Magnetic Behaviour

All the above are contributors to the performance of the motor. Table 4.2 summarises the key issues and effects.

Property	key effects	Importance
Air-gap/ rotor/ stator interface	Determining the flux linkage of the motor.	Extremely
Magnet	Major MMF source of the motor at low excitations.	Important
Back iron/ inter lamination effects	Reduces the effectiveness of the magnet by adding additional MMF loses.	Important
End effects	Causes extra flux linkage in the circuit	Noticeable

TABLE 4.2 - Three Dimensional Effects

4.10 Deriving a Simpler Model of the Hybrid Stepping Motor

Two important features have been shown in the analysis of the motor in 3-D FEM. The first is that the magnet has a zero potential point half way along its axial length. The second is that it can be shown that a second zero potential exists along the centre of the back iron (figure 4.20). These zero potential points can allow a symmetrical model to be developed.

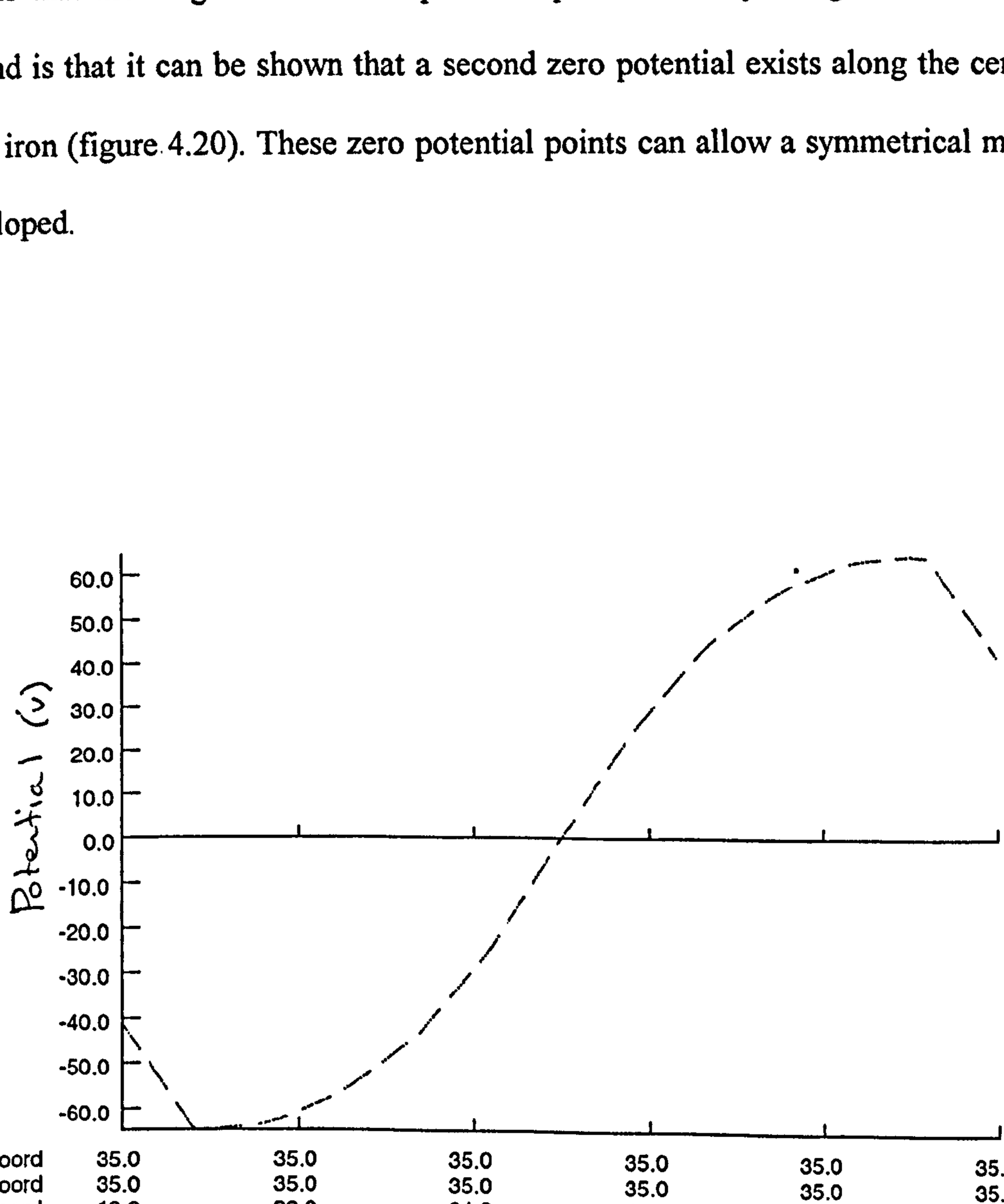


FIGURE 4.20 - Zero Potential in Stator Back Iron

Taking the regions of importance, the hybrid stepping motor can be simplified to give figure 4.21, which shows a full lumped element of a hybrid stepping motor. The magnet is modelled from calculating a load point on its B-H curve. This is converted to a MMF/ flux point using the dimensions of the magnet. The lumped elements and MMF supplies are represented in the figures as linear elements and linear MMF supplies. These elements and MMF supplies are modelled as non-linear elements and MMFs. Each branch is based on the permanence modelling discussed earlier in the chapter. Now using the zero potential points of the FEM analysis the centre of the magnet can be joined to the centre of the back iron. The magnet can be modelled as a source that produces half the amount of MMF per flux unit. The back iron reluctance will also become half of its original value. Figure 4.22 shows the new arrangement. It can be seen in figure 4.22 and 4.23, that each reluctance contains a parallel equivalent. We can simplify the circuit further by halving the reluctances as they are in parallel and of equal magnitude. Therefore we obtain the final simple circuit which gives fewer amounts of elements to calculate (figure 4.23).

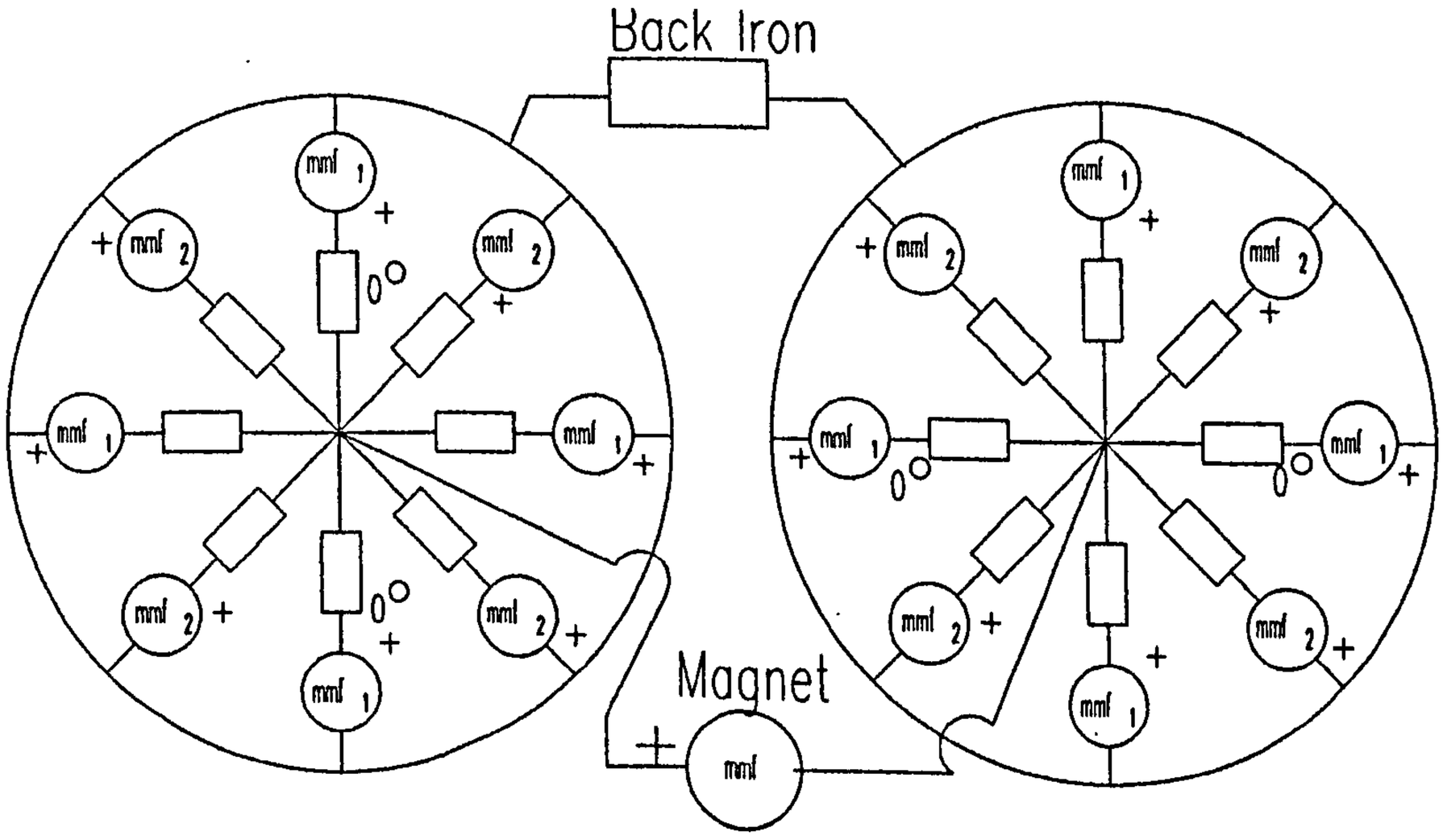


FIGURE 4.21 - Full Lumped Model of the Hybrid Stepping Motor

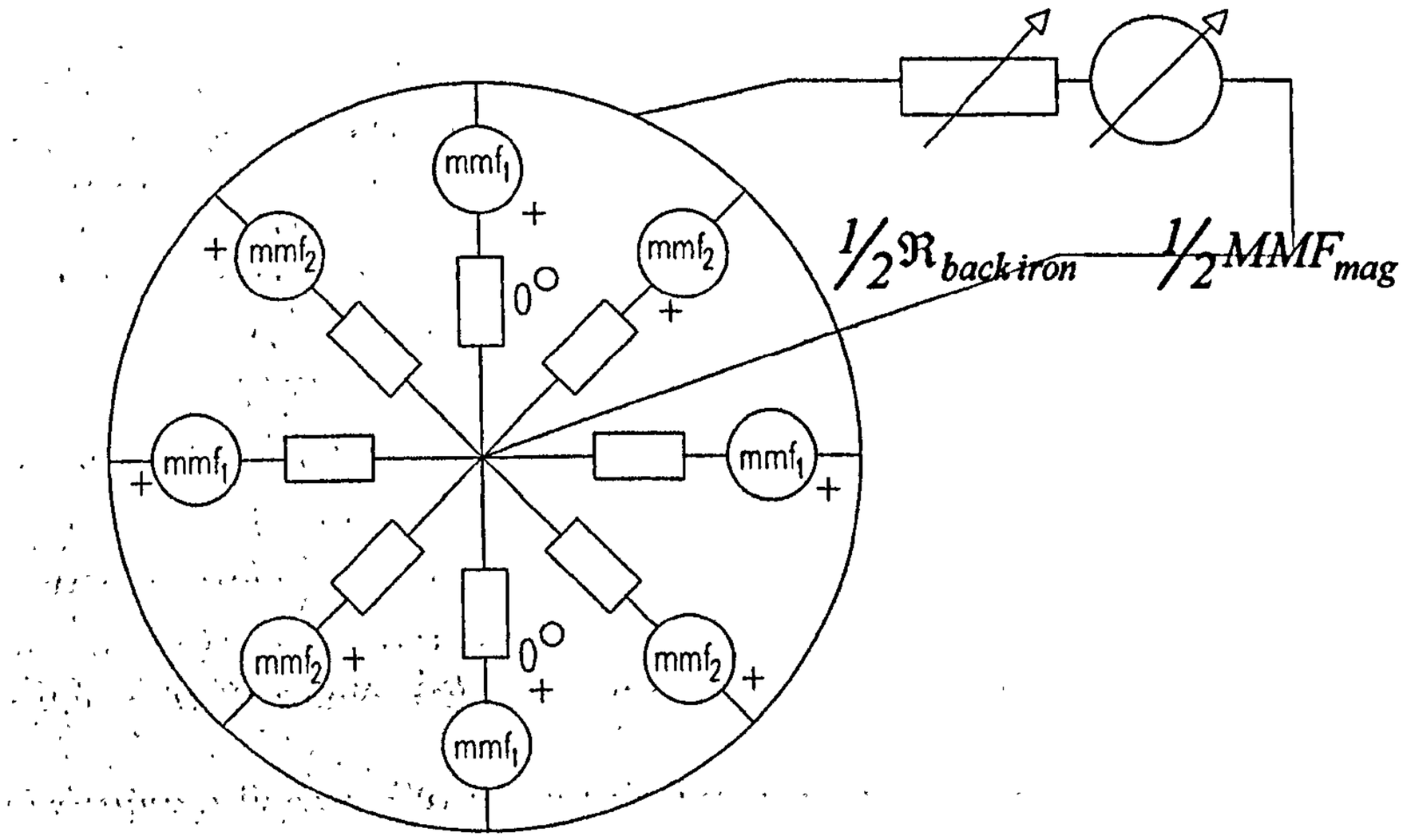


FIGURE 4.22 - Half Lumped Model of the Hybrid Stepping Motor

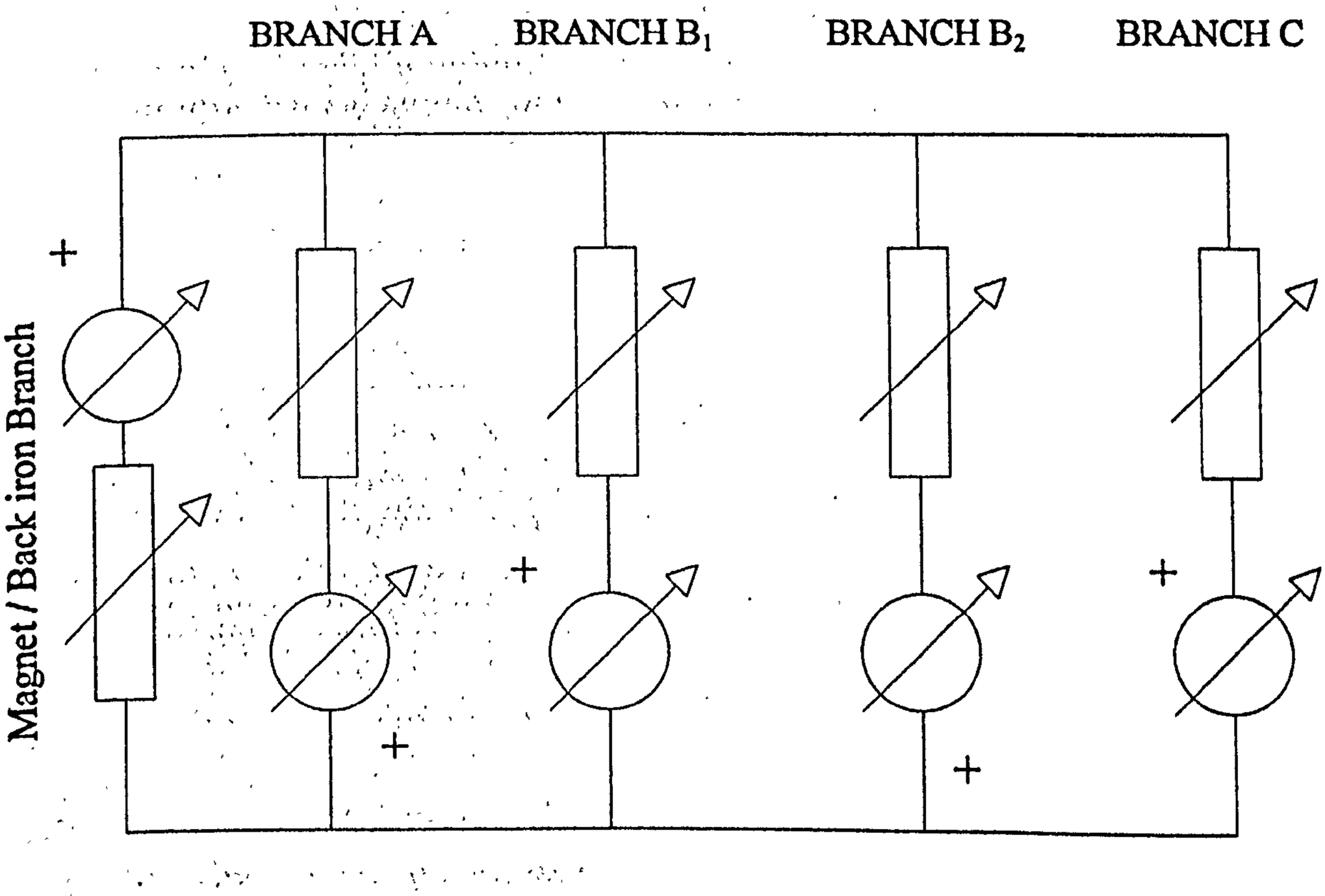


FIGURE 4.23 - Reduced Elements Network

4.10.1 Single Phase Excitation Case

Single phase excitation is when only one phase is switched on at any one time. This does not mean that there is no current in the second winding as the inductance of the coil can cause the current in the commutated phase to decay slowly. However in this static analysis the current is assumed to be in the excited phase only. In the single phase case the circuit can be represented as in figure 4.24.

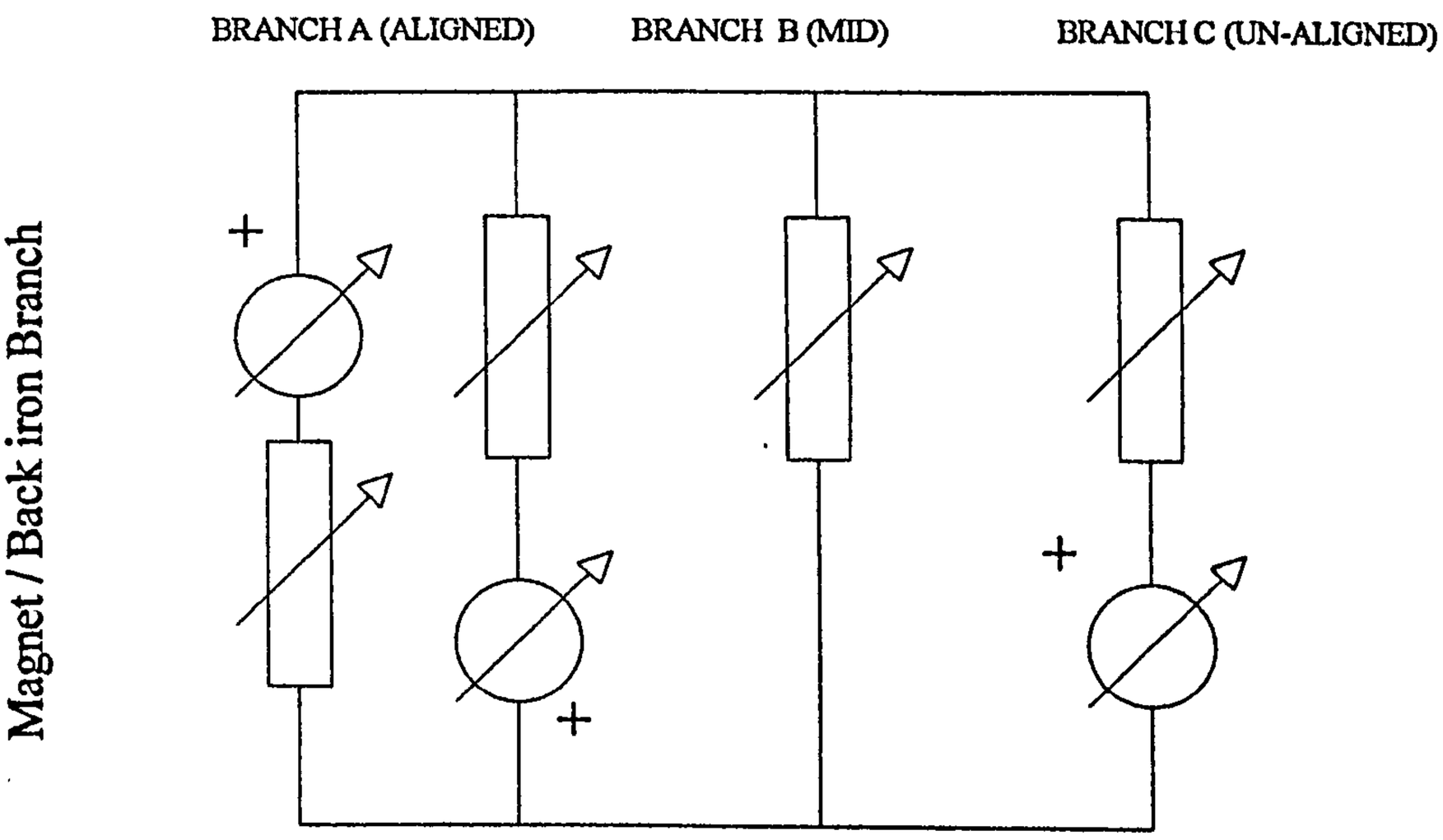


FIGURE 4.24 - Single Phase Network

It has four branches.

1) The magnet and back iron branch,

2) Branch A,

3) Branch B,

4) Branch C.

4.10.2 The Magnet and Back Iron Branch

The magnet is represented by an MMF source which varies according to its load point calculated from its B-H characteristics. These characteristics can either be represented by a table or an equation in the HyStep program.

Firstly the B-H characteristics are converted from B and H to Φ and MMF respectively.

B→Φ $$\Phi = B \times A_{mag}, \qquad (4.33)$$

H→MMF $$MMF_{\frac{1}{2}mag} = H \times \frac{1}{2}\ell_{mag}, \qquad (4.34)$$

Where $MMF_{\frac{1}{2}mag}$ is half the MMF produced by the full length magnet, A_{mag} is the magnet area, and ℓ_{mag} is the magnet length.

The back iron / inter-lamination component of this branch is treated as a reluctance section with a modified length and permeability. The modified length is the distance that the flux will travel in the stator back iron, and the rotor end-caps across the lamination air-gaps. It is estimated by;

$$\ell_{apparent} = \ell_{mag} + (K_z \times 4 \times \ell_{rstk}), \qquad (4.35)$$

ℓ_{rstk} is the rotor stack length.

The K_z coefficient is an approximation of how much of the rotor stacks, the flux travels across the lamination air-gap. From finite element studies 30% was found to be a good estimate thus K_z=0.3. The factor of four multiplier assumes that the travel by flux in the stator is directly related to that of the rotor end-caps.

The permeability of the stack is calculated by equation 4.31 and the B-H curve of steel.

From the original B-H curve steel, a particular flux density has a corresponding field strength H. The permeability of the iron is calculated by;

$$\mu_{iron} = \frac{B}{H}. \qquad (4.36)$$

This μ_{iron} is entered into equation 4.32 along with the packing factor to produce the modified permeability $\mu_{modified}$, which is identical to μ_z. The modified permeability is used to find the modified field strength $H_{modified}$, equation 4.37, which in turn can be used to find the increased MMF drop,

$$H_{modified} = \frac{B}{\mu_{modified}}. \qquad (4.37)$$

For this branch, there is now a reluctance and a MMF source. It can be seen that the back iron / inter-lamination effect will reduce the effectiveness of the magnet as a MMF source.

This can be viewed like an internal resistance to a power source. The greater this reluctance the less of the magnet MMF can be dropped across the other branches.

The magnet MMF versus flux characteristics are non-linear and are shown in figure 4.25A. A modified back iron relationship (solid) is shown along side a non laminated structure (dashed) of the same dimensions in figure 4.25B. The relationships are typically linear because they are affected by low flux density which falls within the linear region of the steel's B-H curve. This would change if the back-iron was made of a low saturating steel or a back iron design where the flux-density was allowed to rise above the linear region.

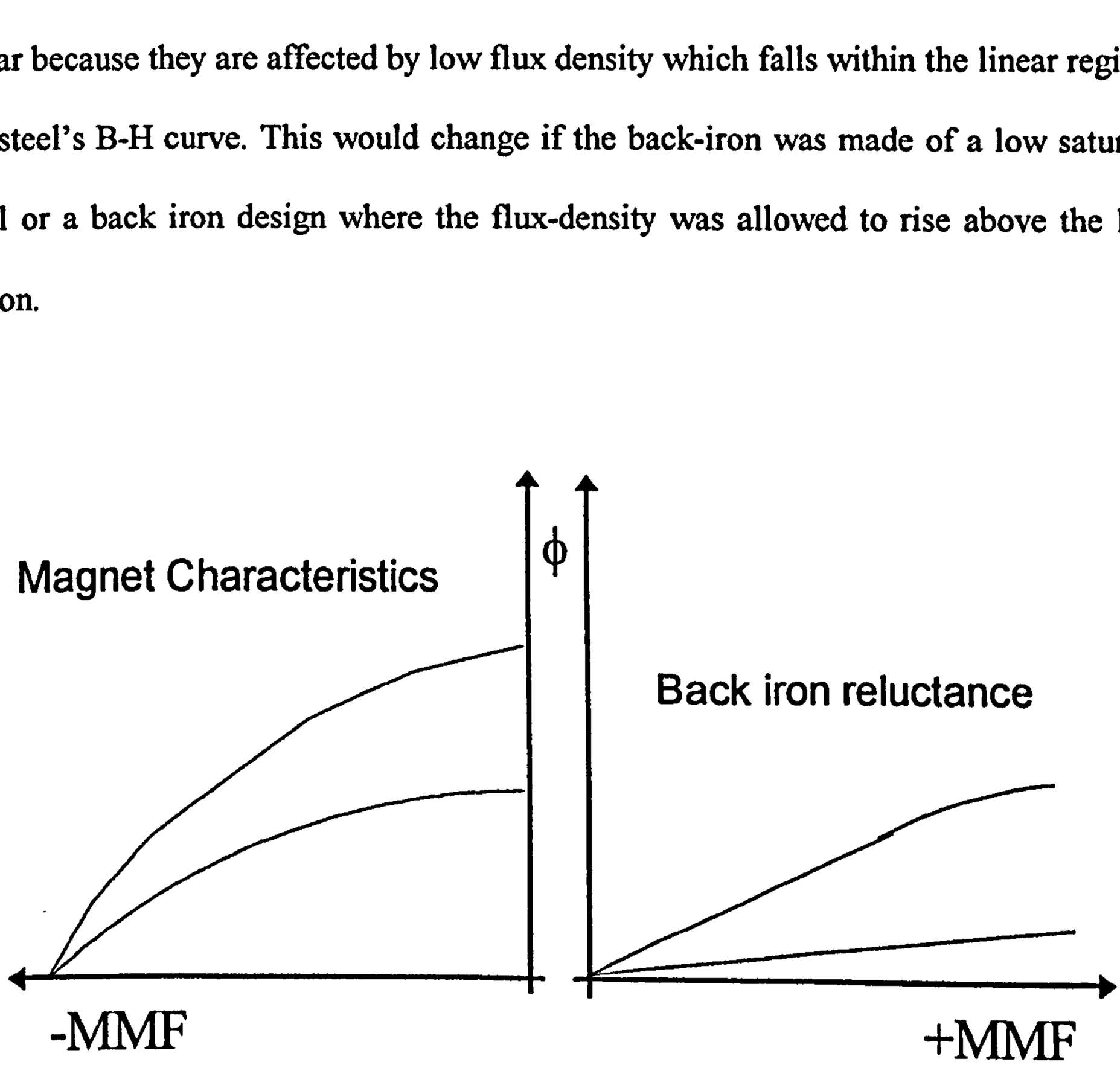


FIGURE 4.25 A/B - Magnet and Back Iron Flux versus MMF Characteristics

To represent the branch as a single identity, we need a representation that includes both the magnet source and the back iron. The MMF of the branch is the magnet MMF

produced minus the MMF dropped in the back iron. The MMF source of the magnet generally works within the second quadrant, so it has a negative value.

$$MMF_{branch} = -(MMF_{mag} + MMF_{back\,iron}), \qquad (4.38)$$

$$\phi_{branch} = \phi_{mag} = f(\mathrm{MMF}_{\mathrm{magnet}}) = \phi_{\mathrm{back\,iron}} = f(\mathrm{MMF}_{\mathrm{back\,iron}}). \qquad (4.39)$$

Both the back iron reluctance and the magnet MMF are a function of MMF. It is easier to create a new relationship for the whole branch than to try and make a combination of both. If the data is in tabular form then a linear interpolation method can be used. If the data is in the preferred form of an equation then the equations are summed to obtain the new results, table 4.3 shows how.

Flux value (Wb)	Magnet MMF (I)	Back Iron MMF Drop (I)	New MMF (I)
0	-1000	0	-1000
1e-4	-800	20	-780
4e-4	-100	100	0
5e-4	0	170	170

TABLE 4.3 - Effect of back iron MMF drop on magnet output

The dashed line in figure 4.25A shows the new relationship of the branch. It has the effect of shifting the peak flux more into the positive MMF region. We now have a relationship for the whole branch which can be stored as an equation or in table form.

4.10.3 The Equation of Magnet B-H characteristics

The MMF versus flux profile of the magnet and hence the total branch characteristics are represented by a negative MMF region. As described earlier, it has been shown that the B-H characteristics of steel can be fitted to equations of 4.3, 4.4, and 4.5. Likewise the MMF versus flux relationships may be modelled by similar equations, for example,

$$\phi = \frac{\alpha(mmf)^n}{\beta + (mmf)^n}, \tag{4.40}$$

or its related modified rational functions;

$$\phi = \frac{\alpha(mmf)^n + \kappa(mmf)^{n+1}}{\beta + (mmf)^n}, \tag{4.41}$$

$$\phi = \frac{\alpha(mmf)^n + \kappa(mmf)^{n+1}}{\beta + (mmf)^n + \varepsilon(mmf)^{n+1}}, \tag{4.42}$$

An offset (*of*) is required by HyStep for the equation to represent a positive quadrant equivalent. This is simply achieved by taking the absolute value of MMF when the flux is less than zero. In the resulting equation the offset is required to be added before solving, i.e.

$$\phi = \frac{\alpha(mmf + of)^n}{\beta + (mmf + of)^n}. \tag{4.43}$$

4.10.4 Branch A

In single phase operation, branch A will include a phase winding and a rotor aligned position reluctance for calculating the aligned flux-linkage curve (in conjunction with branch C). The reluctance will be made up of two identical reluctances in parallel. The same combination is used to find the single phase mid flux-linkage curve but with the coil current set to zero. The reluctances have been found by two dimensional methods, either 2-D FEA or elliptical path methods. The MMF in the windings, needs to be converted from the current and the number of turns,

$$MMF_{coil} = iN, \qquad (4.44)$$

where N is the number of turns on the particular pole.

Because the branch's reluctance is actually two parallel reluctances the flux flowing will need to be doubled. The branches MMF versus flux relationship needs to modified by the MMF in the windings. The relationship for the branch flux is;

$$\phi_{branch\ A} = 2 \times f(MMF_{REL} - MMF_{coil}), \qquad (4.45)$$

where f indicates the 'funtion of'.

The subtraction of the coil MMF to the reluctance MMF indicates that a larger flux will result for an increase in winding current. This addition of MMF means that when the coil current is increased flux in that branch will rise. For example consider figure 4.26. This diagram shows a simplified aligned tooth flux versus MMF profile. The flux flowing in this branch will be dictated by the MMF produced by the magnet if no current is flowing in

the phase winding. For example if the MMF produced by the magnet was 30 Ampere turns then the flux flowing in the branch will be approximately $0.82e^{-4}$ Webers. If for this simple example the magnet MMF does not change but there is now a MMF produced by the phase winding of 20, the flux will now increase to approximately $1.1e^{-4}$ Webers. This is because the MMF equation of the complete circuit is;

$$MMF_{mag} = MMF_{Branch\ A} = MMF_{Branch\ B} = MMF_{Branch\ C}, \qquad (4.46)$$

which implies that the value of MMF_{REL} must equal 50 Ampere turns for a magnet MMF of 30 Ampere turns and a coil MMF of 20 Ampere turns, i.e. 50=70-20.

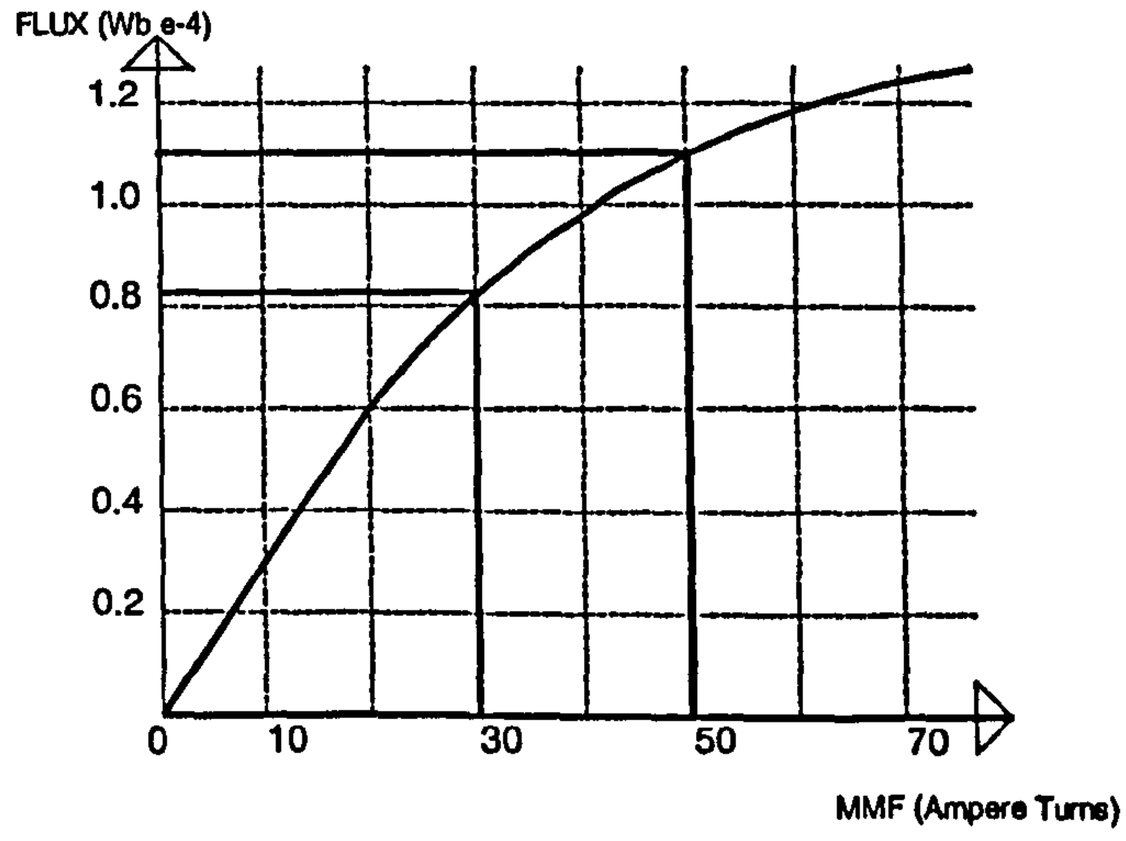


FIGURE 4.26 - Aligned Rotor Teeth Flux/ MMF Characteristics

4.10.5 Branch C

Similar to the branch A, but the coil MMF is reversed, and the reluctance is made up of the un-aligned rotor teeth. When the current is positive the MMF opposes that of the magnet. The flux equation in turn is;

$$\phi_{branch\ c} = 2 \times f(MMF_{REL} + MMF_{coil}) \qquad (4.47)$$

As the current in the winding increases a negative flux may occur. HyStep calculates this negative flux by checking if the MMF is less than zero. If the MMF is less than zero then the flux will be governed by

$$\phi_{branch\,c}\,(MMF_{REL} \pm MMF_{coil} < 0) = -f(absolute(MMF_{REL} \pm MMF_{coil}))\,. \qquad (4.48)$$

4.10.6 Branch B

Branch B represents when the rotor teeth are half aligned with the stator teeth, this position is also known as the mid rotor position. In single phase operation finding the aligned flux-linkages due to branches A and C, does not require branch B to have an excitation in its phase winding. Branch B can then be seen as four parallel reluctances lumped into one reluctance. The flux value is modified to four times the value of a single position.

$$\phi_{branch\,B} = 4 \times f(MMF_{REL}), \qquad (4.49)$$

4.11 Example of Circuit Set-Up to Find Aligned and Unaligned Single Phase Flux Linkages.

To clarify the last section an example is now given. Here the excitation scheme is single phase, i.e. branches A and C will have coil excitation. This example will allow the aligned flux linkage path to be found. The aligned position is represented in figure 4.27 which shows the front of a hybrid stepping motor. The rotor is sitting in the stable equilibrium position with the top stator pole A aligned with the rotor teeth. The rotor magnet flux is

assisting pole A's MMF, when the current flowing in the coil is positive. The network circuit will be as figure 4.24, with branch A being the aligned rotor, branch B, the mid rotor position and branch C, the un-aligned rotor position relative to the pole stator teeth.

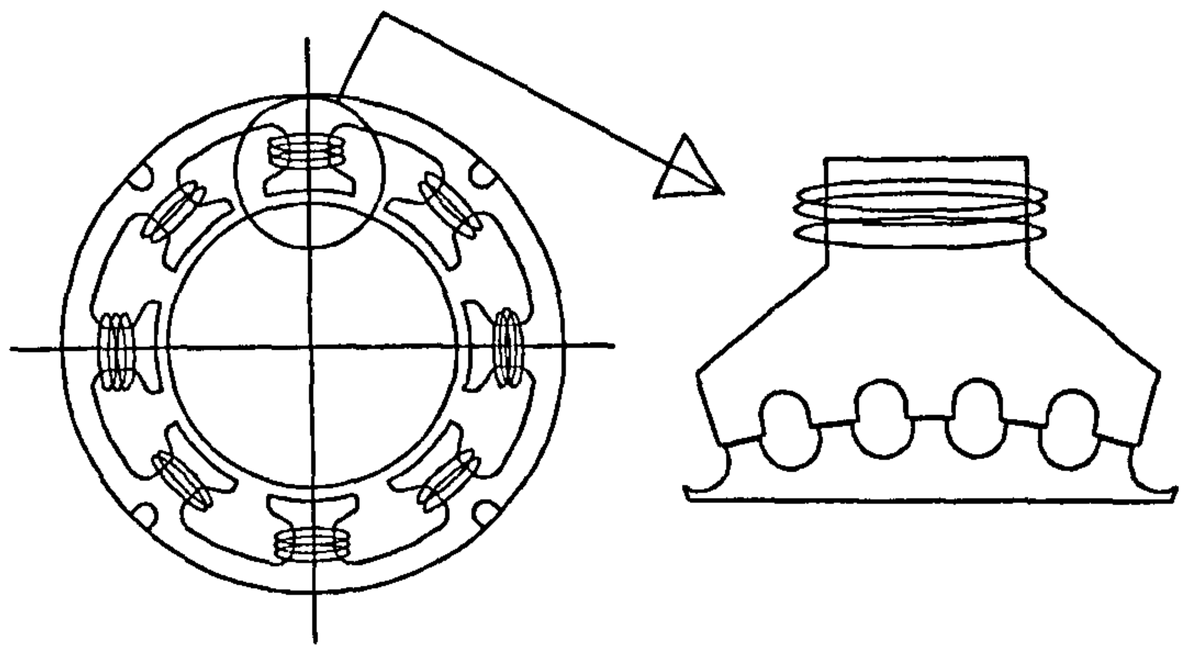


FIGURE 4.27 - Front View of Motor Showing Aligned Position

There will be no current in the branch B, the MMF in the coil in branch C will be equal in magnitude but opposite in polarity to that of branch A. The main equation of the network states that the flux flowing into the back iron branch is equal the sum of the fluxes flowing in Branches A, B and C;

$$\phi_{mag} = \phi_A + \phi_B + \phi_C, \quad (4.50)$$

where $\phi_{mag,A,B,C}$ are a function of MMF described by equations 4.38, 4.44, 4.46, and 4.48 respectively. This can be solved by a Newton Ralpson method as long as the relationships of the branches are known.

HyStep starts its calculations from negative maximum current and increases up to the maximum current. This generates a MMF in the coils which interacts with the magnet

MMF to cause flux to flow in the aligned and un-aligned rotor positions, Branches A and C.

The aligned flux linkage curve (refer to figure 4.3) is the combination of the aligned and un-aligned rotor position flux linkages. The 'aligned flux linkage' flux is

$$\phi_{AL} = \phi_A - \phi_c . \qquad (4.51)$$

The aligned flux linkage curve can be generated by

$$\lambda_{AL(i)} = N\phi_{AL(i)}, \qquad (4.52)$$

where i is the current value and N the number of turns in the phase.

4.11.1 Generation of Unaligned and Mid Flux Linkage Curves

The un-aligned curve (figure 4.4) can be easily generated from the aligned curve as described in section 4.4. The mid curve can be calculated by using the circuit in figure 4.23. The MMF produced by Branches A and C will be zero. The circuit can also be produced from figure 4.24. The Branch B reluctances of figure 4.24 are four parallel reluctances and need to be split into two branches of two parallel reluctance's. The coils on Branch A and C need to be switched to Branches B_1 and B_2 respectively. The equations of the new branches will be ;

$$\phi_{branch\ B_1} = 2 \times f(MMF_{REL} + MMF_{coil}) , \qquad (4.53)$$

$$f_{branch\ B_2} = 2 \times f(MMF_{REL} - MMF_{coil}) , \qquad (4.54)$$

and

$$\phi_{mag} = \phi_A + \phi_{B_1} + \phi_{B_2} + \phi_C . \qquad (4.55)$$

The 'mid flux linkage' flux can be found by

$$\phi_{MID} = \phi_{B_2} - \phi_{B_1} . \qquad (4.56)$$

The Mid flux linkage curve can be generated by

$$\lambda_{MID(i)} = N\phi_{MID(i)}, \qquad (4.57)$$

4.12 Discussion

The analysis provided by the three dimensional finite element modelling has provided a useful insight into the static characteristics of the hybrid stepper motor. The motor has been broken down in regions which have allowed simplified models to be applied to particular issues associated with the motor. Combining these models has allowed a lumped circuit to be produced for the hybrid stepping motor. This lumped circuit has been converted in to a computational simulation packaged named HyStep. HyStep is used for sizing and computing the fundamental areas of performance for the motor. The program is capable of calculating the motor magnetisation curves using internal (analytical) routines. However for non-standard tooth geometry suggestions for using an external 2-D finite element package have been suggested.

New rational functions for modelling B-H characteristics have been presented. Modelling of the flux line distribution by elliptical methods has been examined.

CHAPTER 5

MODELLING VERIFICATION AND DYNAMIC ANALYSIS

The previous chapter has described the static modelling procedure for the hybrid stepping motor using computational analysis, namely HyStep and finite element analysis. The accuracy of these procedures has been checked by modelling the Stebon 1101 type motor, and comparing the predictions with experimental results. The Stebon 1101 is a single stack, bipolar motor, with a 42 NEMA sized frame. This chapter will express the strengths and weaknesses of the particular design methods. Starting with torque calculations it follows on to dynamic verification by considering a dynamic model of the motor. The proposed method comes through characteristics observed in the 3-D static modelling of the hybrid stepping motor and experimental test results.

5.1 Torque Calculation

Torque can be calculated by using the flux-linkage curves. The area enclosed by the curves is named the co-energy of the system. Torque is given by the rate of change of co-energy with respect to the movement of the rotor, at a constant excitation. For single phase excitation, the hybrid stepper motor is not a singly excited system, as it includes a magnet source. The energy balance equation can be written as;

$$\text{(Electrical and magnetic power in)} = \begin{pmatrix}\text{mechanical} \\ \text{output power}\end{pmatrix} + \begin{pmatrix}\text{rate of increase} \\ \text{in magnetic energy}\end{pmatrix}, \qquad (5.1)$$

where the electrical power in is due to the EMF induced in the phase along with the current flowing, and the magnetic power is the voltage generated by the magnet multiplied by the current flowing.

Equation 5.1 can also be written as

$$dW_e = dW_{mech} + dW_{field}. \qquad (5.2)$$

Where dW_e is the change in electrical input power, $dW_{mech.}$ is the change in mechanical output power, and dW_{field} is the change in energy stored in the field.

Introducing Faraday's law, dW_{field} can be expressed as

$$dW_{field} = id\lambda - Td\theta, \qquad (5.3)$$

Where i is the current in a particular phase winding, and λ is the flux linkage in a particular phase; the term T represents (average) torque while $d\theta$ is the angular displacement. However the $d\lambda$ term is not caused solely by the current flowing through the phase inductance, as there will be flux linkage due to the magnetic. Therefore equation 5.3 can be rewritten as

$$dW_{field} = i\left(\lambda_{mag} + \lambda_{coil}\right) - Td\theta \qquad (5.4)$$

Where λ_{mag} is the flux linkage in the single phase winding due to the magnet, and λ_{coil} is the flux linkage in phase one due to the current in the winding.

By application of the chain rule the increment in field energy can be expressed in terms of flux linkage λ, and rotor position θ,

$$dW_{field} = \left.\frac{dW_{field}}{d\lambda_{mag}}\right|_{\theta,\lambda_{coil}=const} d\lambda_{mag} + \left.\frac{dW_{field}}{d\lambda_{coil}}\right|_{\theta,\lambda_{mag}=const} d\lambda_{coil} + \left.\frac{dW_{field}}{d\theta}\right|_{\lambda_{coil},\lambda_{mag}=const} d\theta. \quad (5.5)$$

Comparison of equation 5.4 and 5.5 reveals that the last term of equation 5.5 is related directly to the torque term of equation 5.4.

$$T = -\frac{dW_{field}\left(\lambda_{mag(const)}, \lambda_{coil(const)}, \theta\right)}{d\theta}. \quad (5.6)$$

The co-energy of a system may be derived from,

$$W_c = (\lambda_{mag} + \lambda_{coil})i - W_{field}. \quad (5.7)$$

In order to estimate the average torque produced in a single phase hybrid stepping motor system, it is essential to have the flux linkage relationship curves of the phase winding for the aligned and the mid positions. The change in co-energy of the current carrying phase is the trajectory of the flux linkage curve. To understand the change of co-energy with respect to current and rotor position, the current excitation pattern needs to be defined. In the following example (figure 5.1) the current in the phase is assumed as an ideal square current and the current is switched on when the rotor is in the half-aligned position.

Looking at the change in co-energy in the first 1.8 degrees. For the pole A, subject to current excitation, the rotor moves from the mid position to the aligned. The co-energy curve is the area enclosed from the mid curve to the aligned on the flux linkage diagram.

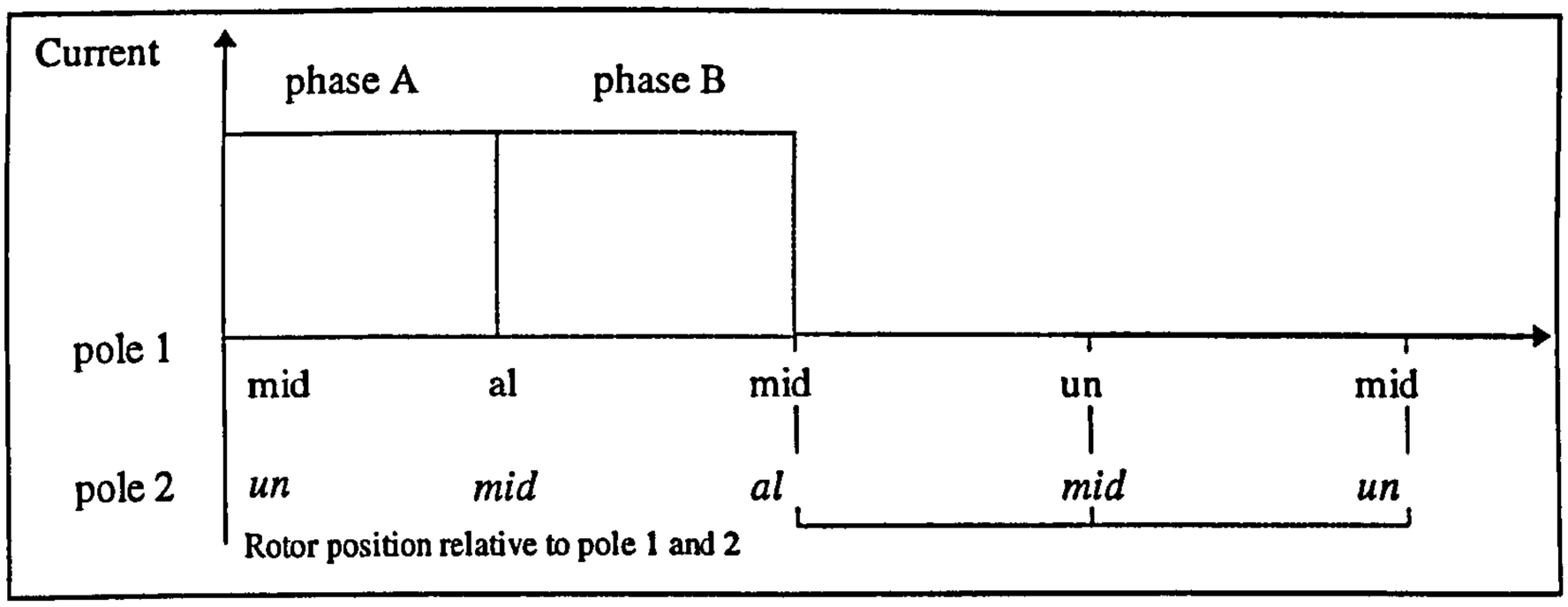


FIGURE 5.1 - Single Phase Winding Excitation

The co-energy repeats itself every 1.8 degrees. When the current becomes negative, the area of co-energy is between the un-aligned and mid curves, for negative current values. This area is equal in size to the aligned to mid region for positive currents. The average torque can be calculated by

$$T = \frac{\Delta W_c \times (\text{number of steps per rev})}{2\pi} = \frac{\Delta W_c \times (360^\circ/1.8^\circ)}{2\pi} = \frac{\Delta W_c \times 100}{\pi}. \quad (5.8)$$

In this example torque will be calculated from the co-energy area enclosed by the mid curve to the aligned curve for a positive current.

5.1.1 Two Phase System

The magnetic equivalent circuit of figure 4.22 is used to perform a two phase analysis. Now each branch has the ability to carry current in its winding. The analysis is much the same as for the single phase case except that the 5 branches will be represented by equations 4.38, 4.45, 4.47, 4.49, 4.54, and 4.55 (from chapter 4).

5.1.2 Average Torque

The system under analysis is a two phase doubly excited system and is very similar to the single phase system in terms of deriving the average torque from equation 5.8. The field energy can be written in terms of i_1 and i_2.

$$dW_{field} = i_1 d\lambda_1 + i_2 d\lambda_2 - T d\theta \ , \qquad (5.9)$$

where i_1 and i_2 are the currents in each phase winding, λ_1 is the flux linkage in phase one (due to both the current in the windings and the magnet flux), and λ_2 is the flux linkage in phase two (again due to currents in the windings and magnet flux). A similar torque equation to equation 5.5, can be developed by following the same method which obtains equation 5.4 from 5.3.

$$T = -\frac{dW_{field}\left(\lambda_{1(\text{const})}, \lambda_{2(\text{const})}, \theta\right)}{d\theta}, \qquad (5.10)$$

where λ_1 and λ_2 are the flux linkages in phases 1 and 2 respectively.

The co-energy of the two phase system may be derived from,

$$W_c = \lambda_1 i_1 + \lambda_2 i_2 - W_{field} \, . \qquad (5.11)$$

The torque T is given as the rate of change of co-energy with respect to the rotor angle,

$$T = \frac{dW_c\left(i_1, i_2, \theta\right)}{d\theta}, \qquad (5.12)$$

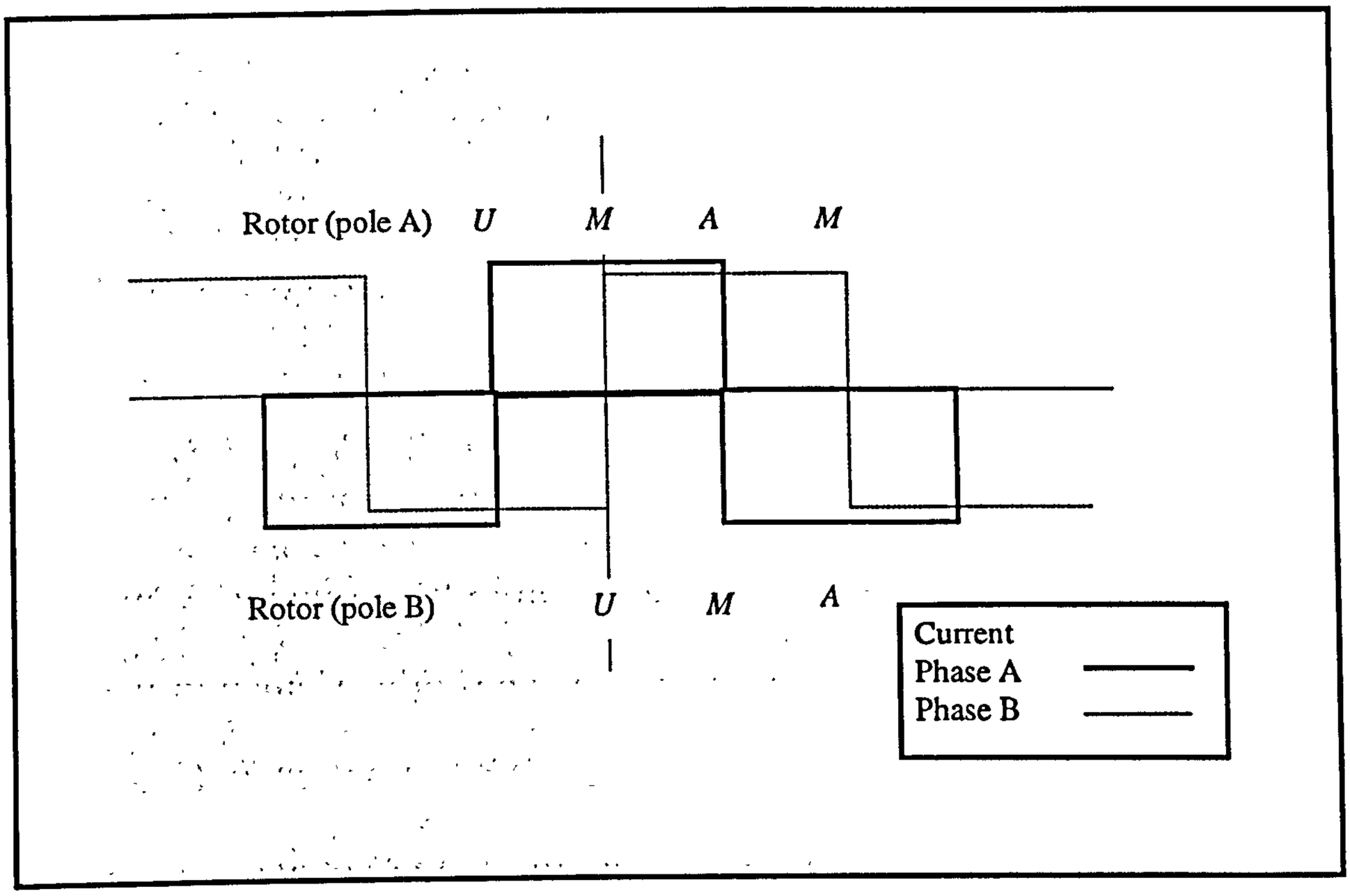


FIGURE 5.2 - Two phase excitation System

Again the flux linkages in both windings are required to be known. The current excitation scheme of figure 5.2, shows the rotor shift in relationship to each pole. (Current levels unequal for clarity).

From the centre of figure 5.2, and looking at phase B which is wound on pole B, the current rises from zero to rated current when the rotor is in the un-aligned position. The current stays at this rated current until the rotor is in the aligned position, where the current falls. The trajectory on the flux-linkage curve is therefore as in figure 5.3a. For pole A which is subject to phase A, the rotor is at mid position when the current is at rated current. The rotor moves to the aligned position where the current drops to negative rated current. The rotor moves to the next mid point at the same current. The total rotor movement is 3.6°. The trajectory on the flux linkage curve is as in figure 5.3b. The areas of the trajectories of figures 5.3a and 5.3b, are equal, as the positive current region of

figure 5.3b is equal to the un-aligned to mid region of figure 5.3a. Therefore for 3.6° movement and ideal current pulses the co-energy will be twice the area between aligned and un-aligned, for one flux-linkage diagram (positive current only).

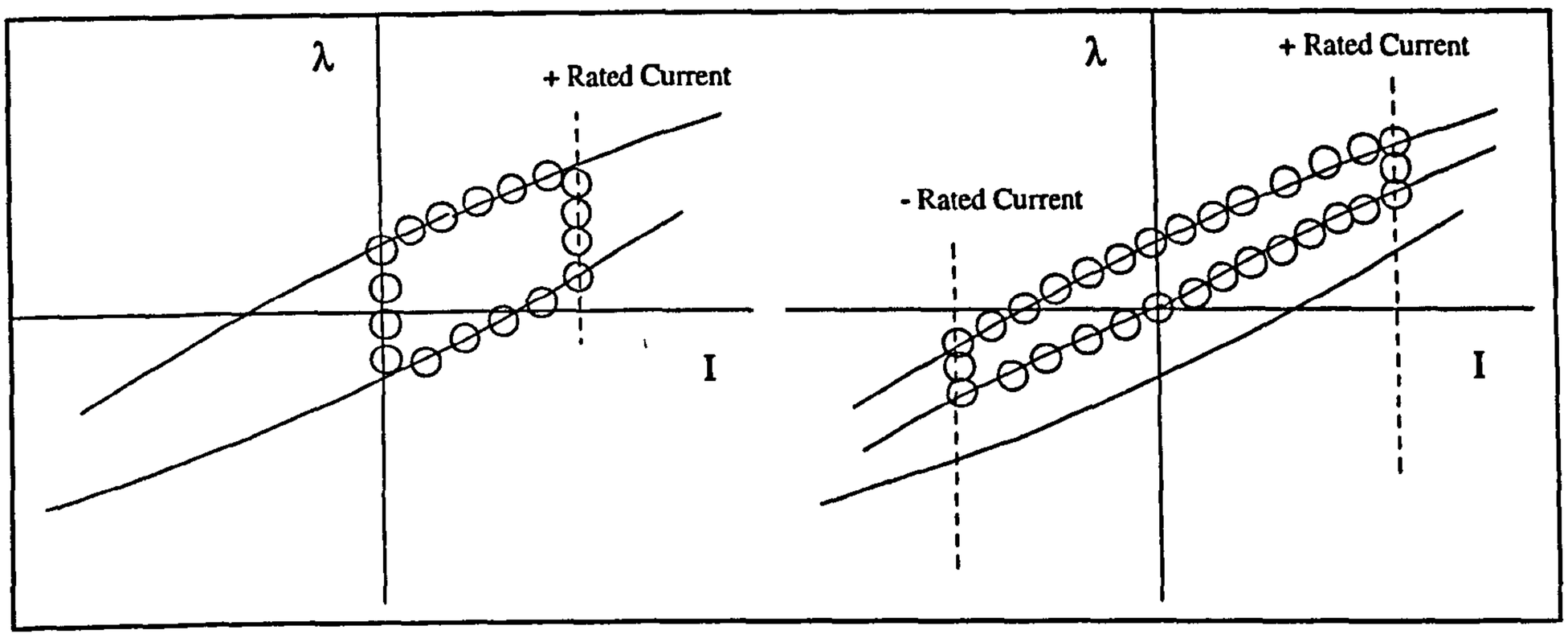


FIGURE 5.3A/B - Two-Phase Flux Linkage Curves

The average torque can therefore be calculated by an adaptation of equation 5.8,

$$T = \frac{\left(\Delta W_{c1} + \Delta W_{c2}\right) \times (\text{number of steps per rev})}{2\pi} \quad , \qquad (5.13)$$
$$= \frac{2 \times \Delta W_{c1} \times ({}^{360°}/_{3.6°})}{2\pi} = \frac{\Delta W_{c1} \times 100}{\pi}.$$

In this example this is for the co-energy enclosed by the aligned to un-aligned curves on the flux-linkage curves, for a positive current.

5.2 Magnet Detent Torque

The theory presented for calculating torque is specifically used for calculating average torque. For the three main positions on the flux-linkage curve (aligned ,mid and unaligned) it can be shown that the net change in co-energy between the positions at zero amps is zero, as the working point of the magnet is the same. Therefore the average torque is zero,

which makes sense as the motor would be able to rotate by itself. However if the change in angular position was smaller, then the co-energy method could be used to calculate instantaneous torque. For the magnet it can be shown that a peak torque is produced at a rotor angle of 0.9°. This can be related to the back EMF / flux linkage plot of the magnet generated from FE analysis in figure 5.4. To find the change in co-energy, the flux and branch MMF value for 0.9° rotor position at zero amps is required. Also the values for the aligned 0° position are required.

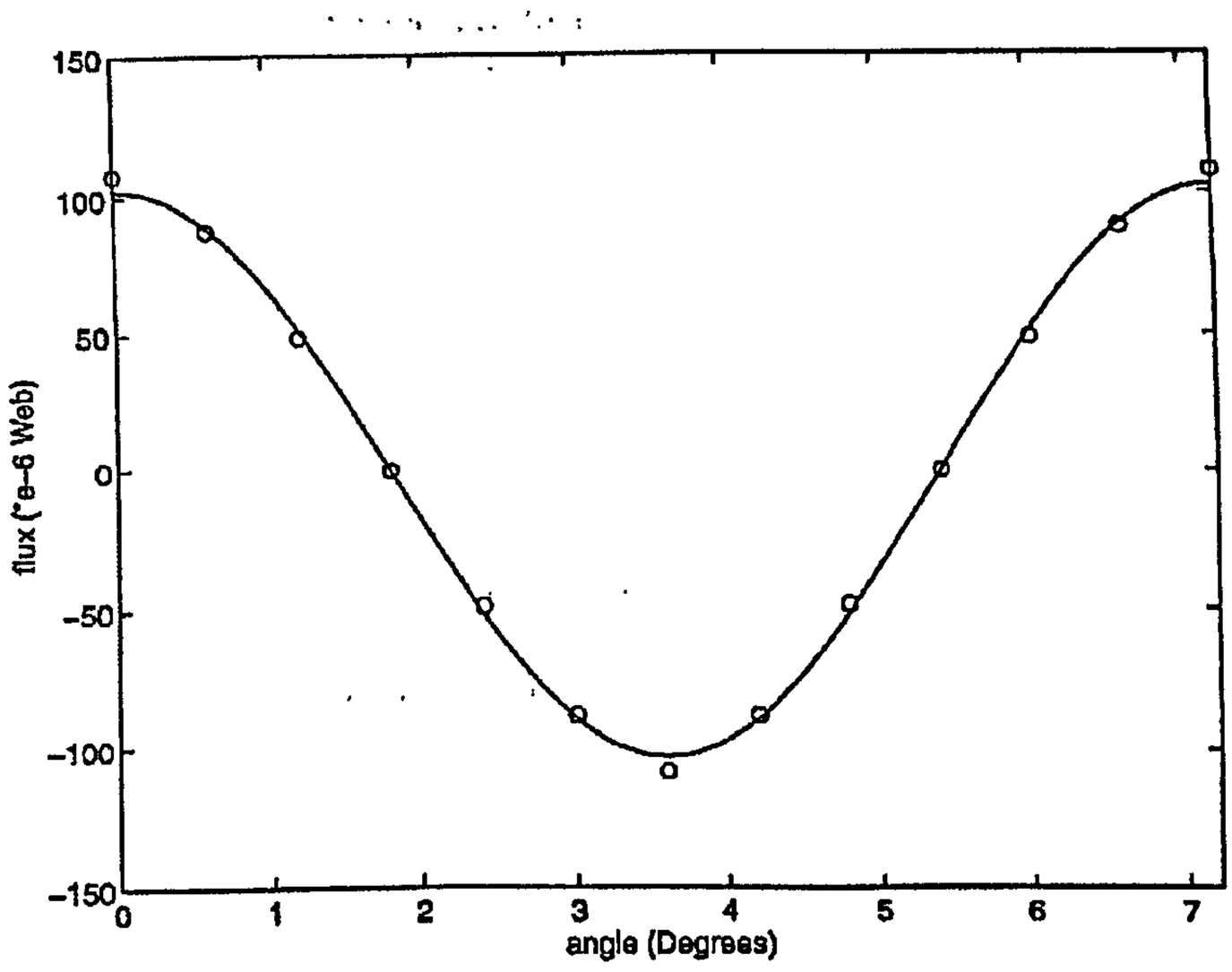


FIGURE 5.4 - Magnet Flux Linkage Plot

The flux value required to calculate the torque, for the 0.9° position, can be obtained from the single phase circuit from chapter 4, figure 4.21. In this circuit the flux can be calculated for the 0° (aligned), 1.8° (mid), and 3.6°(un-aligned) positions. To calculate the flux at 0.9° from the aligned position, the circuit could be solved by generating flux versus MMF relation curves for the 0.9° and 2.7° positions. An easier method is to take advantage of the fact that the flux of a complete pole can be very well approximated by a cosine function. Figure 5.4 shows the points calculated from FEM analysis, and a cosine

function plotted alongside. The relationship can be fitted by second order Fourier approximation such as equation 5.14.

$$y = A_0 + A_1 \cos(\omega_0 x) + A_2 \cos(2\omega_0 x), \qquad (5.14)$$

where ω_0 is the period conversion.

For the angle of 0.9°, the second term of equation 5.14 is zero therefore the flux can be calculated by ;

$$\phi = A_0 + A_1 \cos(\omega_0 \theta), \qquad (5.15)$$

where ω_0 is equal to $\pi/3.6$.

To fit the relationship only the aligned, mid and un-aligned flux need to be obtained. The value of branch MMF would be assumed the same as that of an aligned rotor position as the change in working point of the magnet is very small.

Adapting the co-energy method described before, the change in the co-energy will have flux on the Y-axis and the branch MMF on the X-axis as in figure 5.5. The area for 0.9° rotor position is subtracted from the 0° rotor position co-energy area. In this circumstance the 0.9° rotor movement is $1/400^{th}$ of a revolution. It must be remembered though that the detent torque is a 4^{th} harmonic component of overall torque, i.e. it occurs 4 times in one complete rotor movement (aligned to aligned, figure 5.6). Therefore we need to average the value obtained for one complete revolution by dividing it by four. This relationship is shown in figure 4.31 in comparison to a single phase scheme. The detent torque can therefore be calculated by an adaptation of equation 5.13.

$$T = \frac{\Delta W_c \times (\text{number of steps per rev})}{4 \times 2\pi} = \frac{\Delta W_c \times (360°/0.9°)}{4 \times 2\pi} = \frac{\Delta W_c \times 50}{\pi}. \quad (5.16)$$

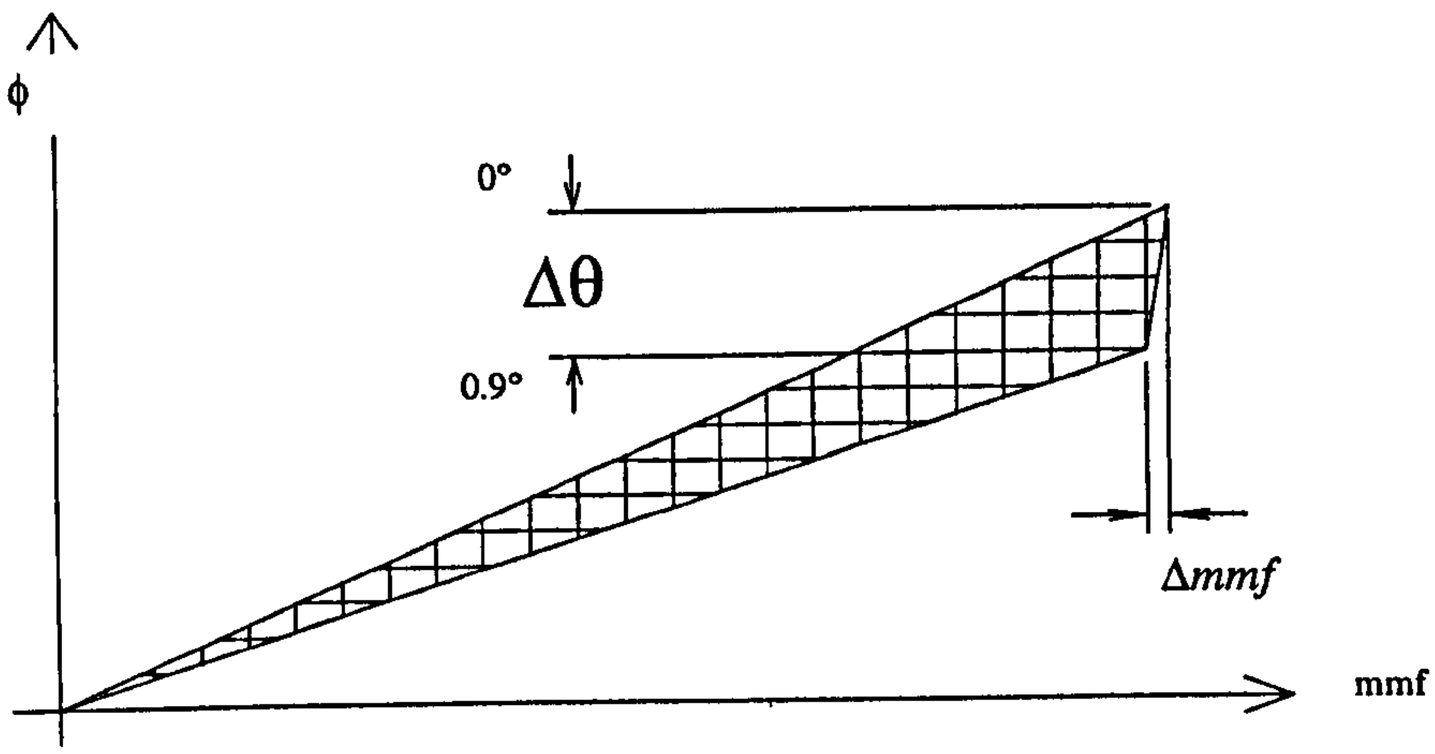


FIGURE 5.5 - Co-Energy of Detent Torque

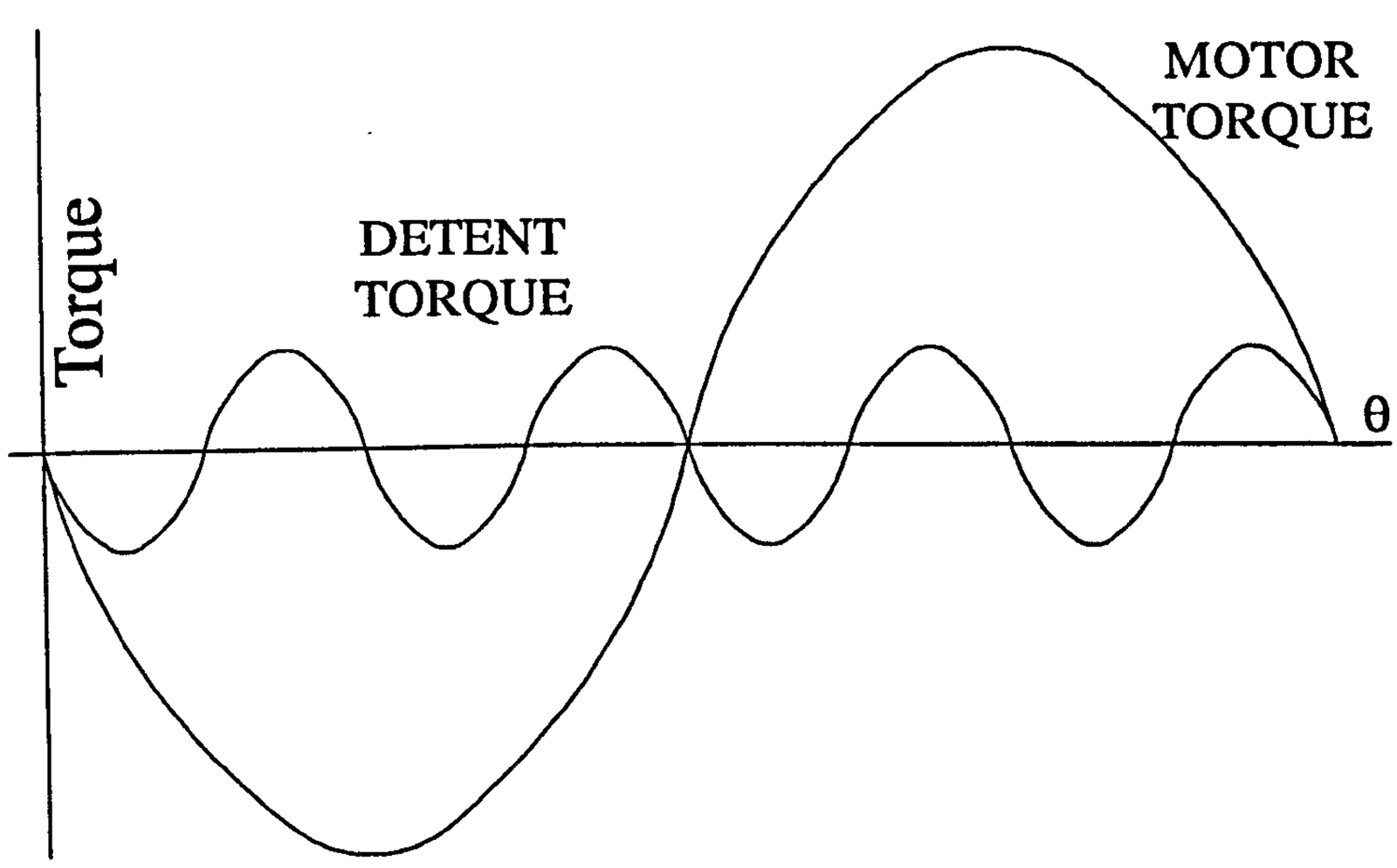


FIGURE 5.6 - Detent Torque shown as 4th Harmonic of Single Phase Motor Torque

Though the detent torque is in phase with the single phase excitation, it will be out of phase with two phase excitation. This phase can be shifted by altering the teeth width of the stator and rotor, in an attempt to move the general maximum phase torque to be in line with the detent torque.

5.3 Experimental and Three Dimensional Finite Element Results

In order to assess the accuracy of the computational methods a particular phase flux-linkage loop was obtained by the experimental method described in chapter 2. The Stebon 1101 motor was modelled by the 3-D FE methods described in chapter 3, and its flux-linkage curve obtained. The comparison is shown in figure 5.7.

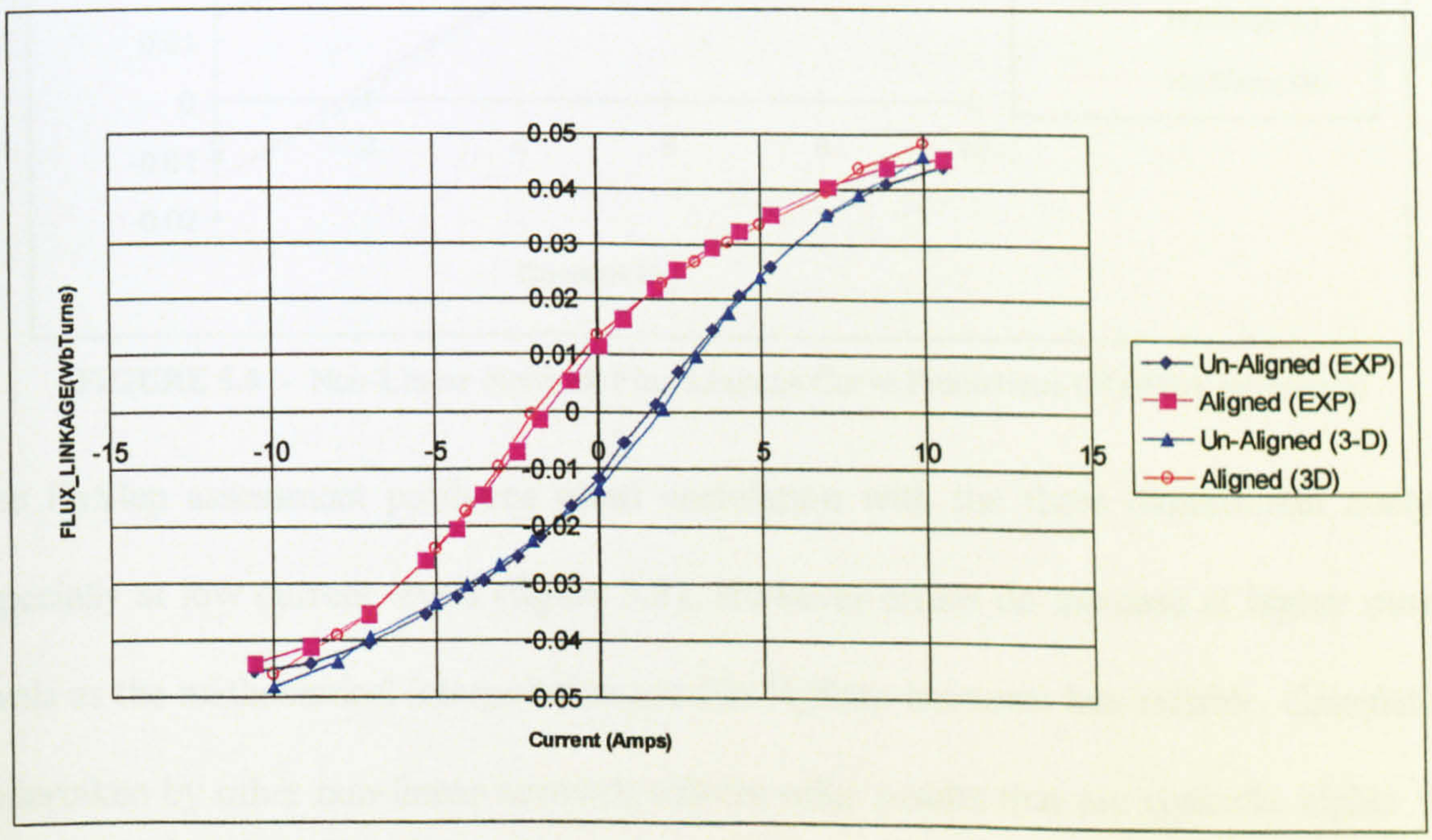


FIGURE 5.7 - Experimental Flux Linkage Curve with 3-D FEA Prediction

The results show good correlation. The three dimensional FEA tends to yield a higher absolute value of flux-linkage including the magnet (zero current) offset. Above 8 Amps the error does increase as saturation occurs. However the overall results are considered to

be within acceptable tolerances for computational, measurement and manufacturing discrepancies. The performance of Alnico magnet materials is highly dependent on the amount of cobalt present. The amount of cobalt present may vary from sample to sample.

5.3.1 HyStep Algebraic Predictions

The 1101 motor had it flux-linkage curves computed using HyStep's internal flux path generator and back iron correction methods.

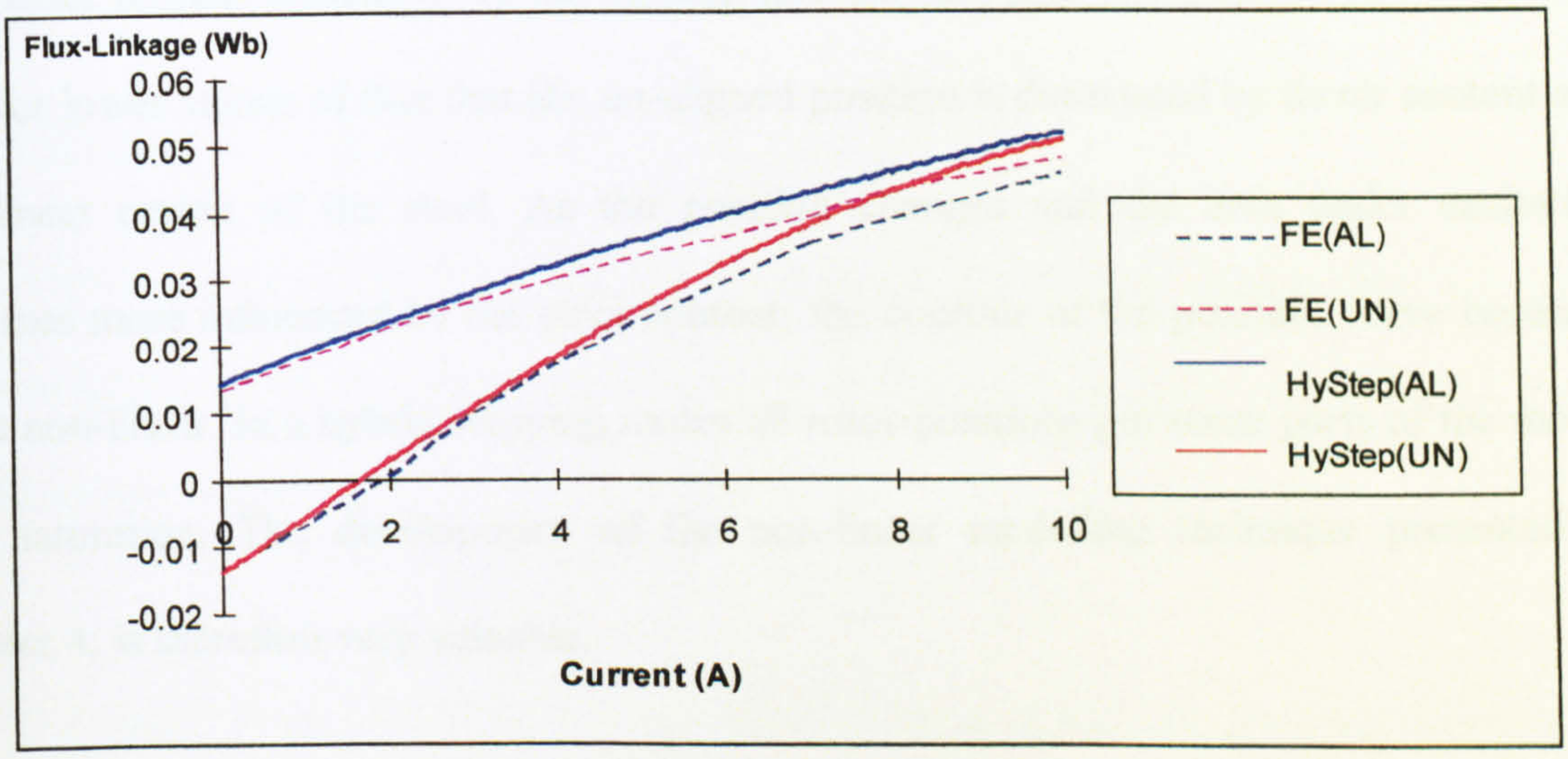


FIGURE 5.8 - Non-Linear Network Flux Linkage Curve Predictions (3D-FEA Reference)

The HyStep assessment produces good correlation with the three dimensional analysis, especially at low current levels (figure 5.8). However errors do increase at higher current levels as the mathematical interpolation used in HyStep becomes less reliable. Calculations undertaken by other non-linear network solvers offer results that are typically higher than the 3-D analysis, mainly due to the absence of back-iron effects and the use of linear interpolation between data points. HyStep offers good prediction when used directly with data obtained from a finite element package. Using HyStep solely or collectively with 2-

dimensional data is very attractive when considering the reduced computational time and significantly lower effort involved.

On closer analysis of the two dimensional data that was generated for use in the network solvers it can be seen that the reluctance of each position is relatively constant at low values of flux (figure 5.9) and all converge at higher values of flux as the iron portion saturates and becomes like air. This can be also seen from the more standard representation of flux against MMF for the tooth profiles in figure 5.10, which is generated from the flux/ reluctance profiles calculated by the HyStep internal flux path calculations. It can be seen that for lower values of flux that the un-aligned position is dominated by its air content and the linear region of the steel. As the position changes and the area under excitation becomes more influenced by the steel content, the contour of the position curve becomes more non-linear. In a hybrid stepping motor all rotor positions put some parts of the motor into saturation. The development of the non-linear modelling technique presented in chapter 4, is therefore very valuable.

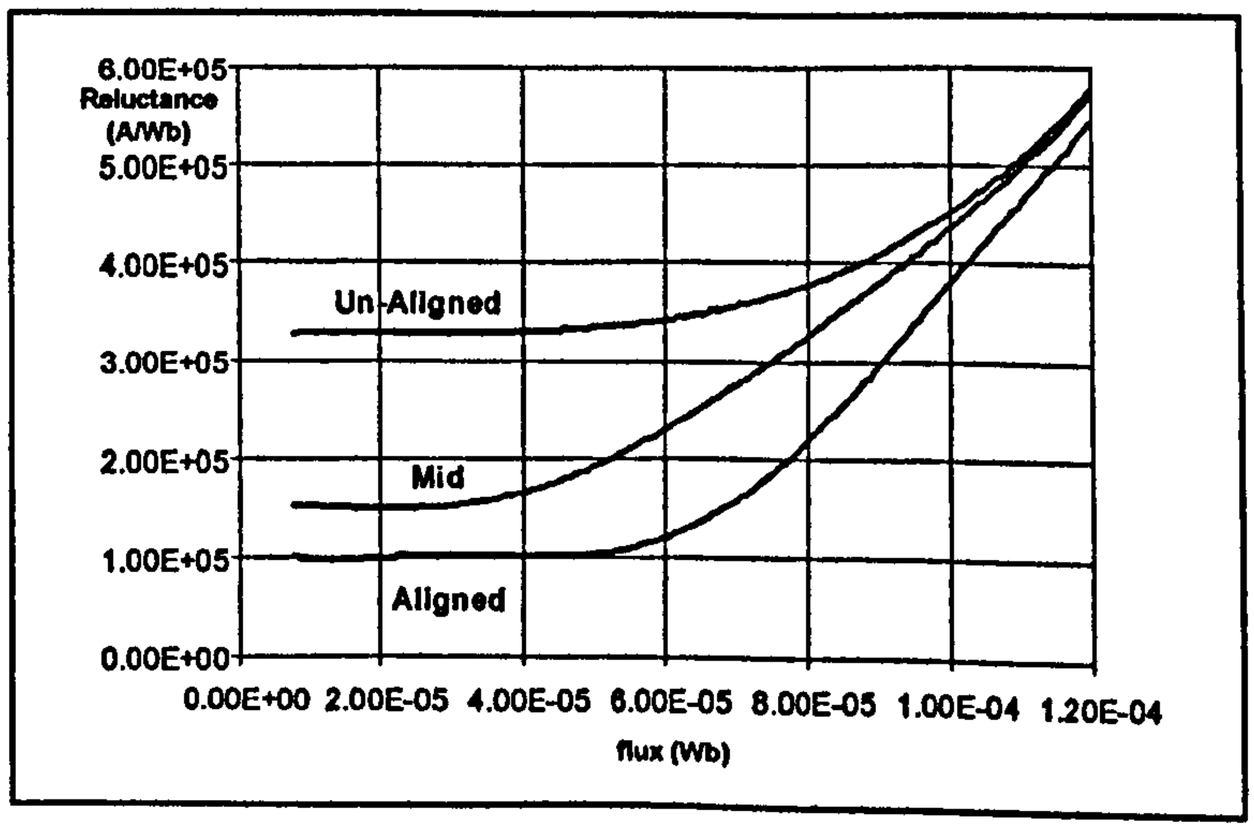


FIGURE 5.9 - Tooth Profile Reluctance

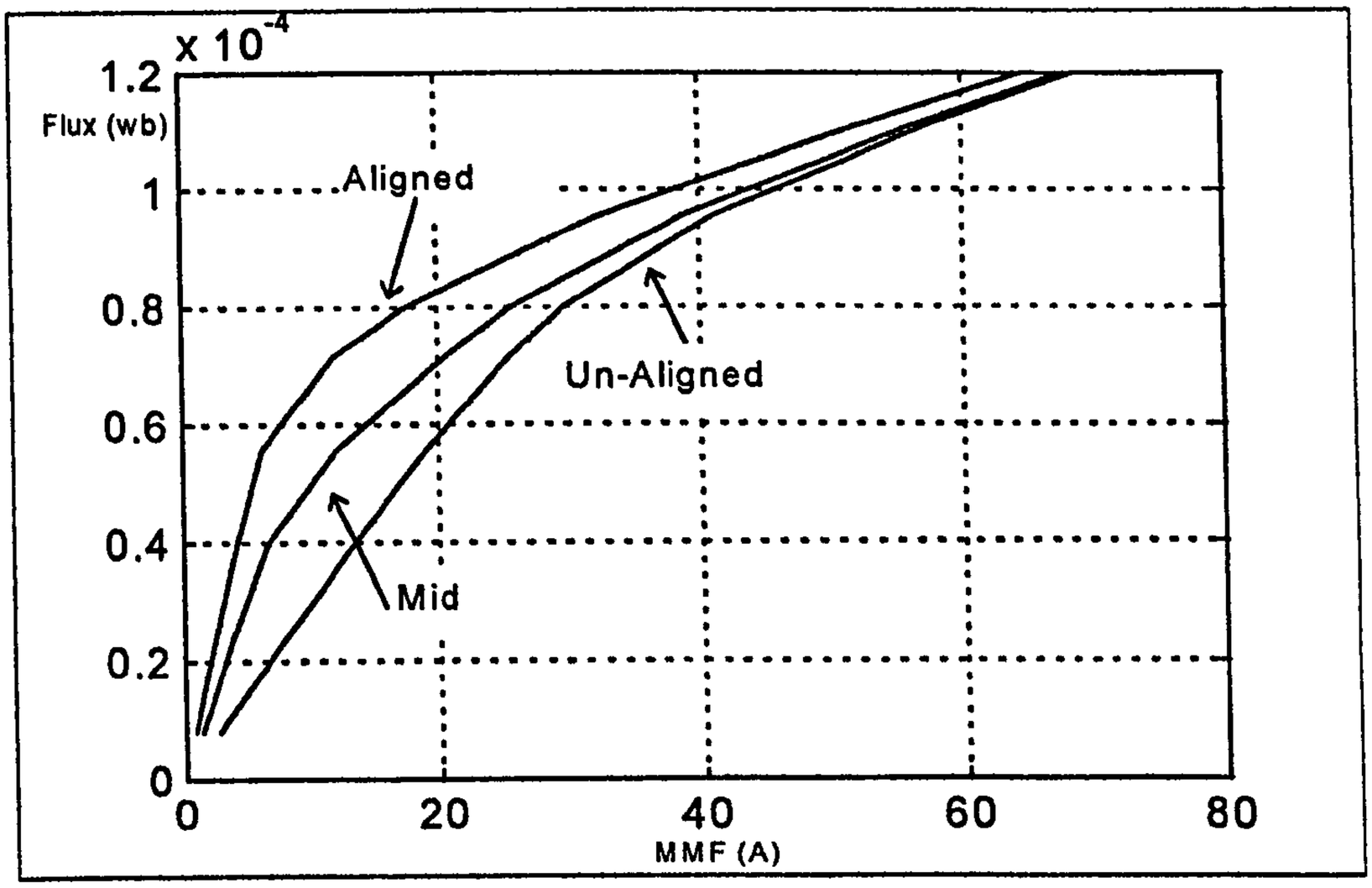


FIGURE 5.10 - Tooth Profile Flux/ MMF Relationships for Tooth Area

5.4 Static Torque Test

A DC current was applied directly to the windings of the Stebon 1101 motor. A torque was then applied, and the displacement from an equilibrium position was recorded. This test is outlined in chapter 2.

The static torque profile produced by the Hybrid stepping motor at a particular angle can also be obtained from 3-D FEA by computing the integral of the Maxwell Stress tensor over the surface of the inter stator/ rotor air-gap. The 3-D FEA static torque prediction and experimental data are shown in figures 5.11 to 5.12. The peak static torque predictions are displayed in figure 5.13.

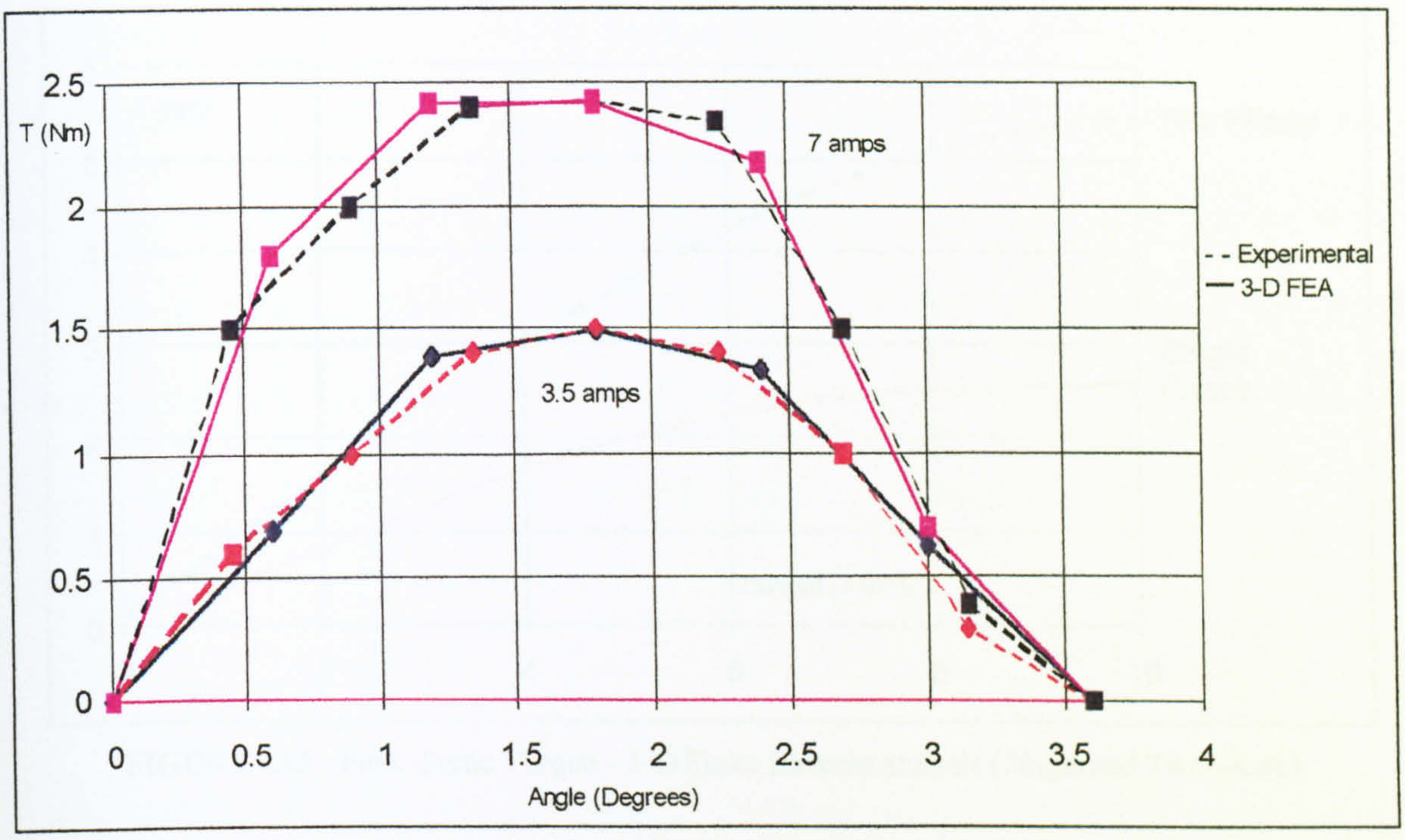


FIGURE 5.11 - Static Torque, Experimental and 3-D Finite Element analysis (single phase) 0° refers to the Aligned Position

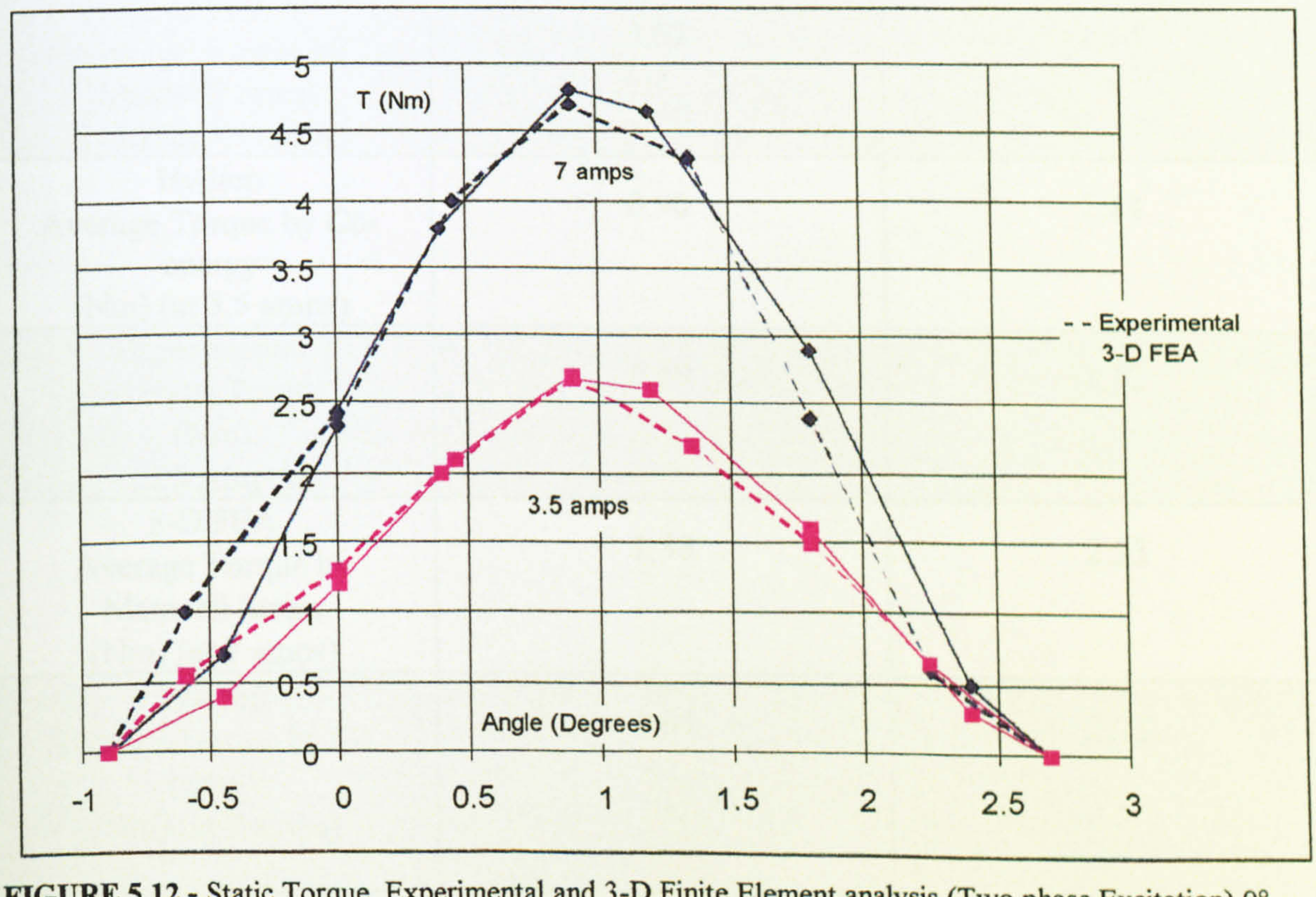


FIGURE 5.12 - Static Torque, Experimental and 3-D Finite Element analysis (Two phase Excitation) 0° refers to the Aligned Position

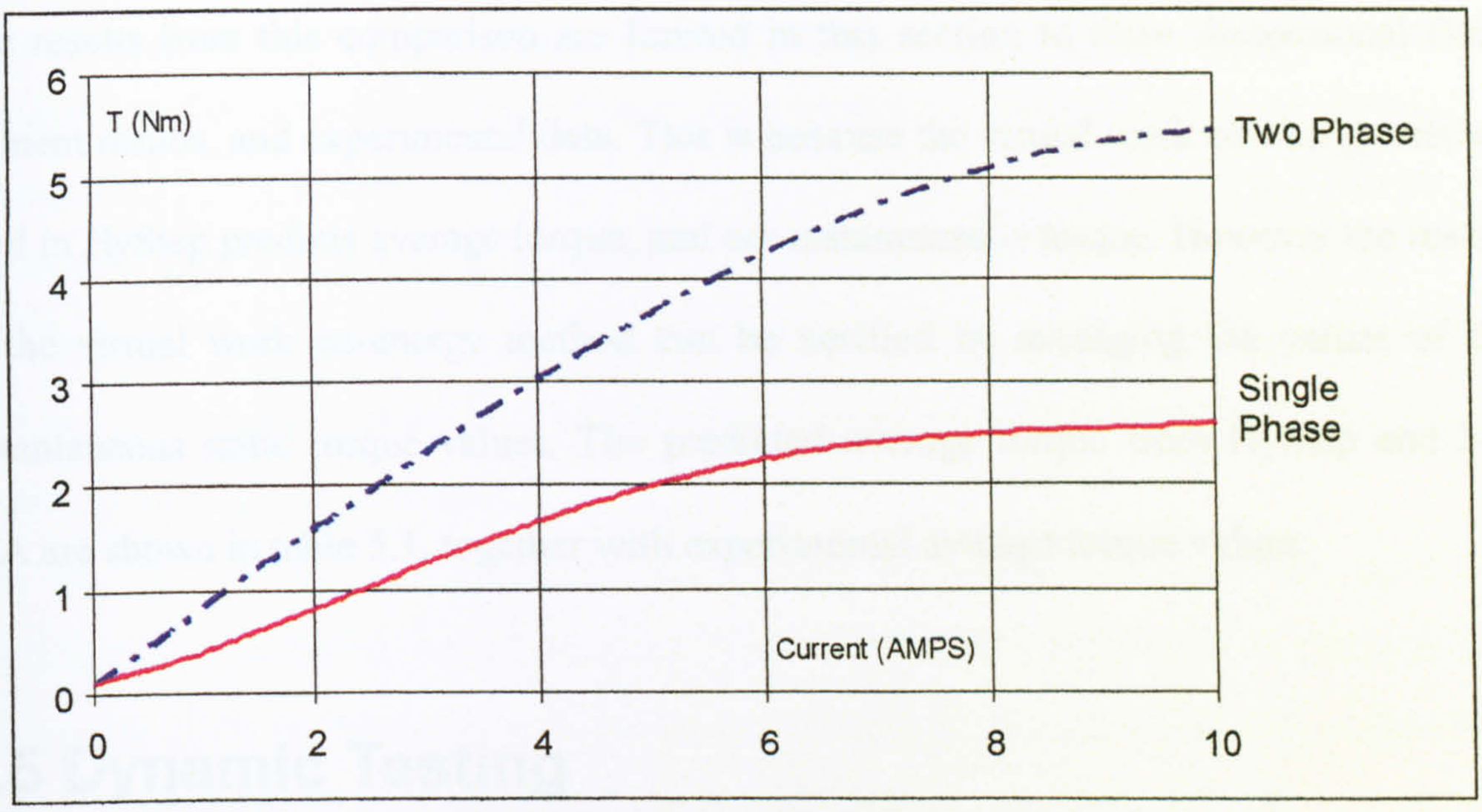


FIGURE 5.13 - Peak Static Torque - 3-D Finite Element analysis (Single and Two phase)

	Single Phase Excitation	Two Phase Excitation
3-D FEA Average Torque by Maxwell Stress (Nm) (at 3.5 amps)	0.92	1.40
HyStep Average Torque by Co-energy (Nm) (at 3.5 amps)	0.96	1.44
Experimental Average Torque (Nm) (3.5 Amps)	0.89	1.32
3-D FEA Average Torque by Maxwell Stress (Nm) (at 7 amps)	1.58	2.53
HyStep Average Torque by Co-energy (Nm) (at 7 amps)	1.64	2.71
Experimental Average Torque (Nm) (at 7 Amps)	1.54	2.41

Table 5.1 - Averaged Torque versus Virtual Work Predictions

The results from this comparison are limited in this section to three dimensional finite element results, and experimental data. This is because the virtual work co-energy method used in HyStep predicts average torque, and not instantaneous torque. However the results of the virtual work co-energy method can be verified by averaging the values of the instantaneous static torque values. The predicted average torque from HyStep and 3-D FEA are shown in table 5.1, together with experimental average torque values.

5.5 Dynamic Testing

Three dimensional finite element analysis produces static results but cannot directly predict dynamic performance, a major requirement for an industrial applications motor designer. The situation is further complicated by the complex variety of stepping motor drives and controllers, each with very different dynamic characteristics. Apart from designs of machines that are specifically for microstepping and mini-stepping, a motor that performs well under full stepping will generally have similar performance characteristics under half stepping, and other increased current resolutions, neglecting situations that produce motor resonance and inductance which will always be most pronounced in full stepping..

5.5.1 The Distribution of Energy

Figure 5.14 shows how energy is distributed within the motor-drive system and how electrical energy is converted into mechanical work and provides a model applicable to any electric motor. The process for this highly controlled system is thus:

- Assuming the rotor is restrained, when a voltage is applied to a phase winding the current builds up over a finite period determined by the inductance of the coil. A magnetic flux associated with this current also builds up, acting as a reservoir for potential energy by storing it in the field of the air-gap.
- If the rotor is now released while the current remains constant, the potential energy stored in the magnetic field is released as mechanical work in moving the rotor to its desired equilibrium position.
- If the rotor remains restrained, but the current is removed, the magnetic field will collapse and the energy stored therein will be released back into the electrical circuit.
- The sum of the energy recovered and the mechanical energy converted will be less than the energy put in by an amount equivalent to the losses.

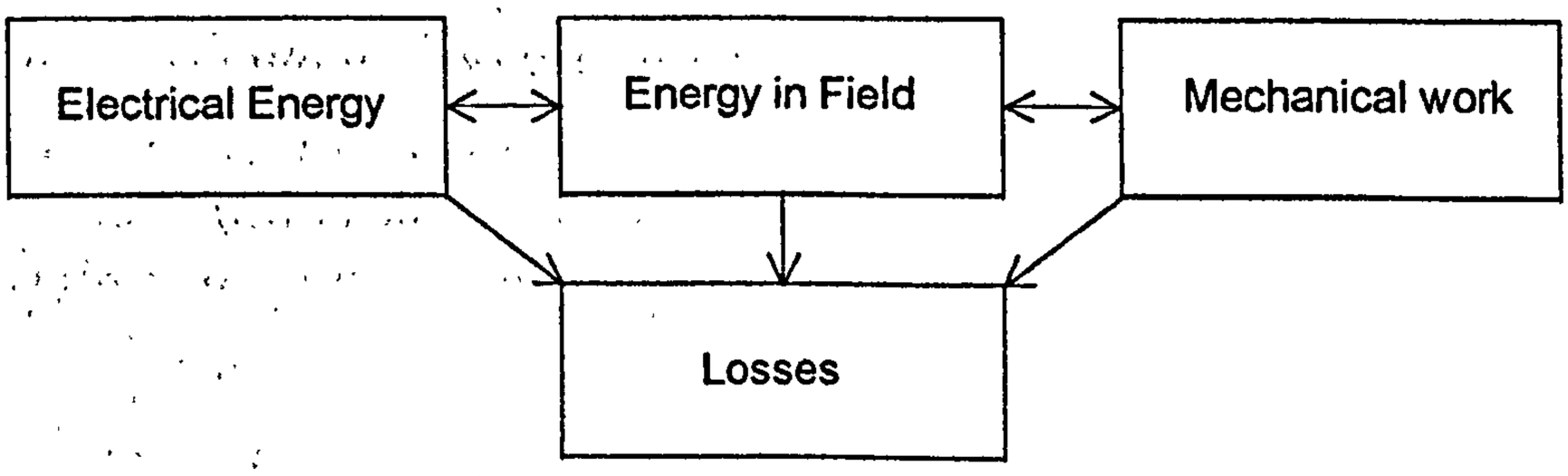


FIGURE 5.14 - The Distribution of Energy Within the Hybrid Step Motor

The losses indicated in figure 5.14 can be separated into three main categories as shown in table 5.2.

Electrical	Electromagnetic	Mechanical
Copper losses (i^2R)	Eddy currents	Friction
Switching losses	Hysteresis	Windage

Table 5.2 - Losses in the Hybrid Stepping Motor

The conversion of electrical energy to mechanical work can be summarised by the general rule:

$$\textbf{Electrical energy in} = \textbf{Mechanical work out} + \textbf{Increase in magnetic energy} + \textbf{Combined losses} \quad (5.17)$$

The balance of this equation becomes far more important under dynamic conditions than static, as most losses increase with speed, with the main exception being copper losses. This is because the winding inductance prevents the build-up of current and hence the current delivered to the windings decreases with speed. It is evident that a purely analytical approach to the calculation of the motor dynamic torque will, at best, be approximate and complex due to the interdependence of variables of the motor in tandem with a drive.

Over each motor operation cycle each of the elements in equation 5.1 apart from the losses, may take on positive and negative values, as energy is interchanged between the supply, the field and the load.

5.6 Mathematical Model for a Phase Winding of a Two Phase Hybrid Stepping Motor

It would be useful and convenient if a dynamic model of the motor could predict performance based on information already calculated by HyStep. Though the hybrid stepping motor is a dynamic non-linear system, further assumptions and simplifications about its operation under dynamic conditions can be ascertained from three dimensional

finite element modelling. In the following sections a mathematical model is derived for the phase windings and simplified from observations from 3-D FEM.

Hysteresis and eddy current losses are neglected in the following modelling. This can be considered as a not too unrealistic assumption, particularly when the motor is run at low speeds. However these effects will reduce flux-linkage, and all results need to viewed in the light of iron losses.

A model of a phase winding of the motor can be achieved by applying Kirchoff's voltage law and Faraday's law of electromagnetism.

$$V = iR + \frac{d\lambda}{dt}, \qquad (5.18)$$

where V is the voltage across the phase winding, and R is the phase winding resistance.

The iR term is the voltage drop due to the winding resistance, the second term of the equation is the induced EMF due to the rate of change of flux linkage with time. For Faraday's rule to be met, the terms on the right hand side must equal the terminal voltage of the phase.

The flux-linkage term can be obtained by either 3-D FEA or HyStep. This has been shown to be a function of magnetic geometry, and the magnetomotive forces developed by the current in the phase windings and those generated by the permanent magnet. In full, the flux linkage of phase 1 is a function of the current in both phases, the rotor position, and the magnet MMF,

$$\lambda_1 = \lambda_1\left(i_1,\ i_2,\ \theta,\ MMF_{mag}\right), \qquad (5.19)$$

where θ is the rotor position.

Thus for phase 1, equation 5.18, is rewritten as

$$V_1 = i_1 R_1 + \frac{d\lambda_1\left(i_1,\, i_2,\, \theta,\, MMF_{mag}\right)}{dt} \quad . \qquad (5.20)$$

A similar expression for phase winding 2 holds true. By use of the chain rule, equation 5.20, can be expanded to give equation 5.21.

$$V_1 = i_1 R_1 + \frac{d\lambda_1}{di_1}\frac{di_1}{dt} + \frac{d\lambda_1}{di_2}\frac{di_2}{dt} + \frac{d\lambda_1}{d\theta}\frac{d\theta}{dt} + \frac{d\lambda_1}{dMMF_{mag}}\frac{dMMF_{mag}}{dt}, \qquad (5.21)$$

The first term on the right is still the resistive drop, because of the current flowing in the windings. The second term is the induced voltage due to the rate of change of current, i_1, in phase 1, with respect to time. It is named the self inductive voltage drop. The third term is called the mutual inductive voltage drop, which arises when voltage is induced in phase 1, because of a change in current flowing in phase 2. The next term arises because of the voltage induced when the rotor changes position with respect to time, it is rotor speed dependent. This is commonly known as the back EMF. The final term is the voltage induced because of the rate of change in the MMF developed by the permanent magnet with respect to time. A similar equation can be formed for phase 2.

The axially magnetised permanent magnet found in the rotor of the hybrid stepping motor, is subject to demagnetising fields produced when current flows in the stator phase windings. Within the normal operating characteristics of the motor, the magnetic flux generated by the magnet may be considered approximately constant once magnetised. This indicates that the magnet MMF will also remain approximately constant because of the

stable load point on the magnetisation curve. If the current in the windings does not exceed its rating, and the motor is not driven high into saturation than this change in MMF may be less than 10%. Table 5.3 shows results obtained from 3-D FEA results with the magnet operating under a full range of currents. The rating current of the motor is 7 Amps.

Current (Amps)	magnet MMF (Ampere turns)	magnet flux density (Weber e-6)
0	255	823
2	254	825
3	255	823
4	260	822
5	264	819
7	276	814
8	283	811
10	298	804
12	313	799
14	328	793

TABLE 5.3 - Magnet Flux and MMF Values versus Winding Current in One Phase Taken at the Aligned Position

For simplicity it will be assumed that the magnet MMF change with time is zero, on the understanding that this may be increasingly inaccurate at higher winding currents.

$$\frac{dMMF_{mag}}{dt} = 0 . \qquad (5.22)$$

This term is therefore eliminated from the equation 5.21.

The mutual flux-linkage between the phases is governed by the $\frac{d\lambda_1}{di_2}\frac{di_2}{dt}$ term in equation 5.21. Three dimensional finite analysis agrees with assumptions [63,64,65] that the flux linking a particular phase is dominated by the current flowing in that phase and that induced by the permanent magnet. This implies that the effect of current in another phase on the

phase winding under examination is negligible. The next figure (5.15) shows the increase in flux (and flux density) in a stator pole as the current rises in its phase winding. Also included in the figure is the rise of flux (and flux density) in the second phase winding which does not have any current excitation. The starting point at zero current shows the flux linkage due to magnet only. Gradually the current in one winding is allowed to rise and the flux-linkage may be calculated for both phase windings by multiplying the flux by the phase turns. The flux-linkage in the second un-excited phase is nearly constant especially under rated conditions (7 Amps). Therefore it can be assumed that the mutual flux-linkage is negligible.

$$\frac{d\lambda_1}{di_2} = 0 . \qquad (5.23)$$

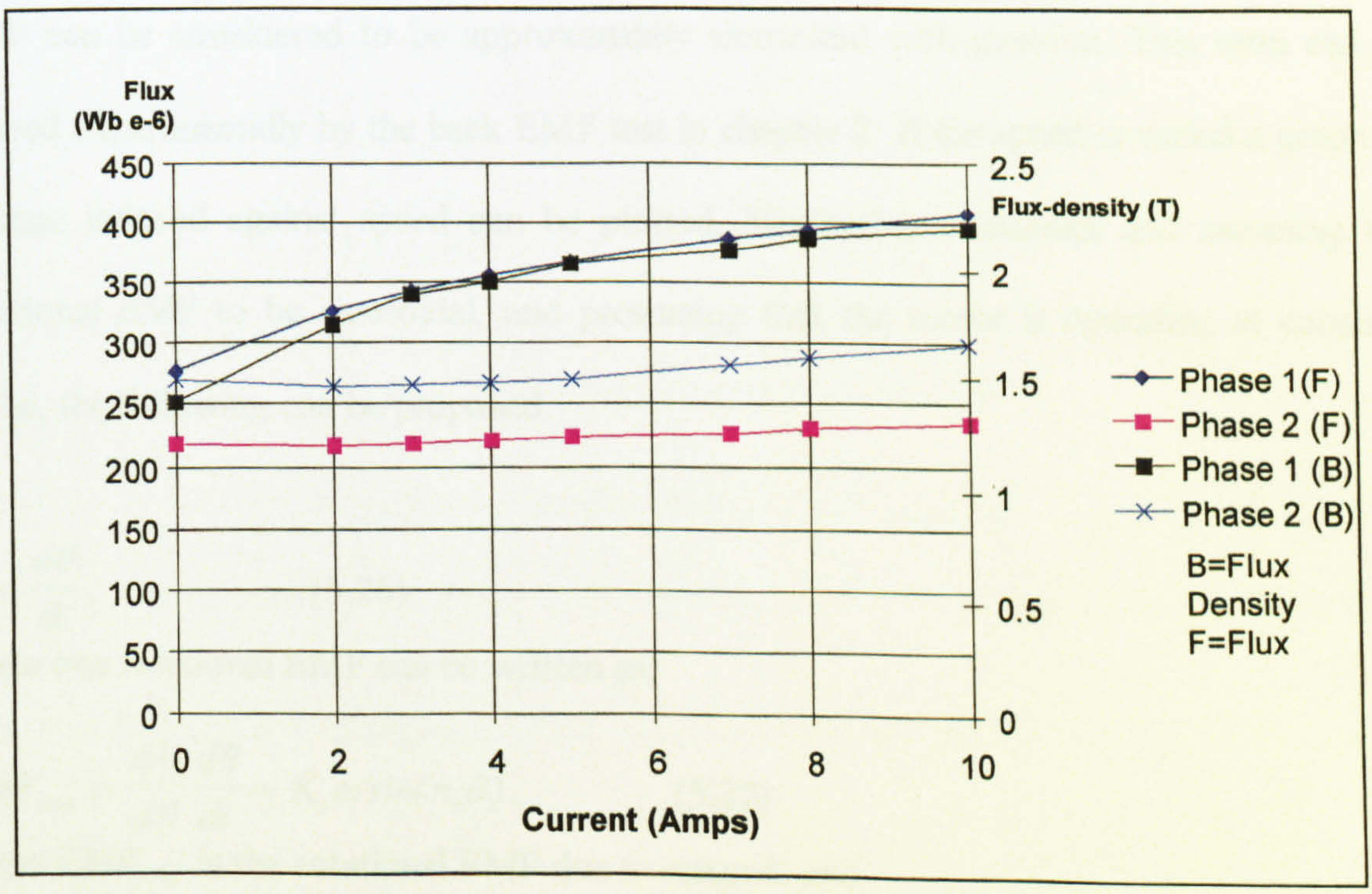


FIGURE 5.16 - Flux Density and Flux in Phase 1 and 2 as the Current in Phase 1 is Varied

The self inductance of the phase winding can be seen as the gradient of the flux-linkage curve, shown in figure 5.16.

$$\frac{d\lambda_1}{di_1} = l, \qquad (5.24)$$

where l represents instantaneous self inductance.

For the linear part of the flux-linkage curve this remains constant, even over a change in position. Rewriting equation 5.15, by eliminating the appropriate terms, the equation of voltage (with respect to previously discussed conditions) may be considered as;

$$V_1 = i_1 R_1 + L\frac{di_1}{dt} + \frac{d\lambda_1}{d\theta}\frac{d\theta}{dt}. \qquad (5.25)$$

The remaining position dependent term is the back EMF. Generally the open circuit back EMF can be considered to be approximately sinusoidal with position. This term can be viewed experimentally by the back EMF test in chapter 2. If the speed is varied a graph of voltage induced against speed can be plotted. Neglecting harmonics and assuming the rotational EMF to be sinusoidal, and presuming that the motor is operating at constant speed, the following can be proposed.

$$\omega = \frac{d\theta}{dt}, \qquad (5.26)$$

Phase one rotational EMF can be written as,

$$EMF_{mag} = \frac{d\lambda_1}{d\theta}\frac{d\theta}{dt} = K_e \omega \sin(n_r\theta), \qquad (5.27)$$

where EMF_{mag} is the rotational EMF due to magnet, and

K_e is the rotational EMF constant.

The other phase on the motor is shifted by an electrical angle of $\pi/2$ radians. This causes the back EMF to be shifted by the same angle. Therefore, for the other phase the equation would be;

$$EMF_{mag2} = \frac{d\lambda_2}{d\theta}\frac{d\theta}{dt} = K_e \omega cos(n_r \theta). \qquad (5.28)$$

The rotational EMF constant can be obtained by taking the gradient of the back EMF versus speed curve (figure 5.17).

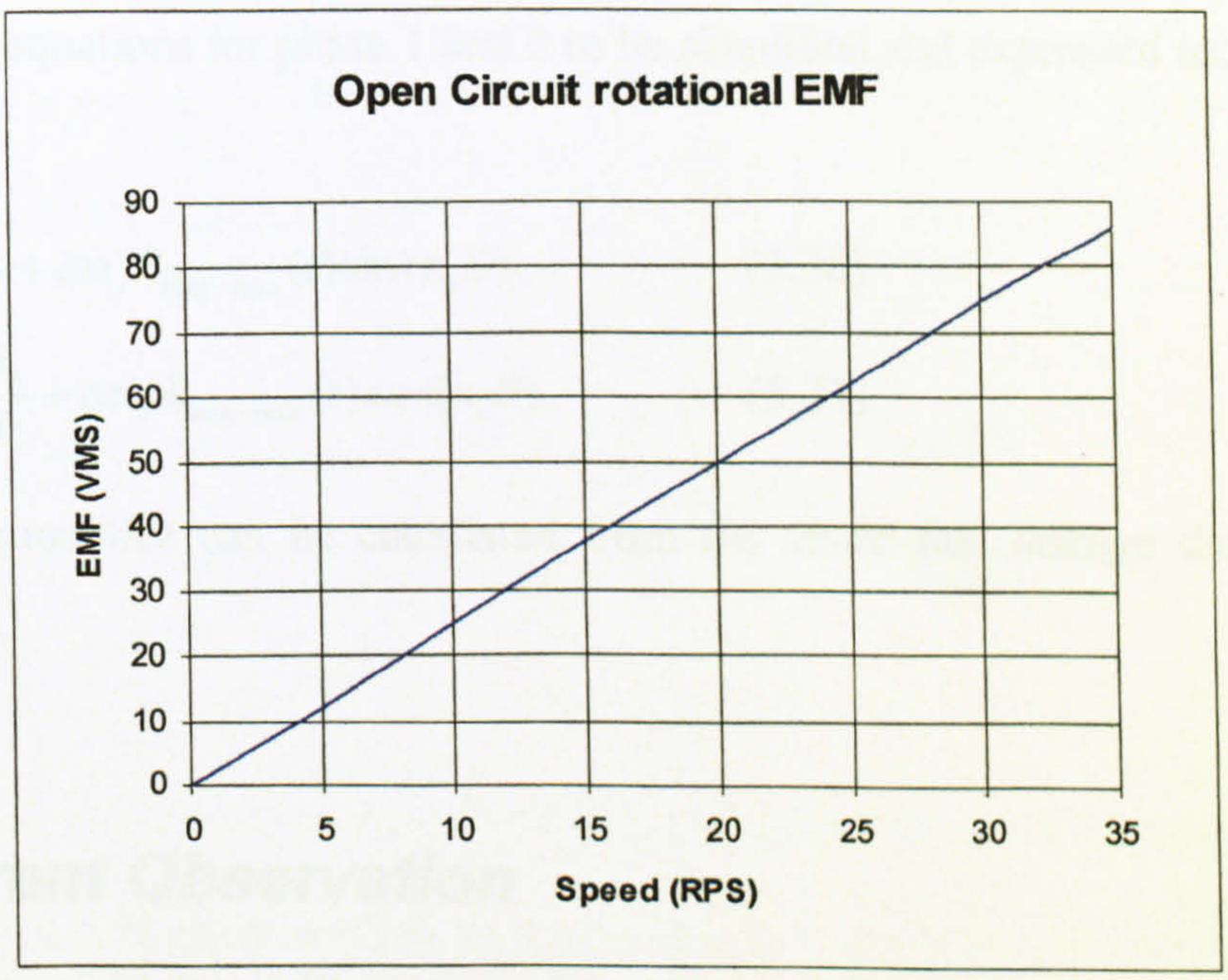


FIGURE 5.17 - Open Circuit Rotational RMS EMF

However as the data is obtained experimentally, it is not available in HyStep. HyStep can calculate flux-linkage versus position. If the sinusoidal relationship of the magnet's flux-linkage relationship is taken as $\frac{d\lambda}{d\theta}$ then we can obtain the expression for the back EMF in phase 1,

$$EMF_{mag1} = \frac{d\lambda_1}{d\theta}\frac{d\theta}{dt} = n_r \lambda_{mag-max}(t)\omega \sin(n_r\theta). \qquad (5.28)$$

where $\lambda_{mag-max}$ is the maximum flux-linkage in the phase due to the magnet at zero current. This will hold unless the motor is pushed high into saturation. The second phase has a similar equation;

$$EMF_{mag2} = \frac{d\lambda_2}{d\theta}\frac{d\theta}{dt} = n_r \lambda_{mag-max}(t)\omega \cos(n_r\theta). \qquad (5.29)$$

This allows the equations for phase 1 and 2 to be simplified and expressed as;

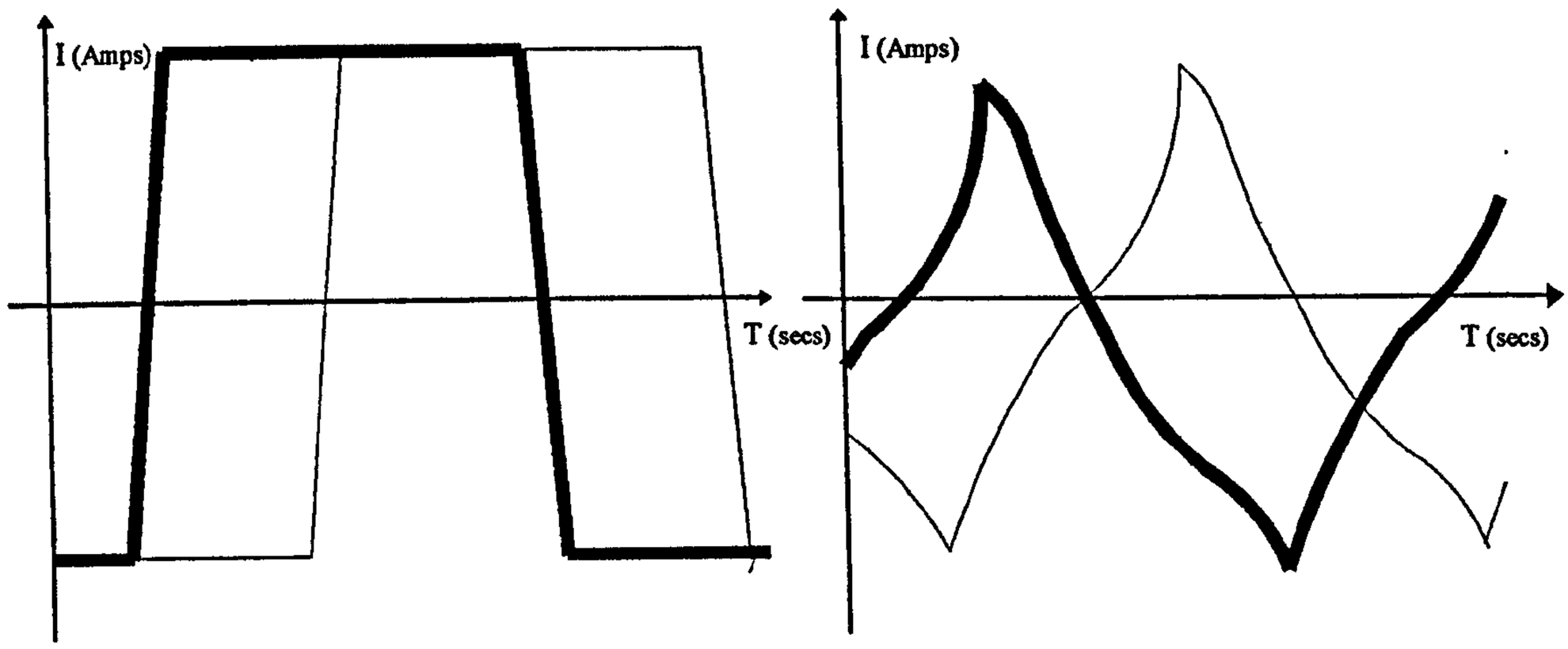
$$V_1 = i_1 R_1 + L\frac{di_1}{dt} + \omega n_r \lambda_{mag-max}(t)\sin(n_r\theta), \qquad (5.30)$$

$$V_2 = i_2 R_2 + L\frac{di_2}{dt} + \omega n_r \lambda_{mag-max}(t)\cos(n_r\theta). \qquad (5.31)$$

All of theses quantities can be calculated from the static flux linkage data available in HyStep.

5.6.1 Current Observation

A key element to generating torque is the current available to produce flux in the motor whilst rotating. The current control schemes adopted all produce differing results. Full stepping produces harder step action than, say, half stepping. Microstepping gives, generally, a torque output closer to full stepping with a smoother operation. It also provides a better torque speed range. Concentrating on full stepping, the current at low speeds, where current has time to rise through the winding inductance, will appear square, figure 5.18a. However as speed increases the current begins to distort (figure 5.18b).



FIGURES - 5.18a/b low speed current profile/ high speed current profile

The current in the winding can be estimated by the phase winding equations 5.30 and 5.31. Rearranging the equations of the phase voltages to be in a form that allows the current to be calculated, gives the expressions;

$$\frac{di_1}{dt} = \frac{1}{L}\left[V_1(t) - i_1 R_1 - \omega n_r \lambda_{mag-mag}(t)\sin(n_r\theta)\right], \qquad (5.32)$$

$$\frac{di_2}{dt} = \frac{1}{L}\left[V_2(t) - i_2 R_2 - \omega n_r \lambda_{mag-mag}(t)\cos(n_r\theta)\right]. \qquad (5.33)$$

These differential equations can be solved by numerical techniques to predict the current wave form. These will allow HyStep to predict the torque by re-calculating the co-energy transfer, with the predicted current in the corresponding phase.

A third order Runge-Kutta method [66] was used to predict the current. This was felt to give a good compromise between accuracy and speed. The equation is of the form;

$$y_{i+1} = y_i + \left[\frac{1}{6}(k_1 + 4k_2 + k_3)\right]h, \qquad (5.34)$$

where

$$k_1 = f(x_i, y_i), \qquad (5.35)$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1\right), \qquad (5.36)$$

$$k_3 = f\left(x_i + h, y_i - hk_1 + 2hk_2\right), \qquad (5.37)$$

x_i, y_i are the initial conditions of x and y,

and h is the numerical step size of the period of integration.

To integrate equations 5.32 and 5.33 'x' and 'y' of equation 5.34 are time and current respectively. The period of integration will be over a step interval of the motor. The period can be calculated from the number of rotor steps per revolution. For a 200 step resolution (full stepping) the time period will be related to the speed in revolutions per second by;

$$period = \frac{steps}{200} \times rps, \qquad (5.38)$$

Where steps are the number of motor steps under observation, and rps are the revolutions per second, where rps is related to ω by

$$\omega = 2\pi \text{ rps.} \qquad (5.39)$$

'h' in equation 5.34 will be a value that is an order of magnitude of this period, i.e. a time step.

The maximum current is limited by a maximum current value, otherwise the current will continue to rise. In this method the drive is assumed to be using ideal switches and to exhibit no overshoot. The rotor position is assumed to in phase with the voltage and running at constant speed.

5.6.2 Verification of Current Waveforms

A series of tests was carried out to validate the dynamic current prediction method used by the HyStep program. The HyStep prediction assumes that the motor is working in a near ideal system. The predicted and experimental torque/ speed curve are obtained for a series connection, peak to peak current of 7 Amps. The current waveform characteristics are shown for speeds of 1 rps and 10 rps in figure 5.19 and 5.20 respectively, which allow the generation of the torque speed curve of figure 5.21.

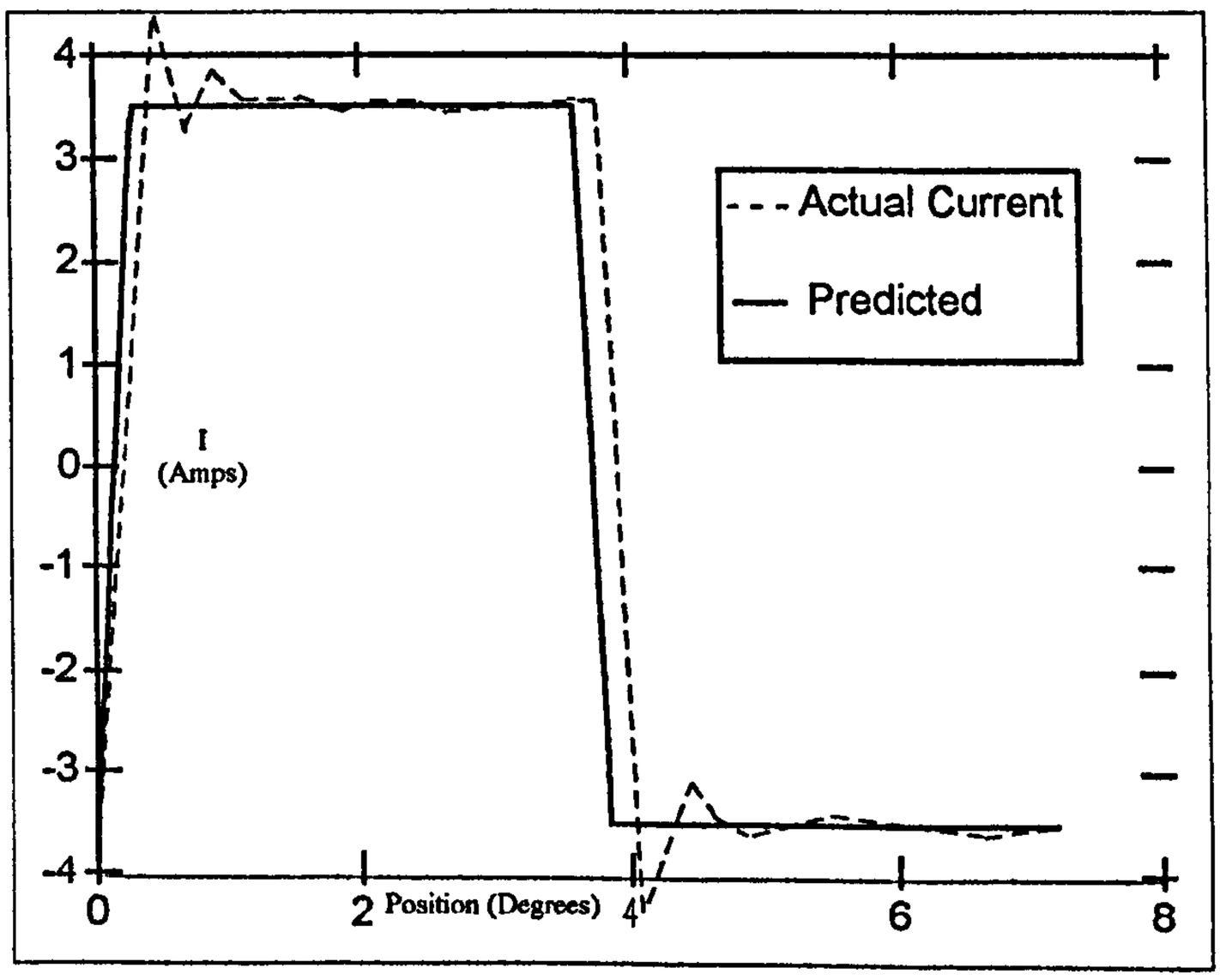


FIGURE 5.19 - Current Waveforms for 1 rps, 7 Amps Peak

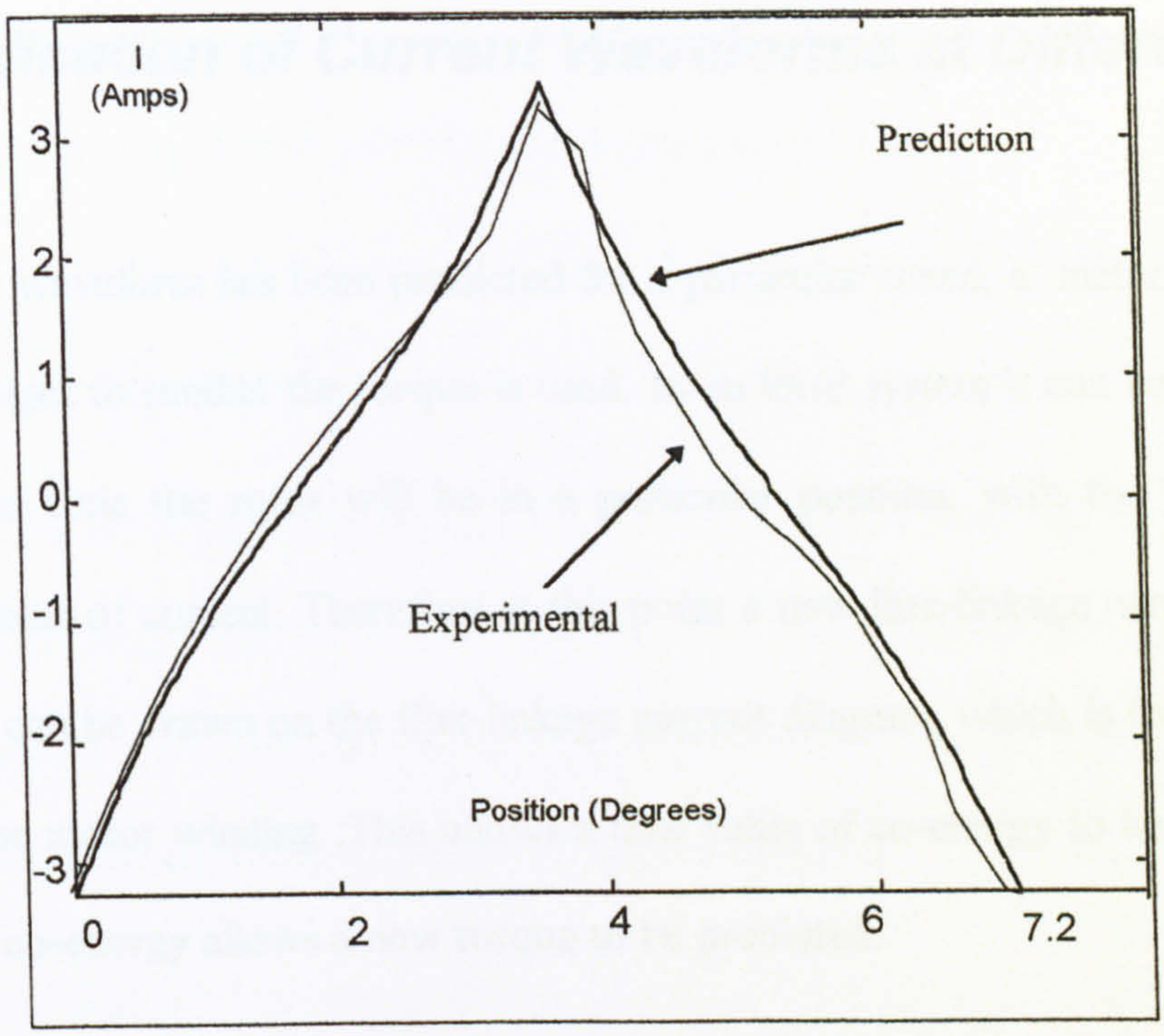


FIGURE 5.20 - Current Waveforms for 10 rps, 7 Amps

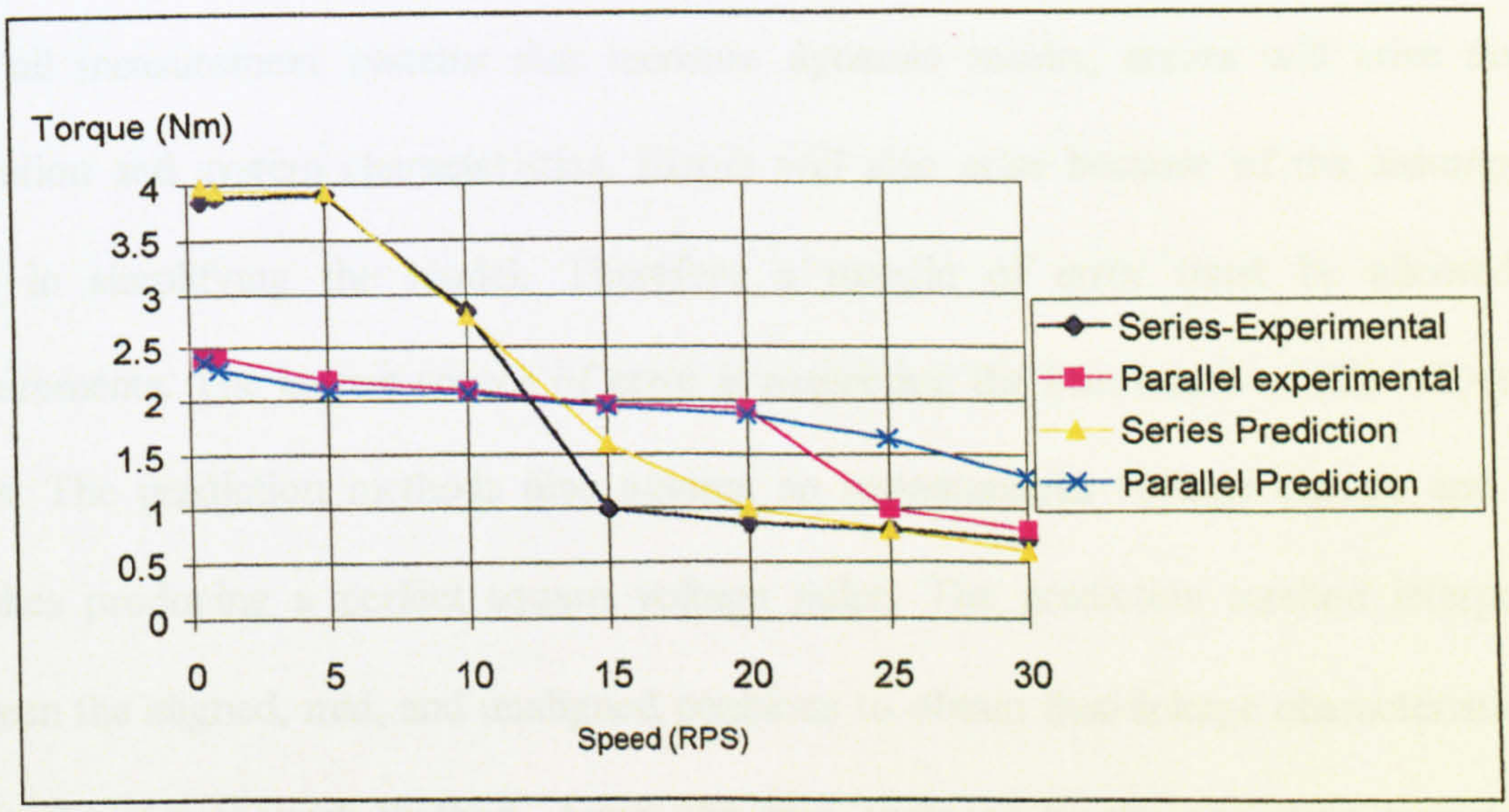


FIGURE 5.21 - Torque/ Speed Curve (Full Stepping - 7 Amps Peak)

5.6.3 Verification of Current Waveforms at Different Speeds

Once a current waveform has been predicted for a particular speed, a method of using the co-energy changes to predict the torque is used. In an ideal system it can be seen that at a certain point in time the rotor will be in a particular position, with the phase winding containing a value of current. Therefore at this point a new flux-linkage can be calculated. A new profile can be drawn on the flux-linkage current diagram, which is the actual profile followed by the motor winding. This allows a new value of co-energy to be obtained. This calculation of co-energy allows a new torque to be predicted.

5.6.4 Sources of Error

With all measurement systems that measure dynamic results, errors will arise due to calibration and system characteristics. Errors will also arise because of the assumptions made in simplifying the model. Therefore a margin of error must be allowed for measurements. The largest source of error is neglecting the iron losses as this will reduce fluxes. The prediction methods also assume an instantaneous voltage change and ideal switches producing a perfect square voltage pulse. The prediction method interpolates between the aligned, mid, and unaligned positions to obtain flux-linkage characteristics for use in the virtual work method. Again the area enclosed by the operating trajectory is integrated by an estimation method, which is limited by the discretisation of the integration steps. At speeds higher than 40 rps the current waveform prediction is more susceptible to instability. The motor generally was subject to resonance when the speed was greater than 25 rps. This is a limiting factor of most full stepping current drives. This resonance effect

was not attempted to be modelled in this simulation as the effect is highly dependent to by the motor shaft coupling, which would be difficult to model.

5.7 Discussion

Adaptation of the co-energy virtual work methods has been carried out to provide torque characteristics for single and two phase commutation. A new understanding of the magnet detent operation has been examined by using the virtual work method and an explanation of how the magnet interacts with the phase excitation is shown.

Verification of experimental static results from the use of three dimensional finite element studies and the mathematical model derived in chapter 4 has been shown. A model has been devised for predicting current in the windings under dynamic operation. The model uses information that was calculated by the HyStep program, hence it can be easily incorporated to allow predictions of the torque versus speed curves for full stepping motion. Despite the assumptions which have been made in the modelling, the results produced are consistent with experimental results.

CHAPTER 6

NEW SIZE 42 FRAME SINGLE STACK HYBRID STEPPING MOTOR DESIGN FOR IMPROVED DYNAMIC PERFORMANCE

6.1 Background

Recently manufacturers have been concentrating on improving the torque performance of hybrid stepping motors by changing the magnet material in the rotor rather than changing the historically rigid tooth designs [12]. Low speed torque has been dramatically improved, though unfortunately, many new designs have suffered from little improvements at mid to high speeds. Stebon Ltd., sponsors of this project specialise in high powered shaft output designs and a design brief was given to improve the 1101 size 42 motor at mid to high speeds without resorting to winding changes. Ideally the physical construction and the manufacturing process of the motor were not to be modified. The criteria pointed towards modifying the tooth structure on the lamination. The following investigation concentrated on the stator lamination, in particular the tooth design. This would avoid expensive retooling and a design could be constructed without any special equipment and training.

Historically the hybrid stepping motor tooth profile has been designed with a typically square (parallel) stator tooth. The stator teeth are generally larger in width than the rotor teeth to achieve improved low speed torque with high positional accuracy. It is well

known that the tooth profile has an effect on the torque output of the motor especially in variable reluctance motors. In the hybrid stepping motor altering the tooth profile changes the inductance, and thus can aid current rise at higher motor speeds. This in turn increases the co-energy area at the operating speed and allows a greater available torque to the motor.

6.2 Modelling a New Tooth Design

According to Singh [67], four sets of equations are required to adequately model a stepping motor. These are:

- Voltage equations of the electrical circuits
- The change of flux linkage and inductance relative to rotor position
- Developed torque, including stored energy and co-energy
- Dynamic equations of the rotor

The modelling by the three dimensional finite element package OPERA-3D offers a reasonable possibility to predict the first three accurately. The HyStep software uses assumptions and simplifications that degrade prediction, but has favourable computation time, file size, and memory attributes and can model all four sets of equations. Furthermore, HyStep's internal flux path calculator, assumes a parallel sided tooth geometry, which makes it difficult to model vastly different tooth designs. However the tooth geometry can be studied using 2-D FEA, as described in chapter 4. This data can then be used in HyStep with the dynamic model described in chapter 5.

6.3 The Effect of Tooth Design on Torque Production

The typical design of tooth profiles is summarised in a few parameters (figure 6.1);

- stator slot depth - Generally around ½ tooth pitch
- Tooth width/ tooth pitch ratio - Found to be between 0.38 to 0.45 (Harris et al propose 0.42 is suggested as an optimum value [68]). Tooth pitch equals the summation of the tooth and slot width.
- Slot shape between teeth, typically semi-circular on the rotor, but can be found to be rectangular on the stator.
- Air gap length which should be kept as small as possible.

The ratio of stator tooth width/ rotor tooth width is known to affect holding torque capability. The higher this ratio, the higher the holding torque as the lines of magnetic flux are held under tension, thus opposing rotor disturbance more rigidly. A balance will have to be obtained for dynamic torque.

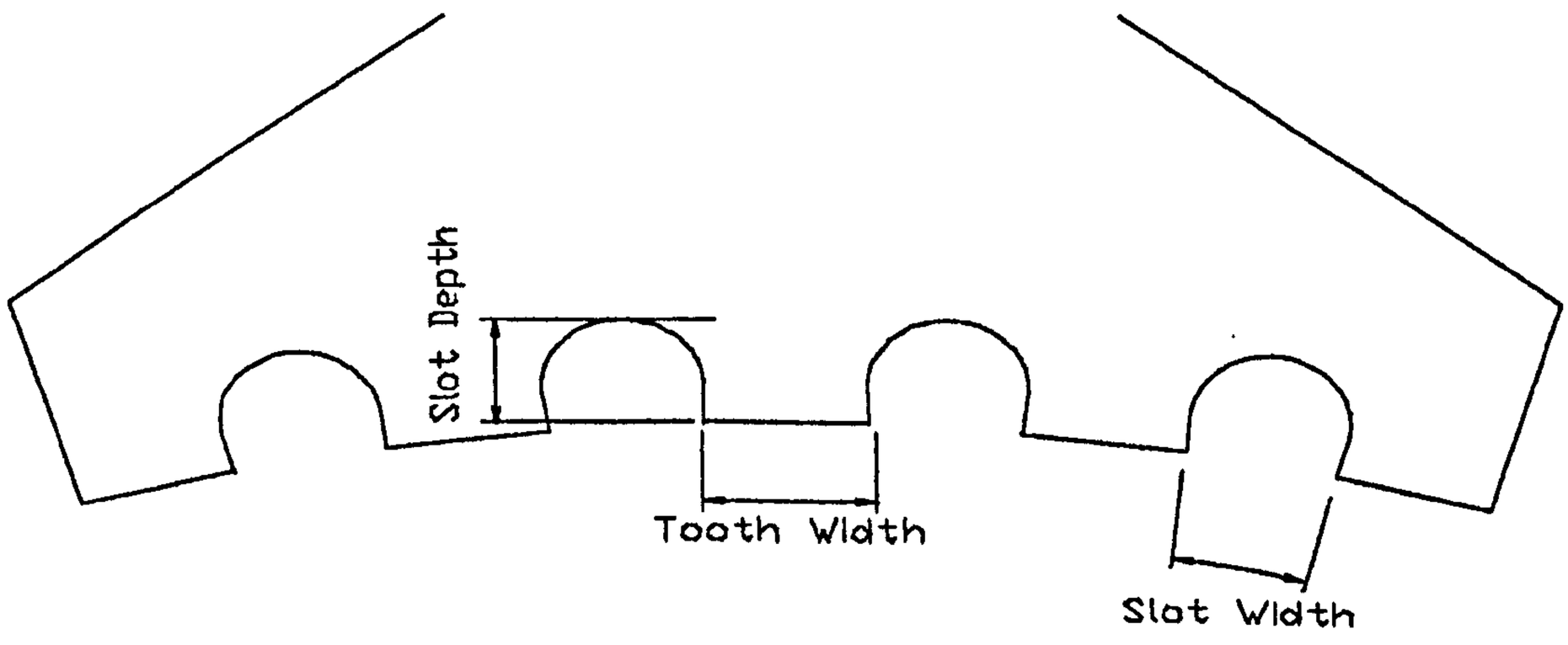


Figure 6.1 - Tooth and Slot depth Considerations

The torque of a motor is inversely proportional to the square of the air gap and directly proportional to the stack length. It is desirable to have a long motor and it is desirable to

have smooth tooth faces, as surface irregularities increase the air gap length, thus reducing torque and impeding smooth operation.

Wagner et al [69] investigated the quantitative effects of tooth/ slot ratio and slot shape on the magnetic circuit. It was concluded that the choice of rectangular or semi-circular stator slots has little effect on influencing the performance of a motor, however the rounded slots were marginally more effective. In a further study Wagner et al [70], analysed the leakage flux between adjacent pole pieces of a DC motor and found within reasonable design structures, that the leakage loss was around 1% to 4% of total flux. D.Torney [71] presented data for doubly slotted structures. He suggested that the tooth slot depth to slot width ought to be higher than 0.4 and that a tooth slot to tooth width ratio ought to be as high as 1.5 to ensure low air gap permanence. However in the modelling in this thesis little difference is observed when making the slot too deep, as in the un-aligned position, flux lines from the edge of the stator teeth tend to fringe into the rotor teeth edges. Keeping the slot type as a semi-circular shaped recess proves marginally more beneficial over a rectangular slot, as long as the rotor teeth slot to stator teeth width ratio is higher than 1.5.

Akiyama and Sueki [72] proposed that the side profiles of the stepping motor tooth contributed to the way leakage flux effectively determines torque production in a variable reluctance motor. It was proposed that tapered teeth could contribute greater torque at higher excitations.

Although the last study was not directly aimed at the hybrid stepping motor it suggests that a new design may affect the dynamic performance of the motor. From the study of current wave forms the effect of the $L\frac{di_1}{dt}$ term of the equation;

$$V_1 = i_1 R_1 + L\frac{di_1}{dt} + \omega n_r \lambda_{mag-max} \sin(n_r \theta), \qquad (6.1)$$

is a major contribution in the effect of current rise time and hence mid/ high speed torque. It has been suggested by Akiyama and Sueki [72] that a dove tail tooth profile would offer maximum holding torque while an inverted or tapered tooth a higher dynamic torque, figure 6.2a, and 6.2b. The idea of tapered teeth leads to the use of exponential shaped tooth profiles to improve motor performance.

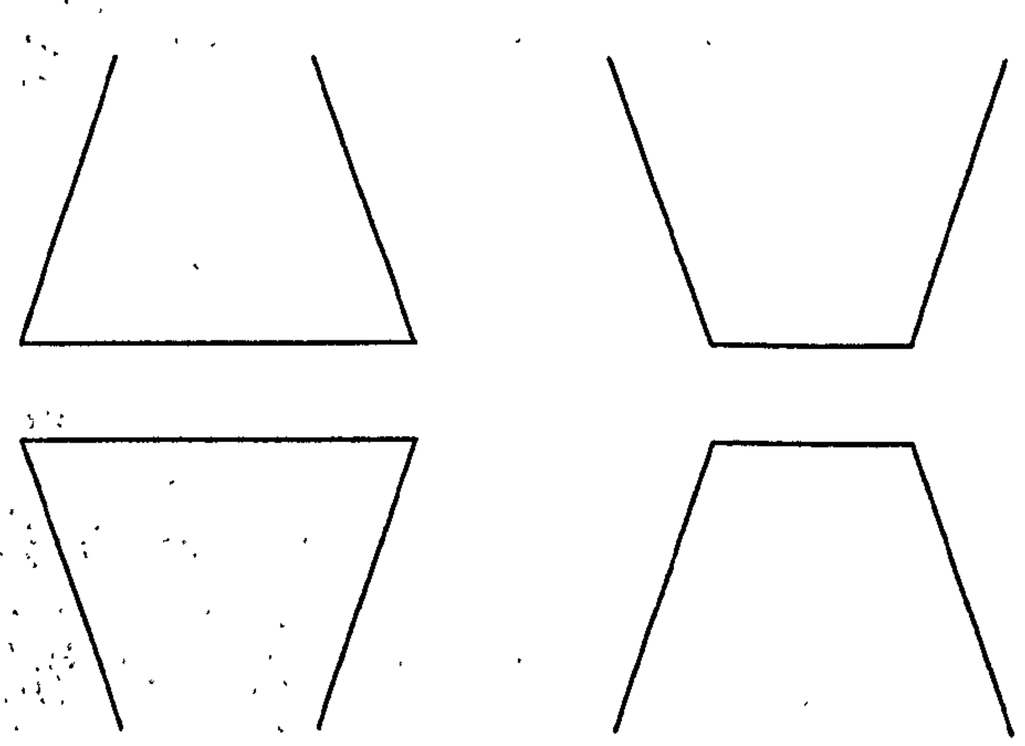


Figure 6.2a/b - Dove Tail and Tapered Tooth Profile

6.4 Selection of a New Tooth Profile

To overcome problems that might be encountered with providing sufficient space for the copper conductors, the advantages that may have been obtained in altering the reluctance path in the back iron and stator pole width were disregarded to concentrate on the tooth profile effects. Before selecting a new profile the standard design of tooth profile was analysed.

6.4.1 Standard Configuration

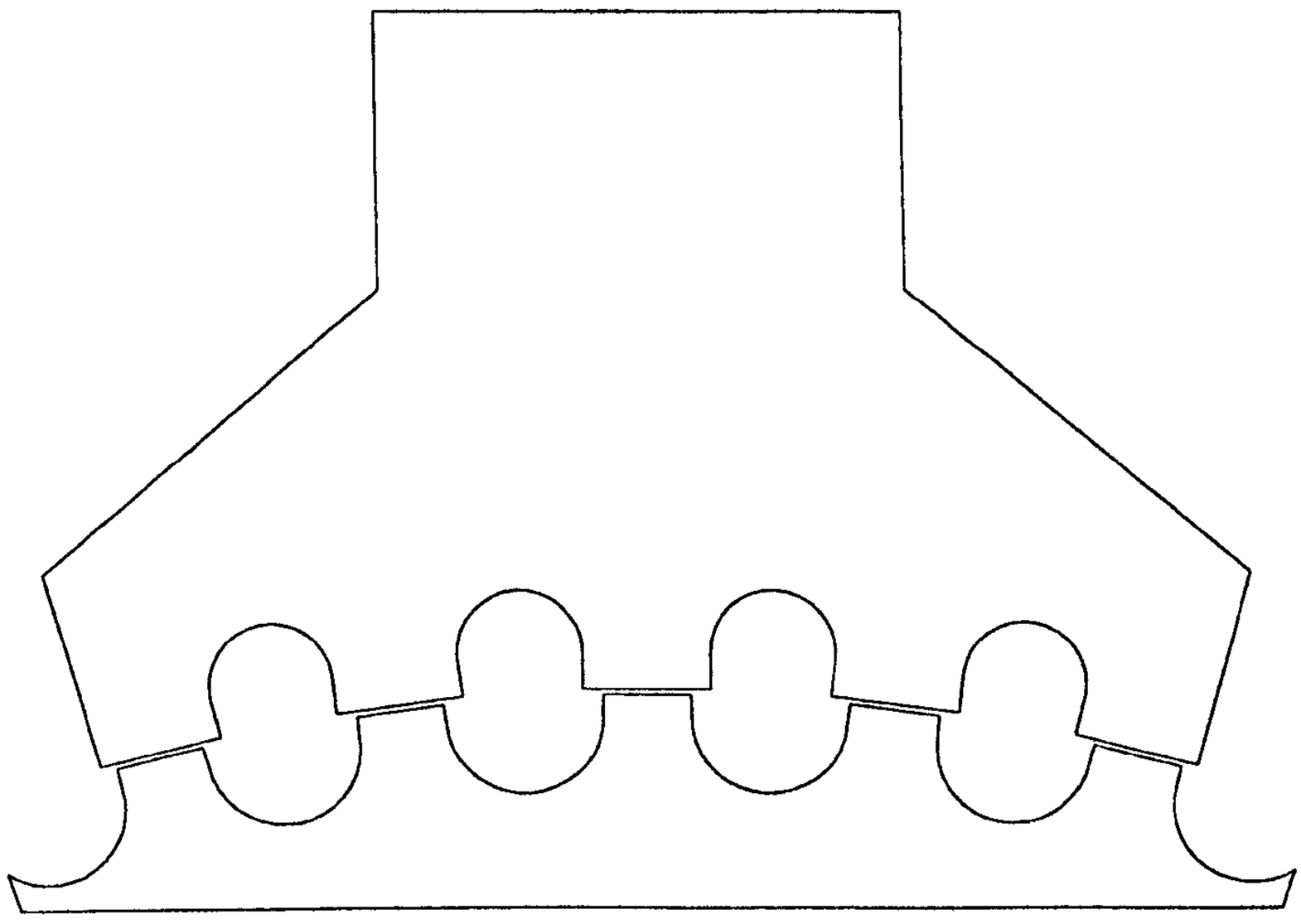


Figure 6.3 - Standard Configuration Tooth Geometry

This design (figure 6.3) is the standard Stebon 1100 series hybrid stepping motor with square rotor and stator teeth. It provides the comparison datam against which any changes will be measured.

Analysis was undertaken using the Opera 2-d finite element package, using the methodology presented in chapter 4. Flux densities were produced showing the magnitude of flux density across the tooth faces of the stator and rotor. These showed that the vast majority of aligned flux passes through the central portion of the stator teeth with little contribution from the corners. However, most unaligned flux passes through the corners.

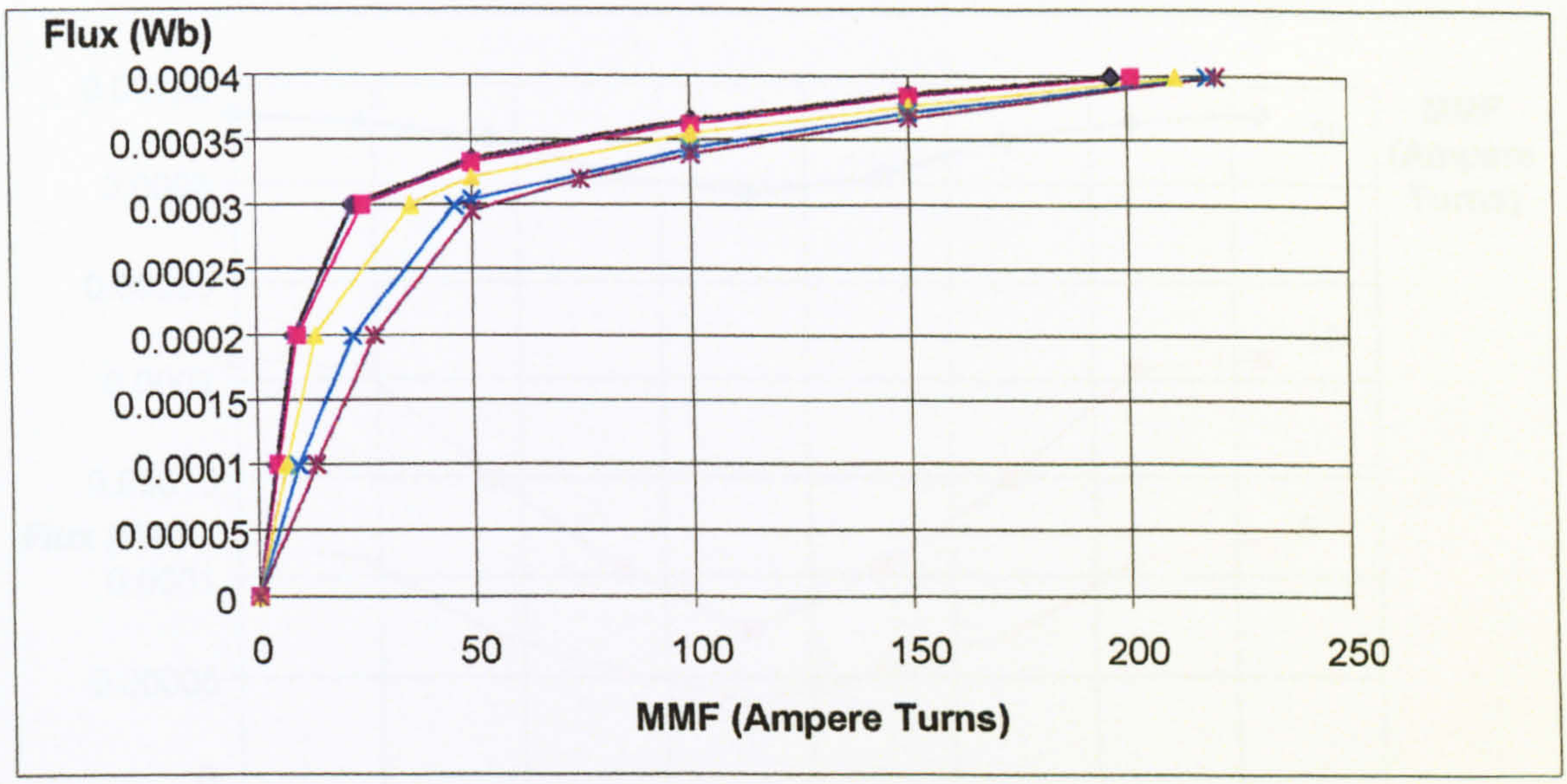


Figure 6.4 - Graph of Flux vs. MMF for the 'Standard' Configuration in (top to bottom) the Aligned, ¾ aligned, Semi--aligned, ¼ aligned and Un-aligned Positions

Figure 6.4 shows the flux versus MMF relationship for several positions between the aligned and unaligned locations. From top to bottom, the lines represent fully aligned, ¾ aligned, semi--aligned, ¼ aligned and fully unaligned respectively. The area under each section was integrated to obtain changes in co-energy and these are presented in table 6.1.

	Change in Co-energy (J)
Aligned to unaligned	7.03
Aligned to semi—aligned	4.07
Semi--aligned to unaligned	2.96

Table 6.1 - Performance of the Standard Configuration

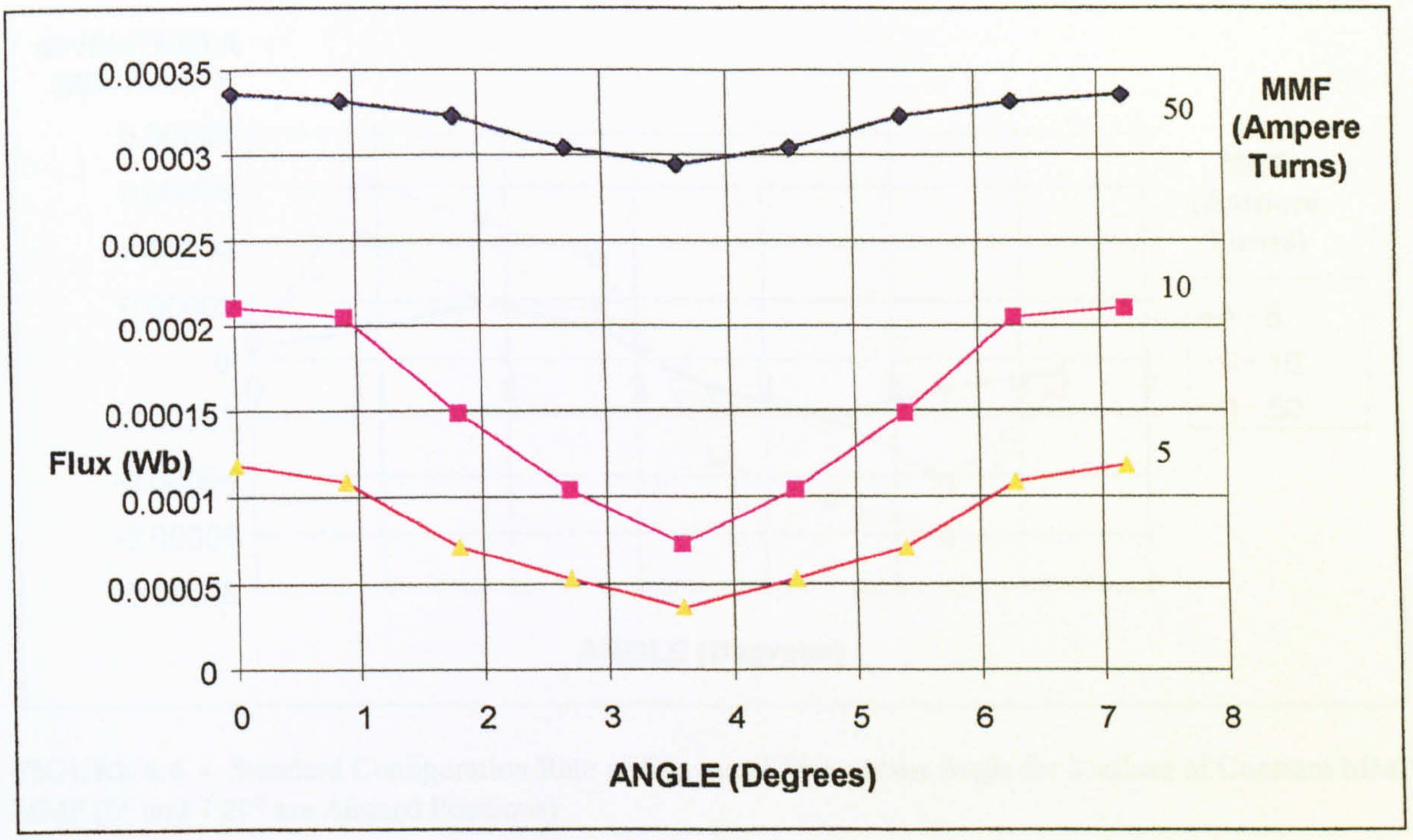


FIGURE 6.5 - Standard Tooth Configuration: Flux versus Angle for 3 values of Constant MMF (0° and 7.20° are Aligned Positions)

Figures 6.5 and 6.6 show the variation in flux and rate of change of flux respectively against angle. In the flux versus angle relationship the waveform resembles a sinusoid with some low harmonic content. This harmonic content is due to the square edges of the teeth creating sharp rates of change of reluctance. As a consequence, the induced EMF profile, which is important when designing control circuits and considering high speed operation, is not sinusoidal. This also indicates torque ripple over a full step, where a larger change of flux with angle is found with constant MMF. This effect is multiplied as the co-energy change is concentrated in the area before the knee of saturation in the flux/ MMF curves.

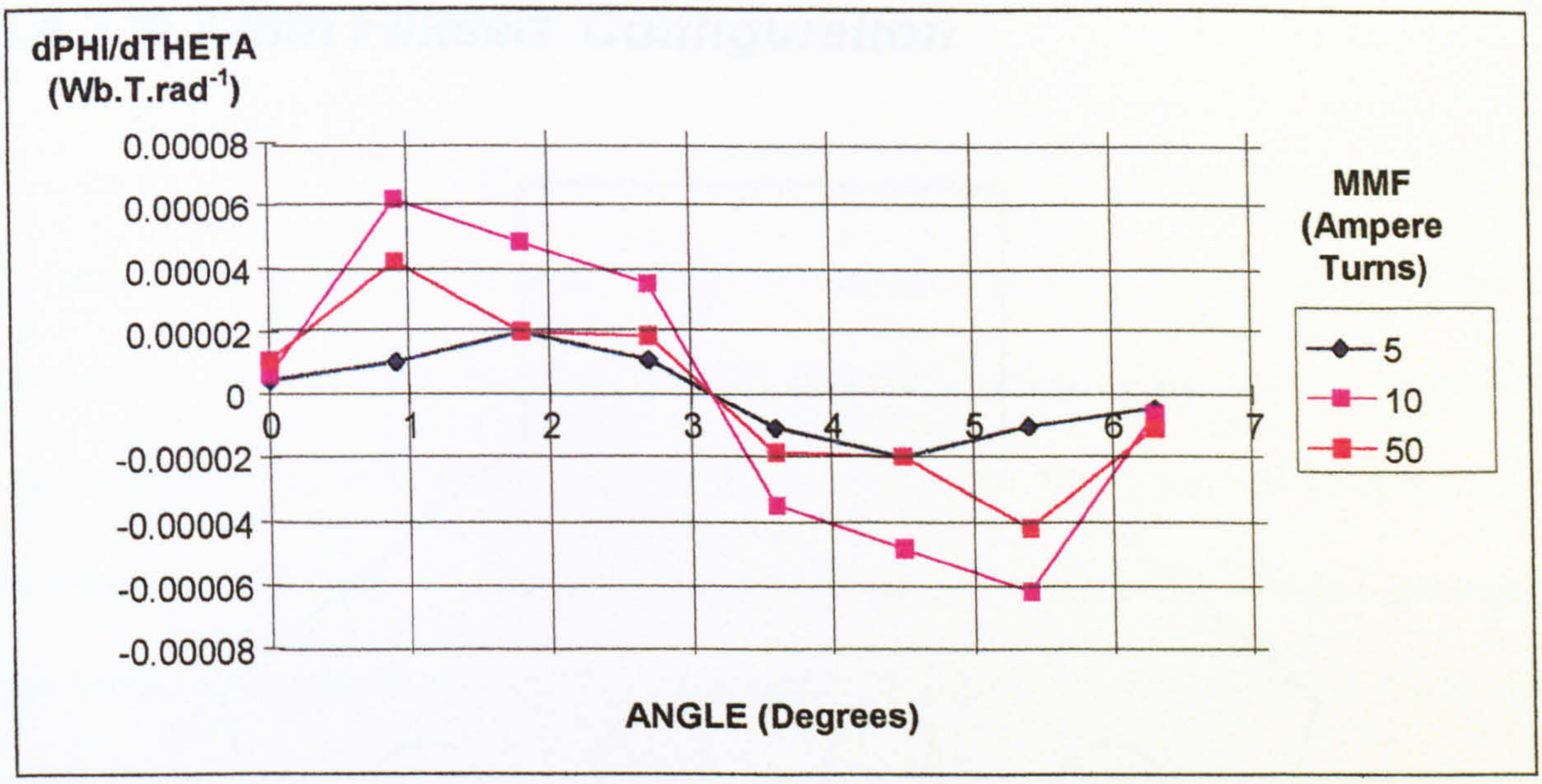


FIGURE 6.6 - Standard Configuration Rate of Change of Flux versus Angle for 3 values of Constant MMF MMF (0° and 7.20° are Aligned Positions)

From studies of flux paths across the teeth of the standard motor and inspection of the flux density and flux paths it was noted that the central area of the tooth carries most of the flux efficiently, it would appear impossible to improve aligned flux without, say, reducing the air-gap. However, the corners of the stator teeth which carry little aligned flux also carry most undesirable unaligned flux. Therefore, it is a modification of the corners of the teeth that has been concentrated upon.

6.4.2 Proposed Designs

An investigation was undertaken to find the methods of reducing the harmonic effects of the tooth edges, whilst maintaining a high aligned to un-aligned co-energy value. Two designs were selected from the investigation and are detailed in the following sections.

6.4.3 '0.7 mm Filleted' Configuration

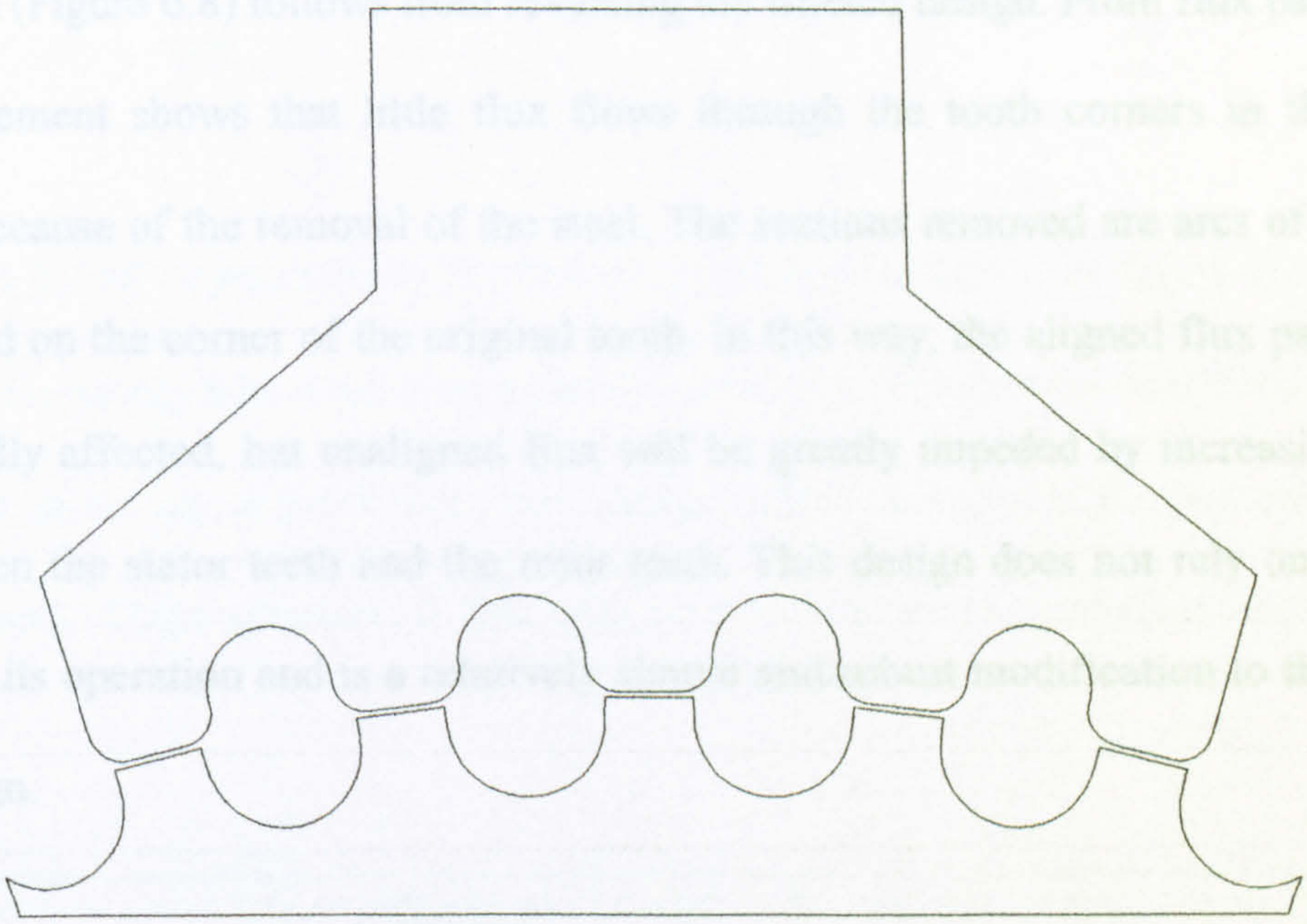


Figure 6.7 '0.7 Filleted' Configuration Proposed Tooth Geometry

Figure 6.7 shows the alternative design which has a 0.7 mm filleted radius on the tooth edge. This attempts to produce a smoother transition between rotor positions. Variations of the filleted radius were attempted. Using a 0.7 mm radius appeared to give the most reasonable results. Reducing the radius of the fillet in this tooth design did not provide a sizeable advantage over the standard profile.

An examination of the flux path diagrams showed that some flux will still flow into the sides of the rotor teeth which will maintain holding torque performance, while unaligned flux is impeded by the length of the air-gap created and the corners of the rotor teeth saturating.

6.4.4 '0.5 mm Chopped' Configuration

This design (Figure 6.8) follows from reversing the filleted design. From flux path studies, this arrangement shows that little flux flows through the tooth corners in the aligned position, because of the removal of the steel. The sections removed are arcs of radius 0.5 mm centred on the corner of the original tooth. In this way, the aligned flux paths should be minimally affected, but unaligned flux will be greatly impeded by increasing the air-gap between the stator teeth and the rotor teeth. This design does not rely on saturation effects for its operation and is a relatively simple and robust modification to the standard tooth design.

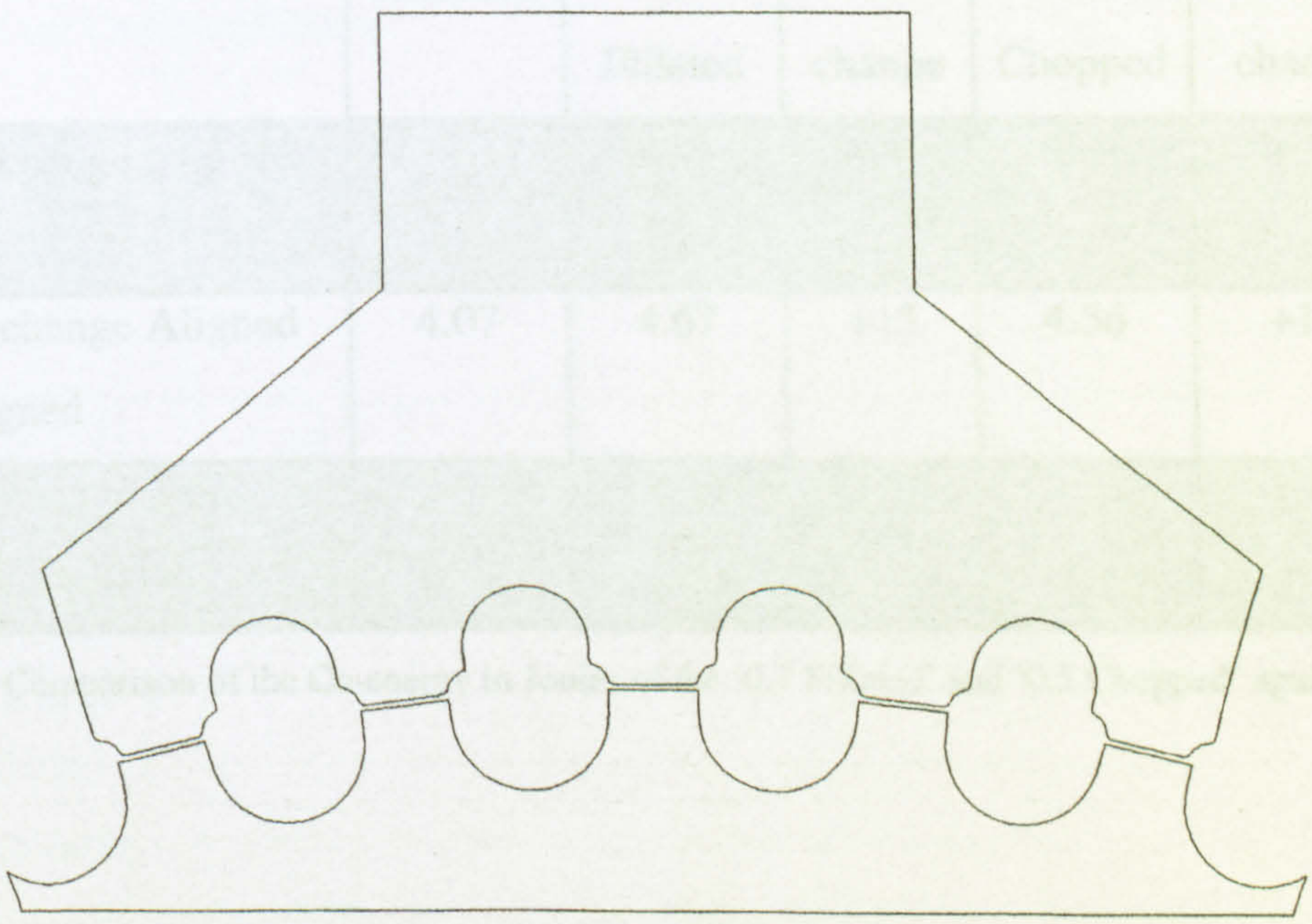


FIGURE 6.8 - 'Chopped' Configuration Proposed Tooth Geometry

6.4.5 Comparisons of Electromagnetic Characteristics

The flux to MMF characteristics of the filleted design are shown in figure 6.9. Table 6.2 shows the comparable change in co-energy. For the filleted design significant improvements were found to be between the semi-aligned and unaligned positions where

the improvement is around 27%. From the aligned to un-aligned and semi-aligned, there is also a substantial increase in co-energy. This increase of co-energy is available to be transformed into torque.

The flux to MMF characteristics for the chopped design shown in figure 6.10 give the changes in co-energy which are summarised in table 6.2. The data demonstrates that, while not greatly affecting aligned flux, this design offers a substantial improvement over standard teeth. Three quarters of this gain is between the semi-aligned and un-aligned positions.

	Standard	0.7 Filleted	% change	0.5 Chopped	% change
Co-energy change Aligned to un-aligned	7.03	8.42	+20	8.51	+21
Co-energy change Aligned to semi-aligned	4.07	4.67	+15	4.56	+12
Co-energy change Semi-aligned to un-aligned	2.96	3.75	+27	3.96	+34

Table 6.2 - Comparison of the Co-energy in Joules of the '0.7 Filleted' and '0.5 Chopped' against 'Standard'

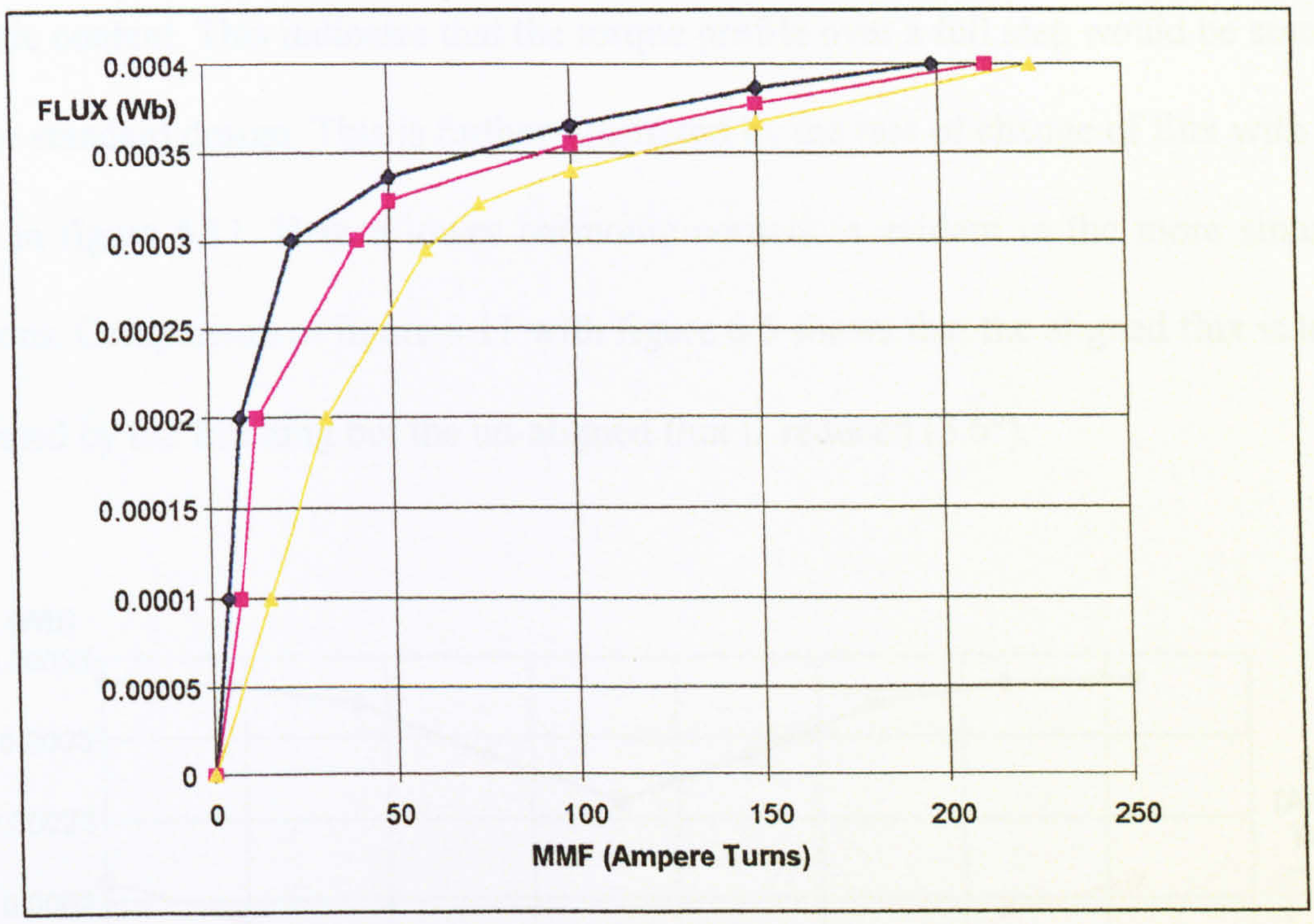


FIGURE 6.9 - Graph of Flux vs. MMF for the '0.7 Filleted' Configuration in (top to bottom) the Aligned, Semi-aligned and Unaligned Positions

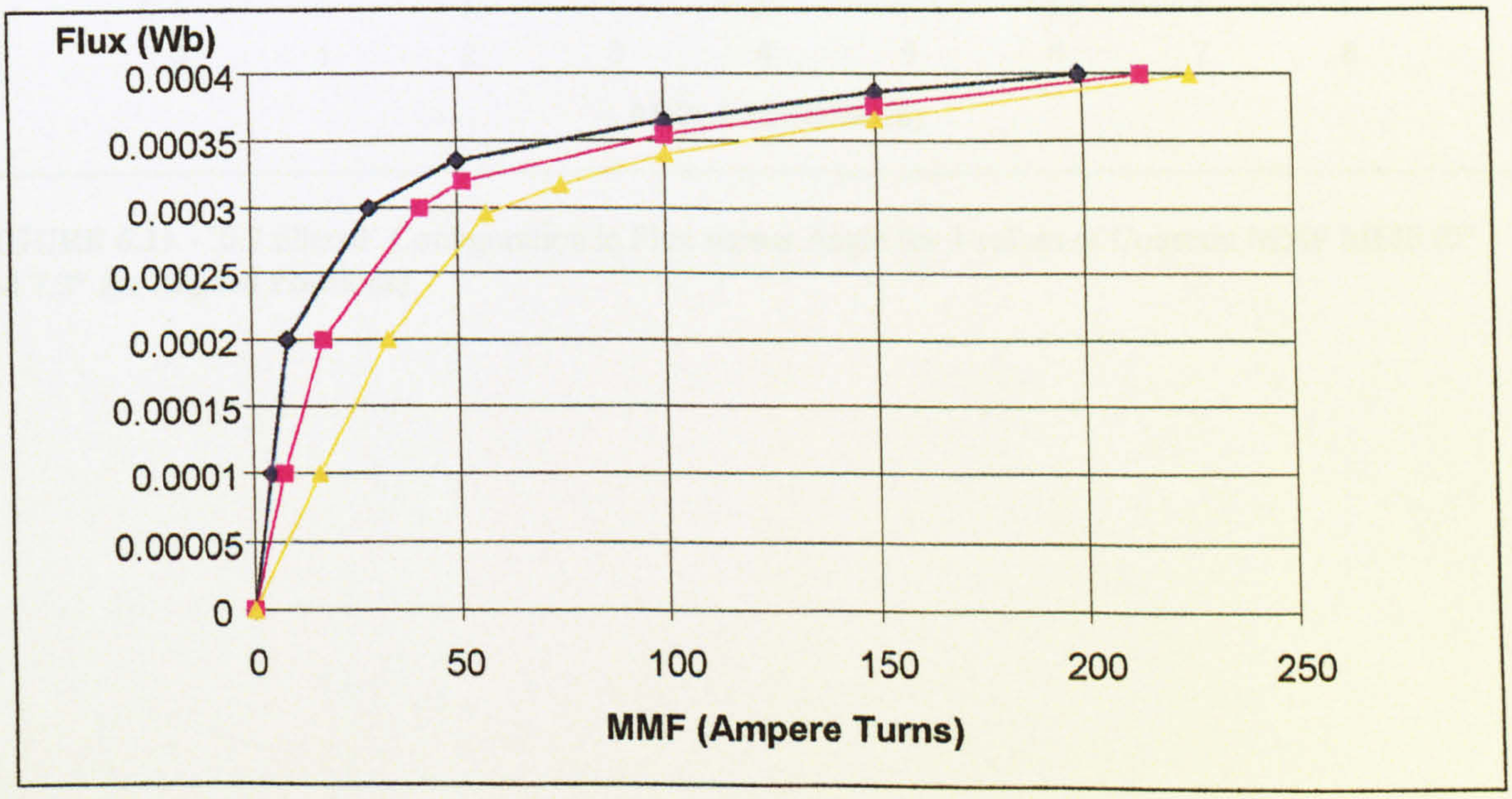


FIGURE 6.10 - Graph of Flux vs. MMF for the 'Chopped' Configuration in (top to bottom) the Aligned, Semi-aligned and Unaligned Positions.

Figures 6.11 and 6.12 show the flux and rate of change of flux respectively for the '0.7 mm filleted' design plotted against angle. The flux variation has a reduced low order

harmonic content. This indicates that the torque profile over a full step would be smoother over the standard design. This is further illustrated by the rate of change of flux with angle shown in figure 6.11. Here a lower harmonic content is evident in the more sinusoidal variations. Comparison of figure 6.11 with figure 6.5 shows that the aligned flux is largely unaffected by the filletting but the un-aligned flux is reduced (3.6°).

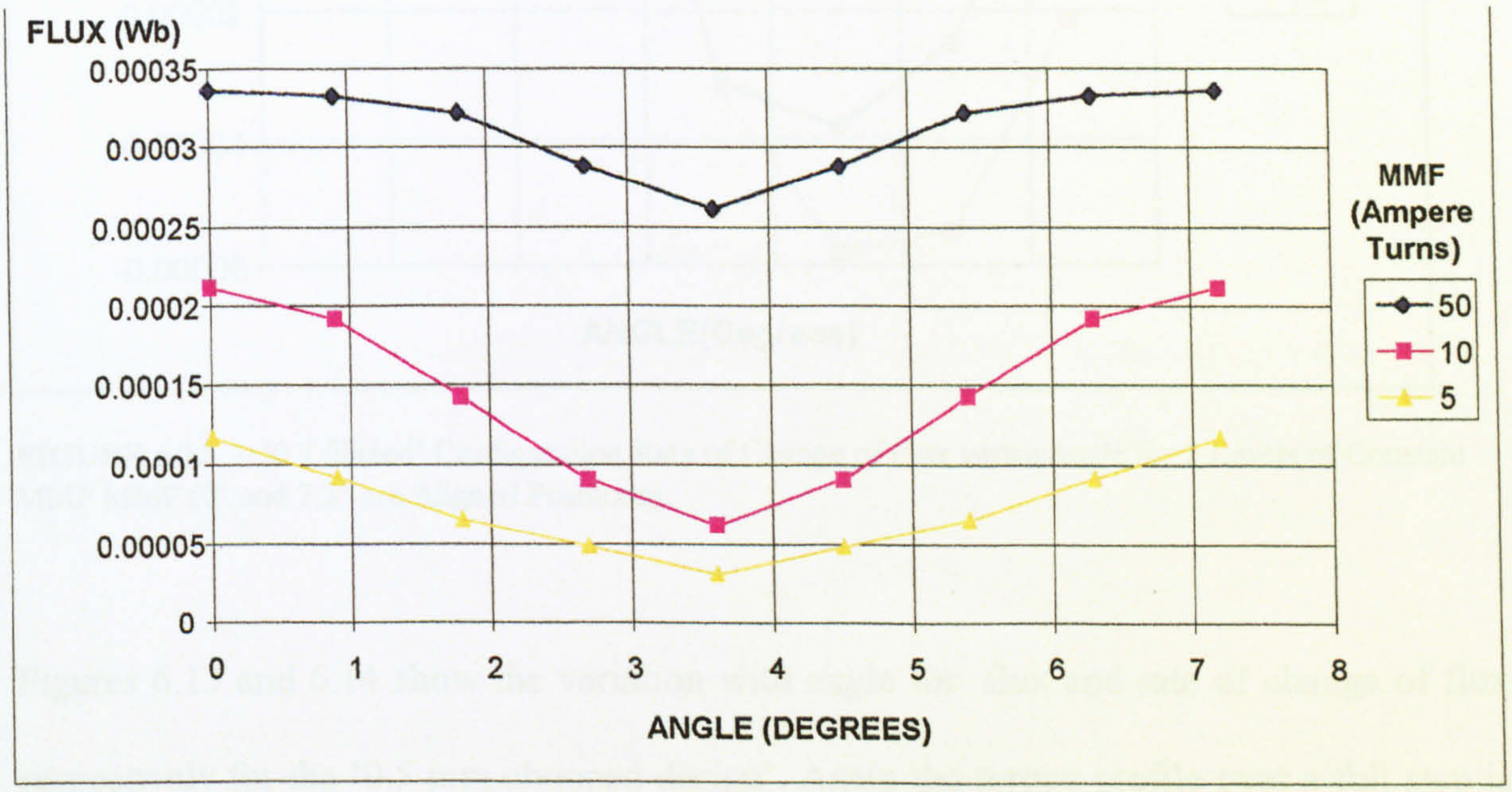


FIGURE 6.11 - '0.7 filleted' Configuration in Flux versus Angle for 3 values of Constant MMF MMF (0° and 7.2° are Aligned Positions)

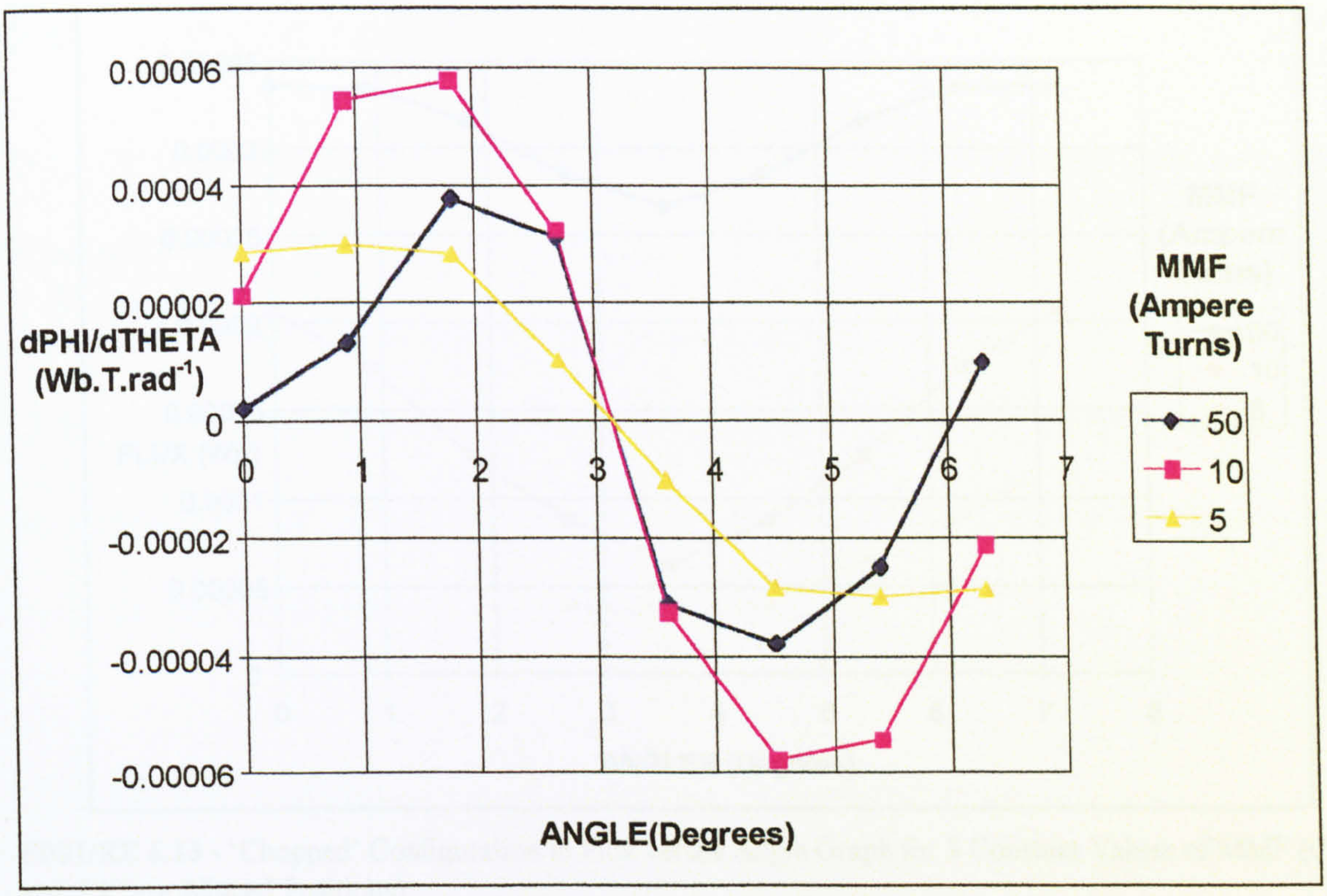


FIGURE 6.12 - '0.7 filleted' Configuration Rate of Change of Flux versusAngle for 3 Levels of Constant MMF MMF (0° and 7.2° are Aligned Positions)

Figures 6.13 and 6.14 show the variation with angle for flux and rate of change of flux respectively for the '0.5 mm chopped design'. Again the torque profile over a full step is will be smoother, as a result of the more consistent sinusoidal characteristic.

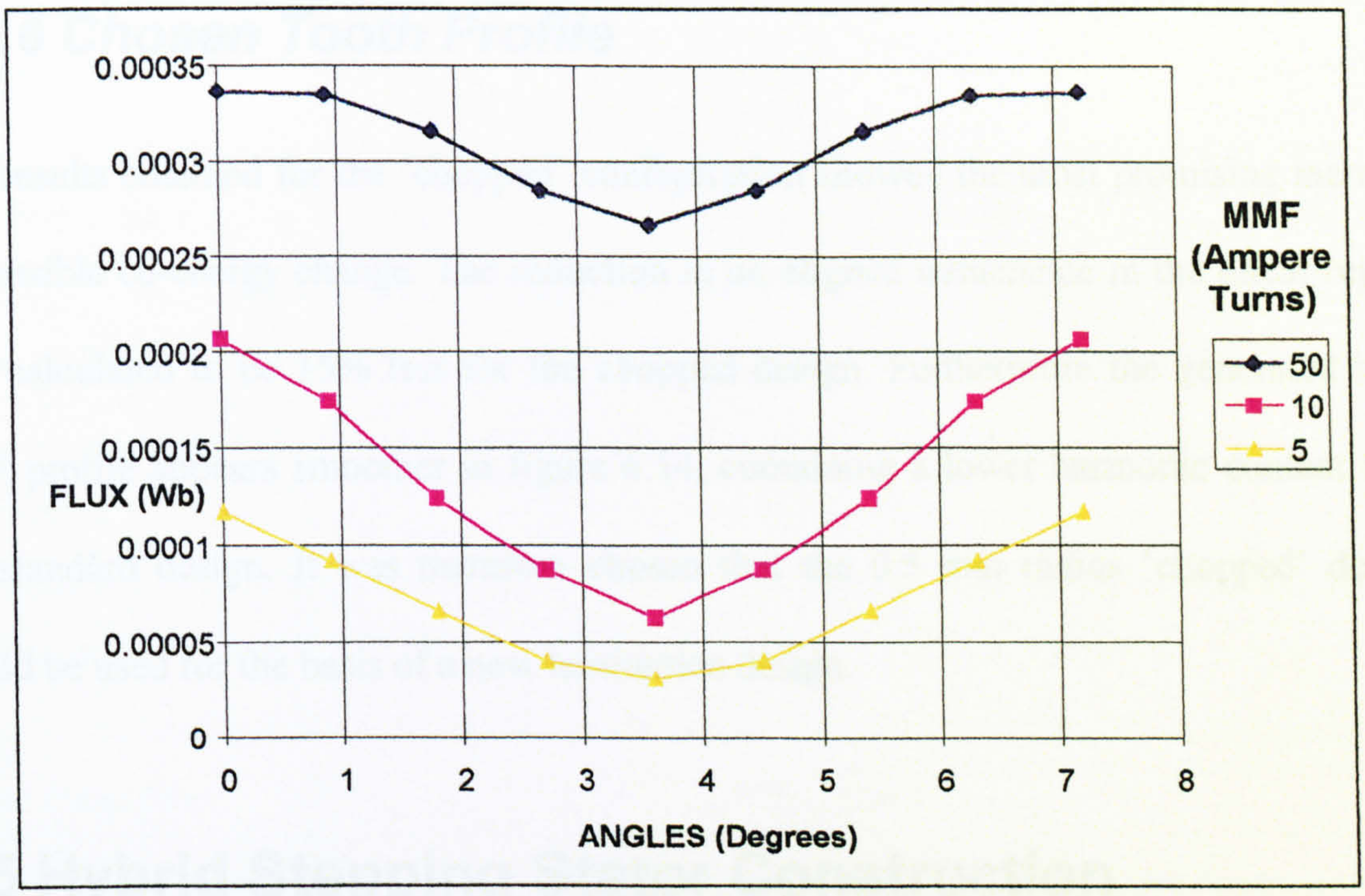


FIGURE 6.13 - 'Chopped' Configuration in Flux versus Angle Graph for 3 Constant Values of MMF (0° and 7.2° are Aligned Positions)

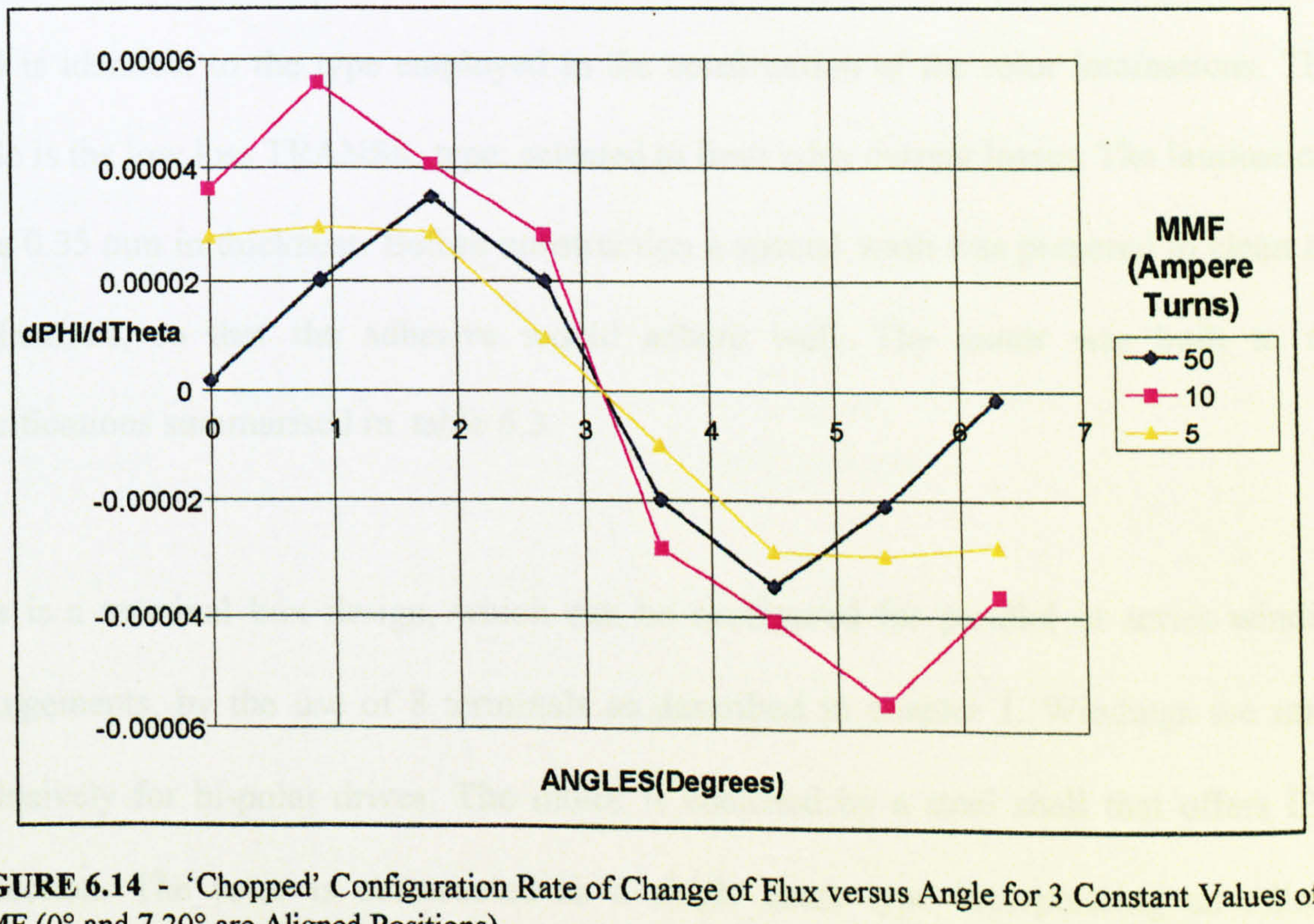


FIGURE 6.14 - 'Chopped' Configuration Rate of Change of Flux versus Angle for 3 Constant Values of MMF (0° and 7.20° are Aligned Positions)

6.4.6 Chosen Tooth Profile

The results obtained for the 'chopped' configuration showed the most promising increase in possible co-energy change. The reduction in un-aligned inductance in the linear region was calculated to be 15% less for the chopped design. Furthermore the generated back EMF profile appears smoother in figure 6.14, containing a lower harmonic content than the standard design. It was therefore chosen that the 0.5 mm radius 'chopped' design would be used for the basis of a new lamination design.

6.5 Hybrid Stepping Stator Construction

The stator laminations were wire eroded from sheets of silicon steel. The choice of silicon steel is identical to the type employed in the construction of the rotor laminations. This grade is the low loss TRANSIL type, selected to limit eddy current losses. The laminations were 0.35 mm in thickness. Before construction a special wash was prepared to clean the laminations, so that the adhesive would adhere well. The motor was built to the specifications summarised in table 6.3.

This is a terminal box design, which can be configured for parallel or series winding arrangements, by the use of 8 terminals as described in chapter 1. Windings are made exclusively for bi-polar drives. The motor is enclosed by a steel shell that offers IP44 protection. The rotor is constructed as a single stack type incorporating an Alinico Hycomax-3 magnet. The motor's outside dimensions are shown in figure 6.15. The procedure of constructing the motor is described in chapter 2. The design met the design criteria of being able to be built without any major training or equipment change.

Winding Type	250-75
Phase Resistance	0.26 Ohms in parallel
Maximum Bipolar Amps per Phase	7 Amps (two phase on) per phase.
Rotor Inertia	3.65 $Kgcm^2$
Total weight	4.8 Kg
Frame/ Flange Size	NEMA size 42
Step Angle	1.8 Degree Standard (200 steps per rev)
Winding Insulation	Class F to IEC 85 and BS2757
Turns per phase	96
Maximum supply voltage	400V DC
Bearing Loading	Axial, 180 N max. Radial 330N max.

Table 6.3 - Motor Specifications

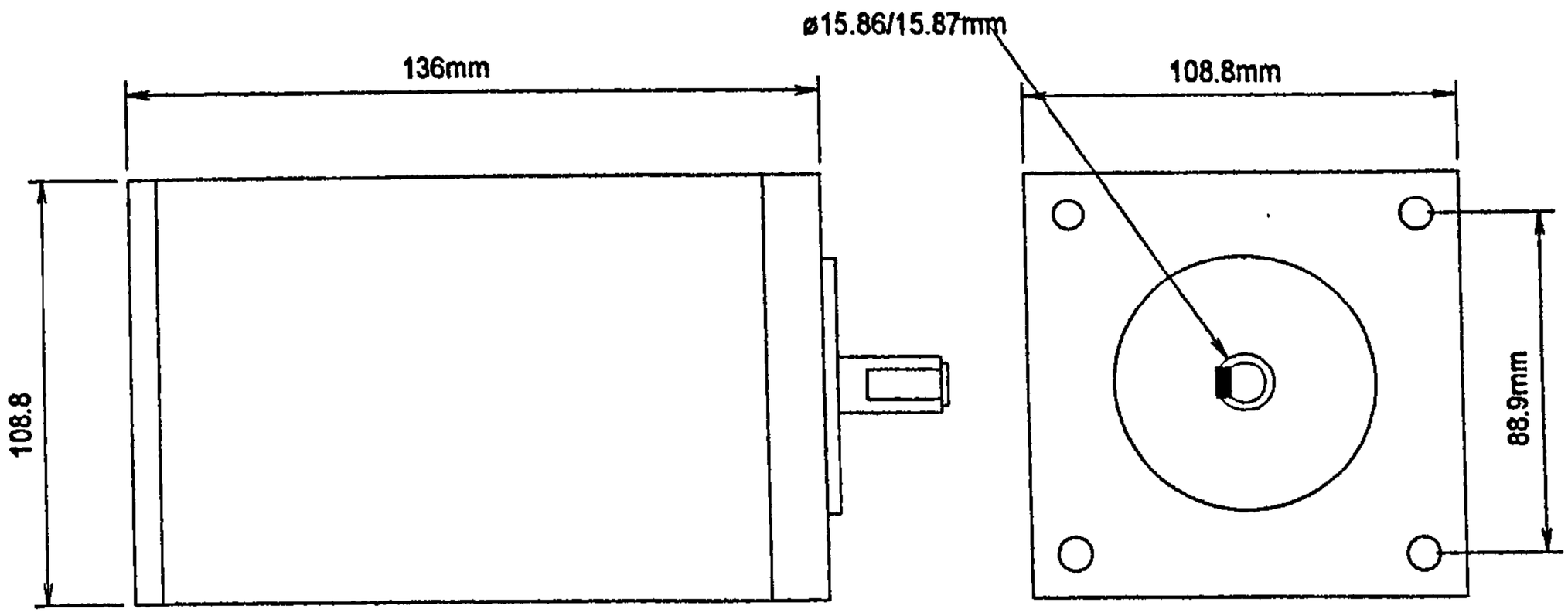


FIGURE 6.15 - Hybrid Stepping Motor Type 1101 External Dimensions

6.6 Static Torque Expectations

In the standard design the stator tooth width is greater than the rotor tooth. Generally this aids two-phase static torque as, when the rotor sits between the two phase poles, there is still a large steel path present. In the simplified stepping motor of figure 6.16, a two phase holding position is shown. If the 'A' phases are attracting the rotor teeth, and the 'R' are repelling the teeth, it can be understood that the larger the tooth width the greater the holding torque. A larger tooth width will produce a greater stable equilibrium torque, and a smaller unstable torque when a repelling action occurs.

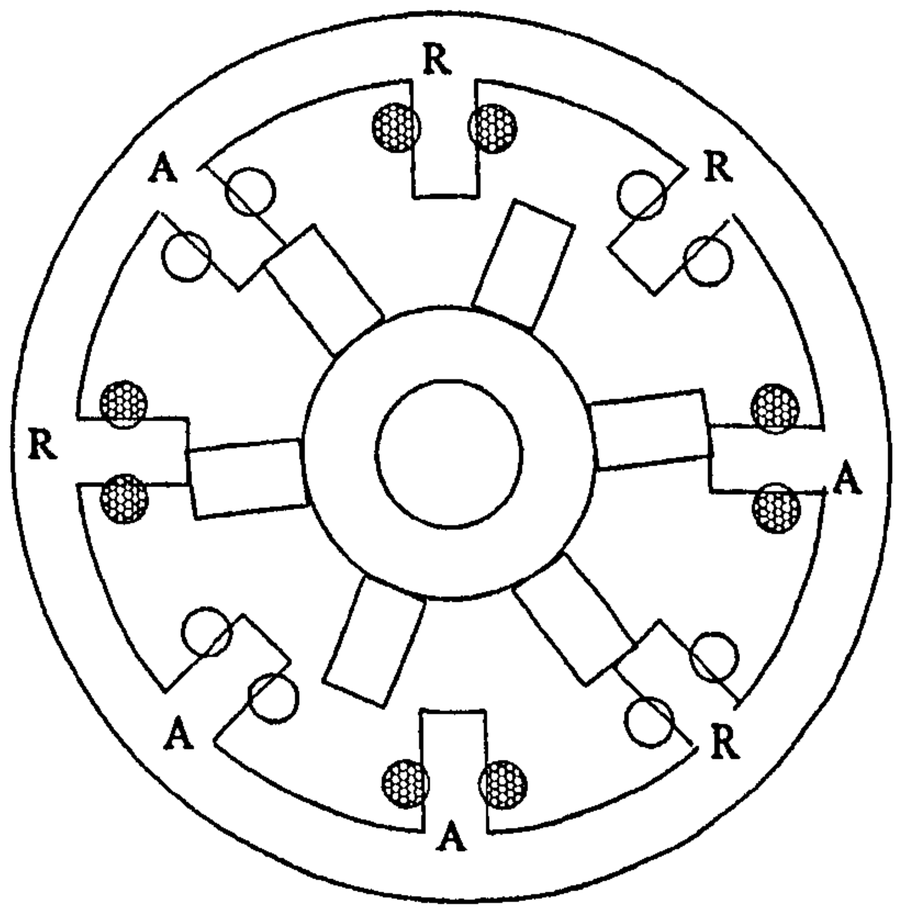


FIGURE 6.16 - Simple Stepping Motor with Two Phase Holding Torque Position

With a smaller tooth width this situation reverses. The unstable equilibrium torque will increase, but the stable holding torque will reduce; the unstable torque being more advantageous in dynamic torque production.

6.6.1 Experimental Low Speed Torque

Table 6.5 shows the torque output at 1 rps. This data is also shown in figure 6.17. This particular test was done with the drive (SX8) supplying a sinusoidal current waveform with a 5000 step resolution into the parallel windings of the motor. If the test had been undertaken in series the current would have to be halved, but the effects of inductance would have been negligible at this test speed.

Current (RMS AMPS)	8	7	6	5	4	3
Standard Design	3.76	3.44	3.06	2.63	2.1	1.67
Dynamic Design	3.65	3.3	2.94	2.57	2.06	1.61

Table 6.5 - Low Speed (1 rps) Torque Differences between Motor Designs

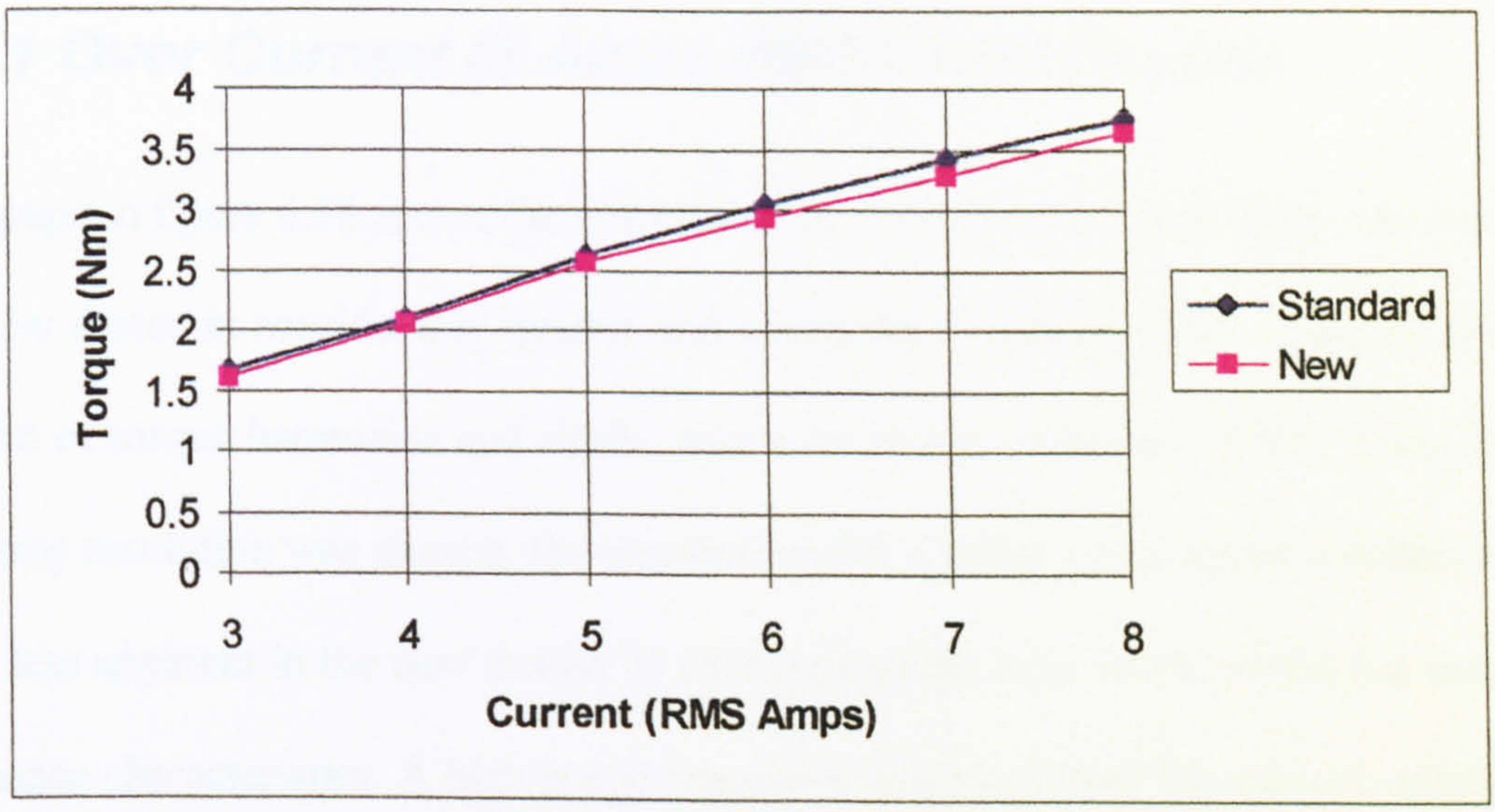


FIGURE 6.17 - Low Speed Torque on SX8 Drive 5000 Steps per Revolution

As the standard design has more steel present in the rotor to stator path in the holding position, it fairs better at high excitation currents. The top current is actually higher than the motor's rated value (7 Amps RMS). The torque reduction from the new design ranges between a 3 - 3.6 %.

6.7 Dynamic Testing

Dynamic testing of both the motors was undertaken on several drives. The drive used for the majority of the tests is known as the Parker-Compumotor SX8 [3]. This drive offers a wide speed range due to its 120 V DC bus and offers substantial current control in terms of choice of magnitudes and resolution. Maximum current is 8 Amps RMS with a resolution of 25,000 steps per rev. Testing was undertaken on the test rig described in chapter 2. The majority of the tests were with the motor windings in parallel, and the drive resolution set to 5000 steps per revolution. This would deliver an approximate sinusoidal current waveform into the motor.

6.7.1 Over Current (8 Amps RMS) Test Results

The graph in figure 6.18 shows the dynamic performances at 8 Amps RMS. At low speeds the new motor is considerably quieter and smoother in motion. This is due to the lower content of torque harmonics and ripple, plus a more sinusoidal back EMF. Even though a high step resolution was chosen, the standard motor exhibits a mid speed resonance, which is far less apparent in the new design. It appears that the new tooth profile has better anti-resonance characteristics. A high speed resonance occurs in both the motors, although this is not as severe in the new motor. As the new motor has a lower inductance it can achieve a higher operating speed. In this instance the new motor has an increase in speed of over 600 rpm, whilst achieving a torque which is equal to the standard design's mid speed torque. This increased speed leads to an improved shaft power output. The changes can be seen (in figure 6.19) to be quite dramatic with over 25% increase in maximum power output.

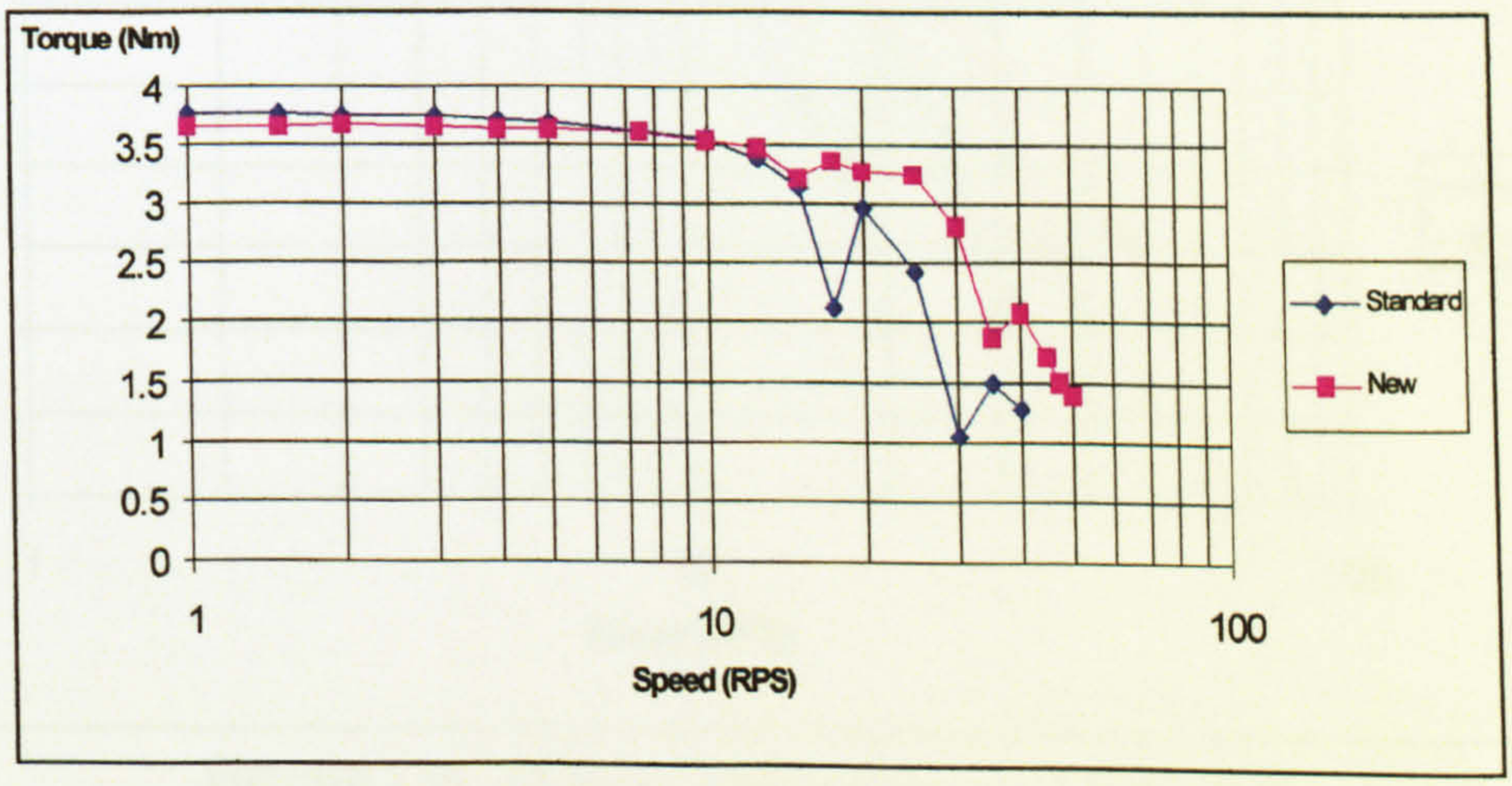


FIGURE 6.18 - S8 Drive, 5000 steps per rev, 8 Amps RMS.

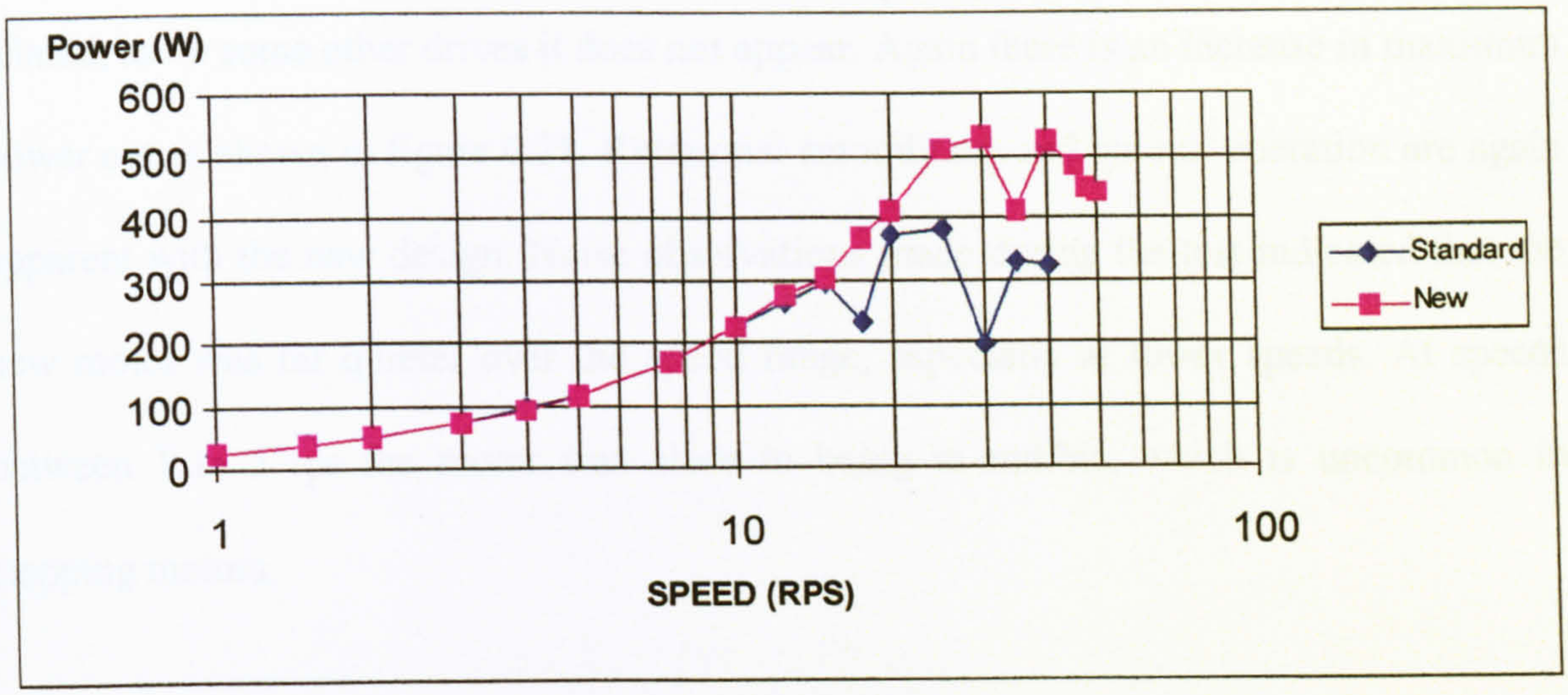


FIGURE 6.19 - Shaft Power Out at 8 Amps and 5000 Steps per Revolution Current Profile

6.7.2 Rated Current (7 Amps RMS) Test Results

At rated current the performance characteristics become more explicit. The graphs of figure 6.20 show that the mid speed resonance has virtually been eliminated by the new design.

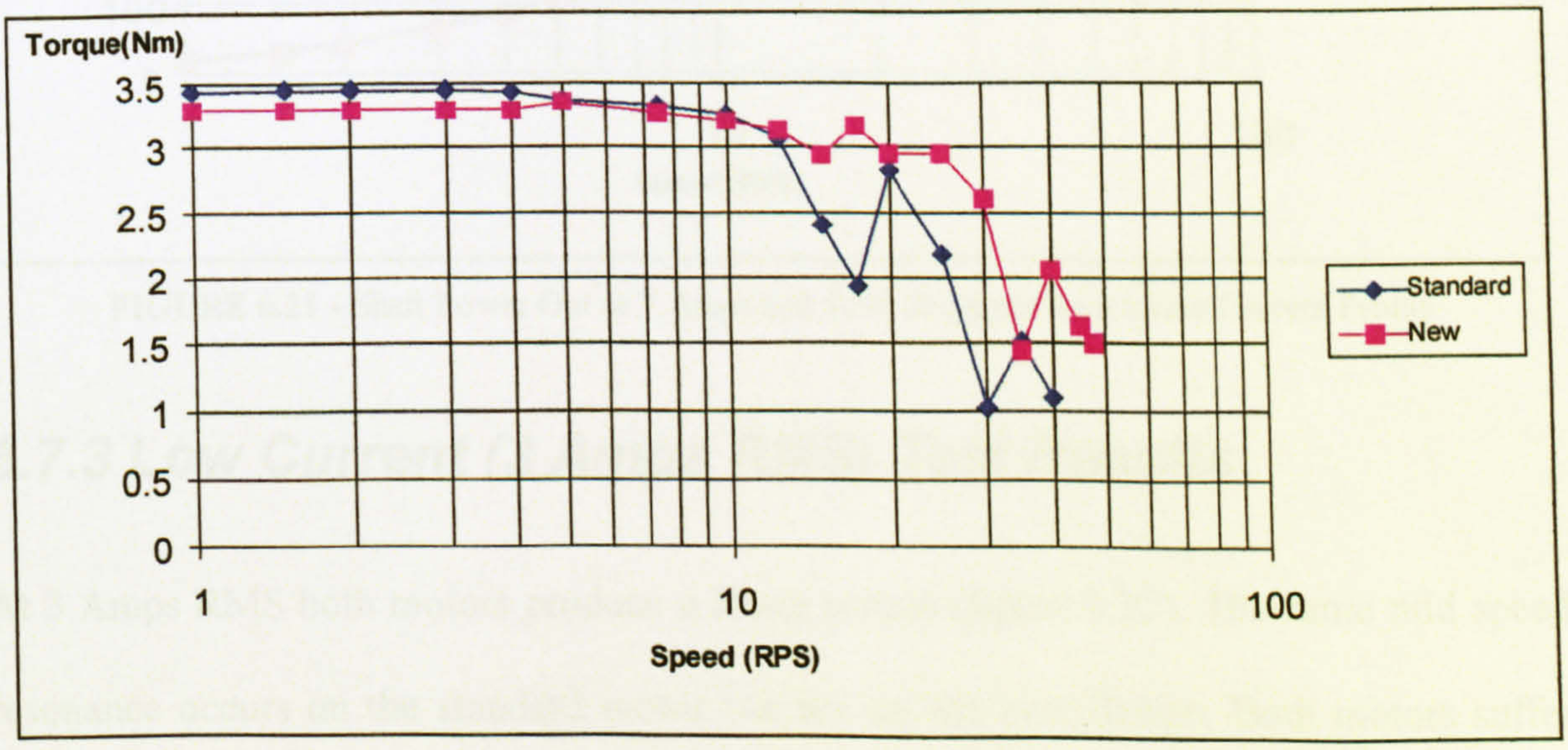


FIGURE 6.20 - S8 Drive, 5000 steps per rev, 7 Amps RMS.

Furthermore the high speed resonance appears to be pushed to a higher speed on the torque speed curve by the new design. This indicates that this resonance is possibly drive

related, as on some other drives it does not appear. Again there is an increase in maximum power out as shown in figure 6.21. Rotational smoothness and quieter operation are again apparent with the new design. Noise observations made during the test indicated that the new motor was far quieter over the speed range, especially at lower speeds. At speeds between 1 to 5 rps the motor was close to being in-audible, which is uncommon in stepping motors.

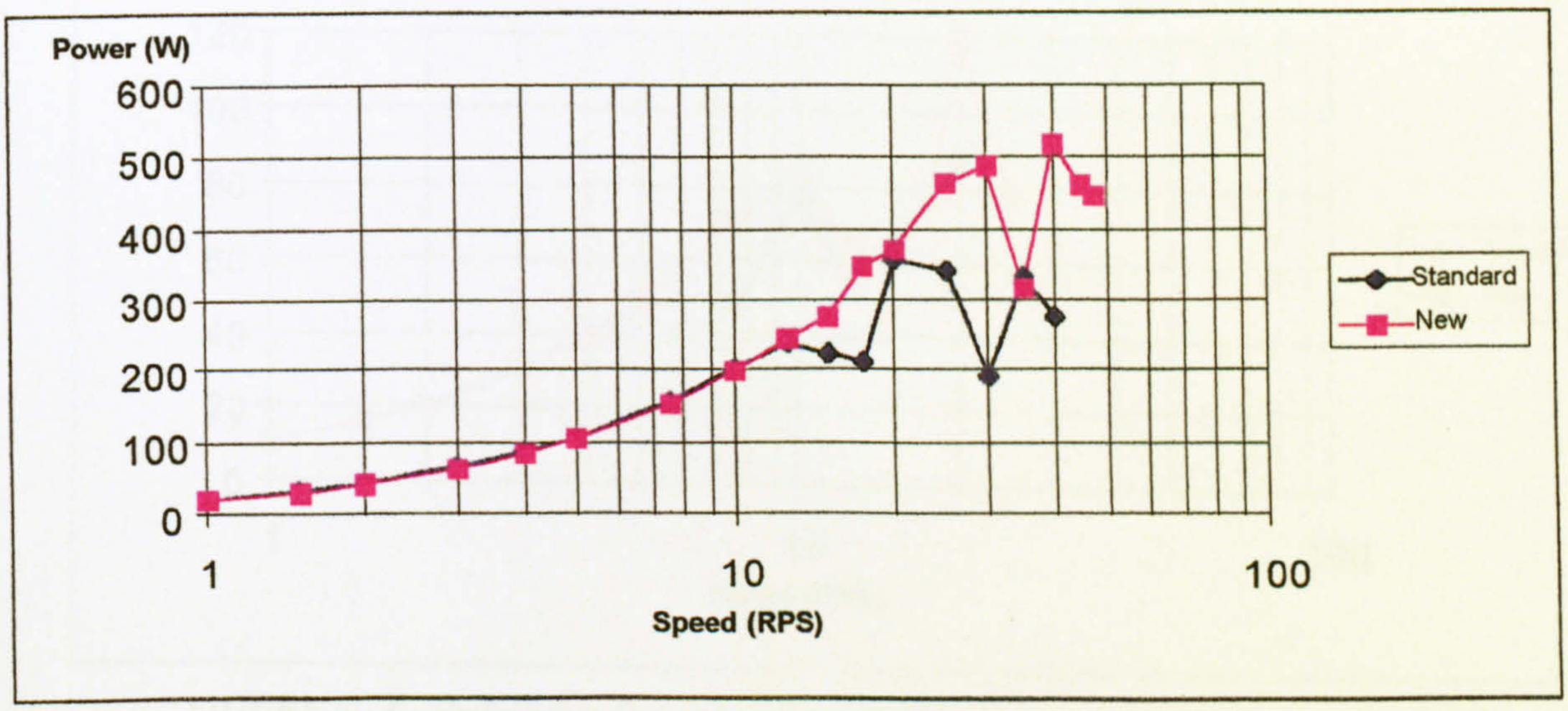


FIGURE 6.21 - Shaft Power Out at 7 Amps and 5000 Steps per Revolution Current Profile

6.7.3 Low Current (3 Amps RMS) Test Results

At 3 Amps RMS both motors produce a lower torque (figure 6.22). The same mid speed resonance occurs on the standard motor but not on the new design. Both motors suffer from the high speed drive resonance as they do not have significant current in the windings to over come this. This in turn produces rough power out curves, figure 6.23. Hence neither of the motors would be suitable to run at low currents on this drive.

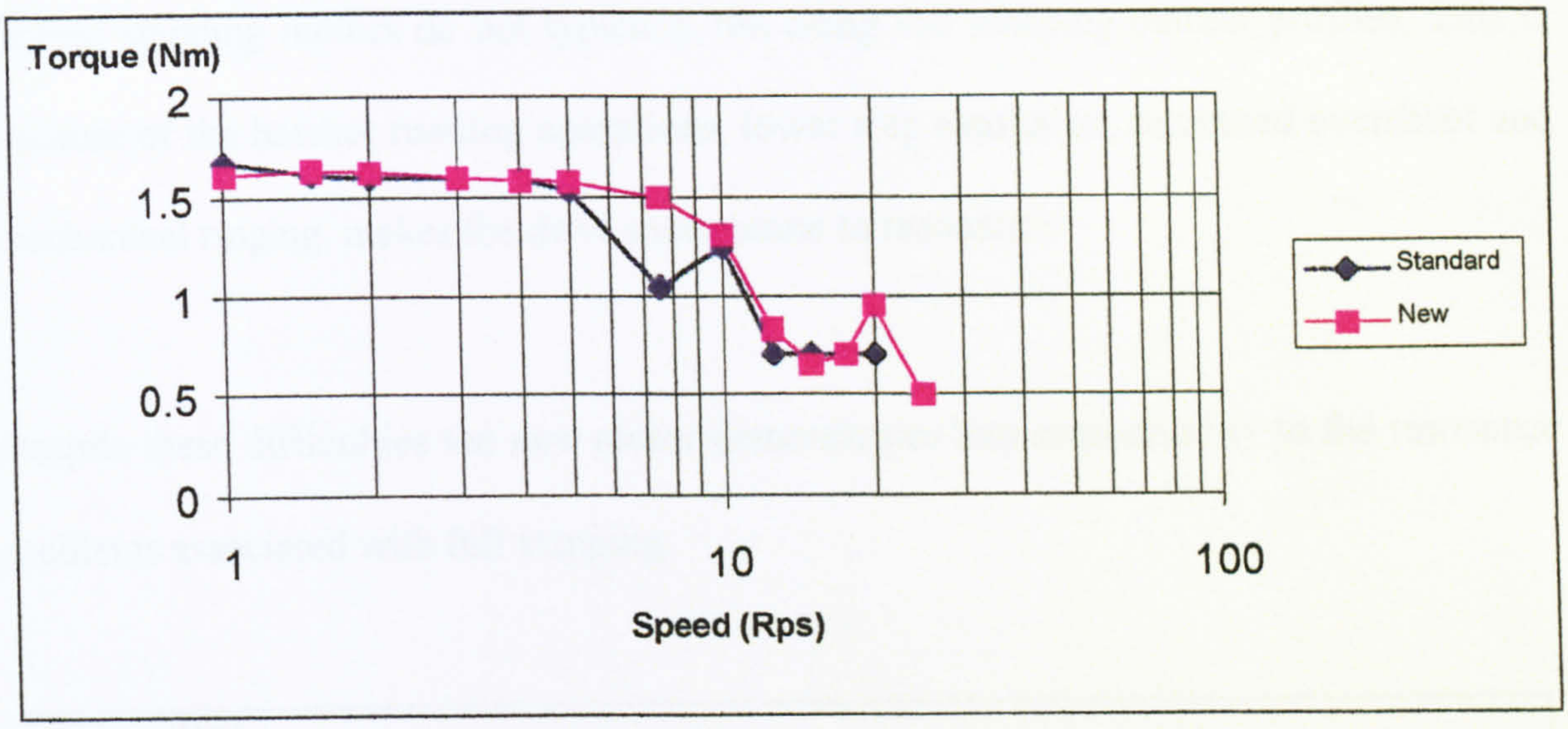


FIGURE 6.22 - S8 Drive, 5000 steps per rev, 3 Amps RMS.

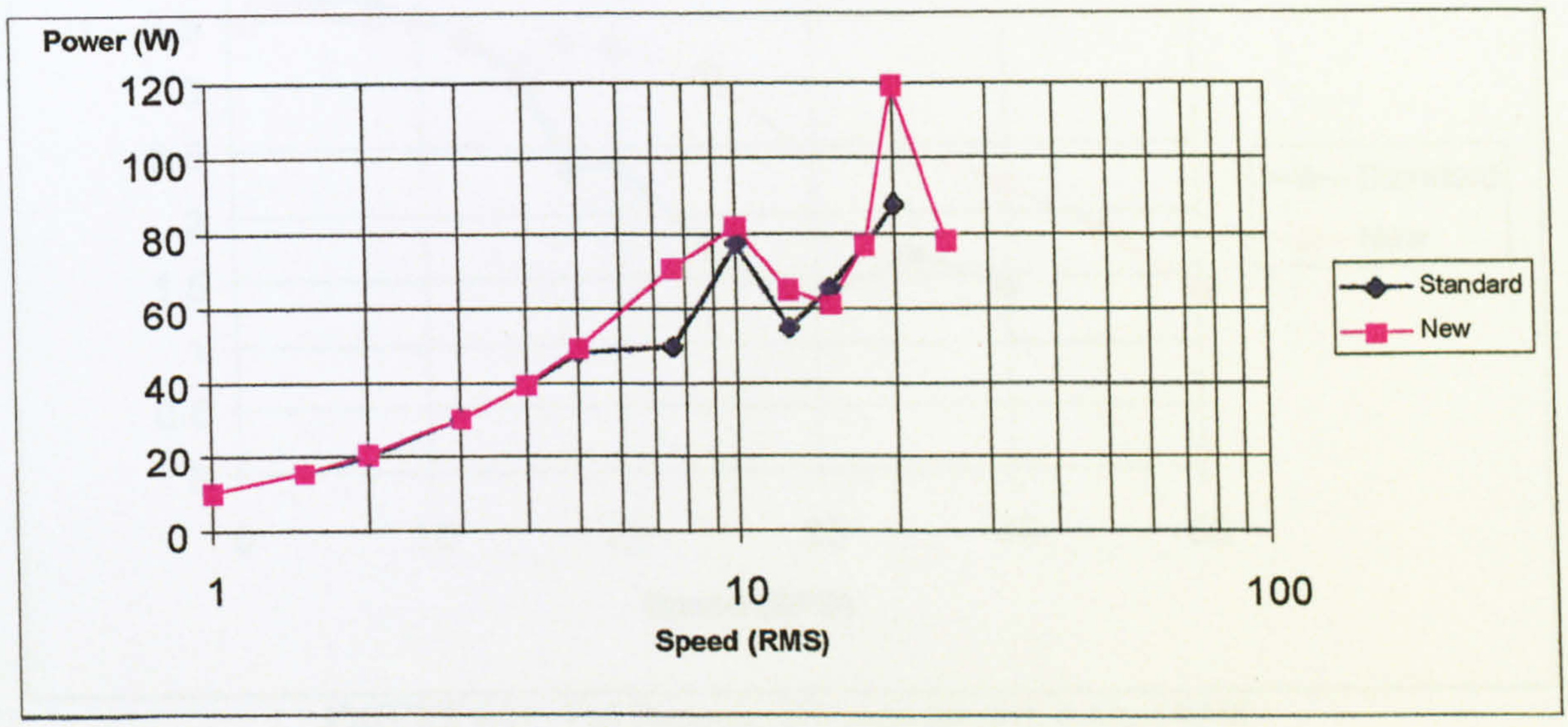


FIGURE 6.23 - Shaft Power Out at 3 Amps and 5000 Steps per Revolution Current Profile

6.8 Change of Resolution Tests Results

Figure 6.24 shows the test results of the motors running under full stepping operation. The new design of motor exhibits an increased torque output when the motor speed is over 15 rps. This resonance period affects both designs, with the new design responding more favourably. The new motor shows a significant increase in speed range.

Hybrid stepping motors do not typically run using full stepping current profiles. This is because of the harsher running operations, lower step resolution, increased overshoot and mechanical ringing, makes the drive more prone to resonance.

Despite these difficulties the new motor demonstrates less susceptibility to the resonance problems associated with full stepping.

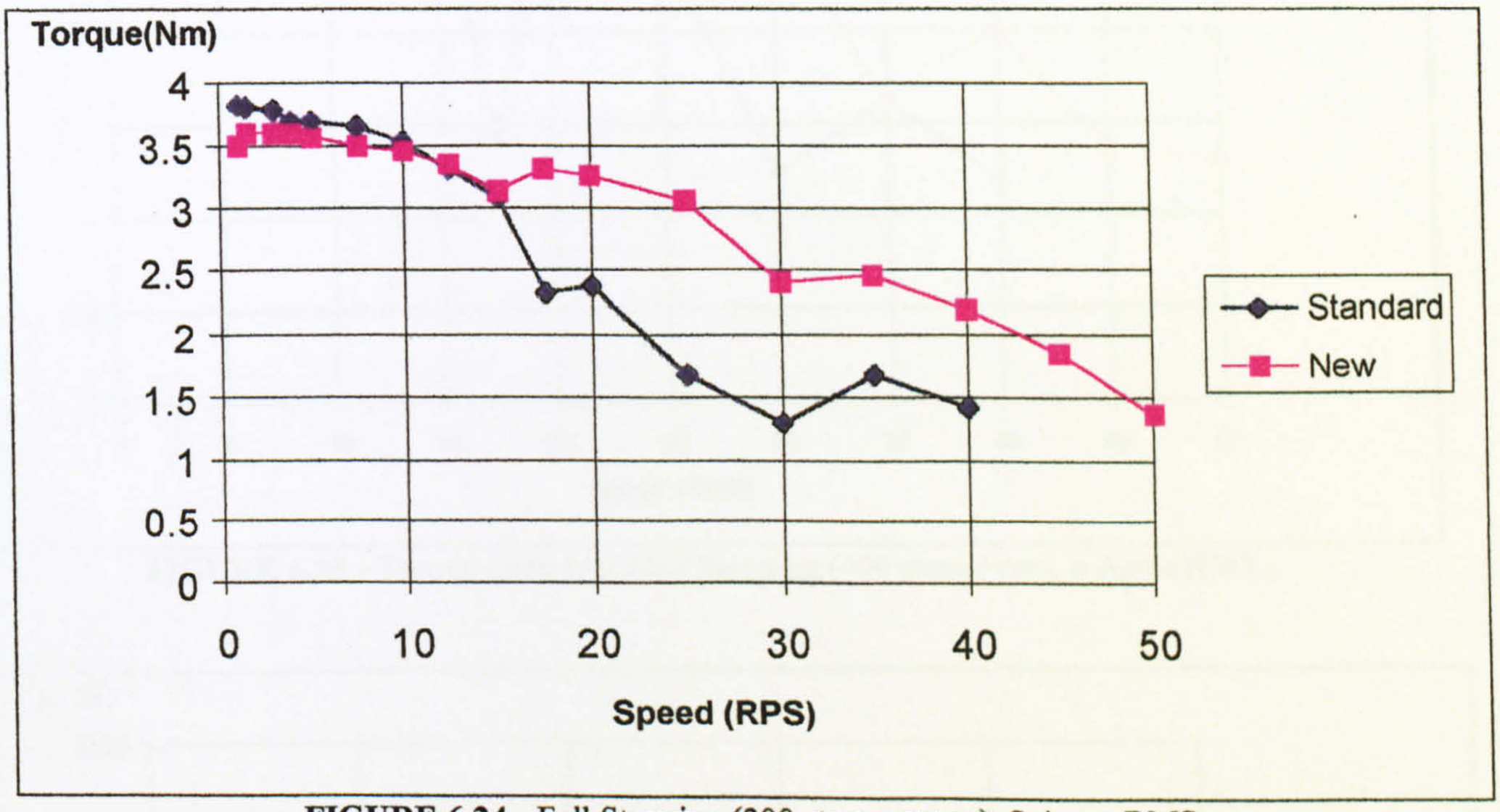


FIGURE 6.24 - Full Stepping (200 steps per rev), 8 Amps RMS.

Figure 6.25 shows the data recorded during half stepping. Most industrial applications make use of this stepping mode. There is a slight loss of torque production, but this mode typically results in much better low speed running, reduced overshoot, and less ringing at the end of each step. Performance in this mode therefore is a good industrial performance measurement. The mid speed resonance in the standard motor at 17.5 rps would cause some applications to stall. The torque dip of the standard motor would be from around 3 Nm to the measured value of around 1.5 Nm, a 50% loss in torque. This same resonance,

still present in the new design, results in a drop from 3 Nm to 2.5 Nm. Some resonance occurs in both motors at 30 rps. The power out curve is recorded in figure 6.26.

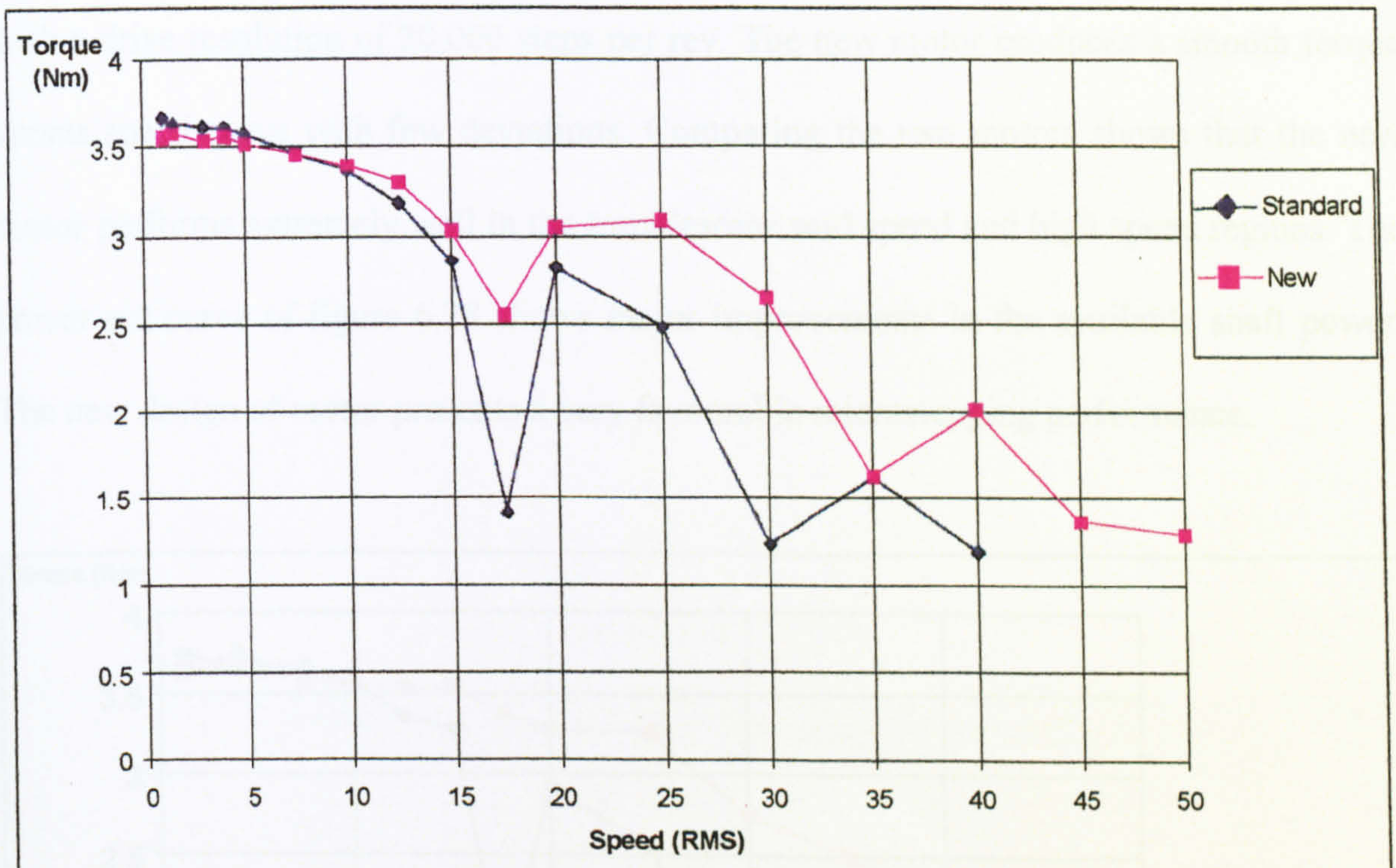


FIGURE 6.25 - Torque Output in Half Stepping (400 steps / rev), 8 Amps RMS.,

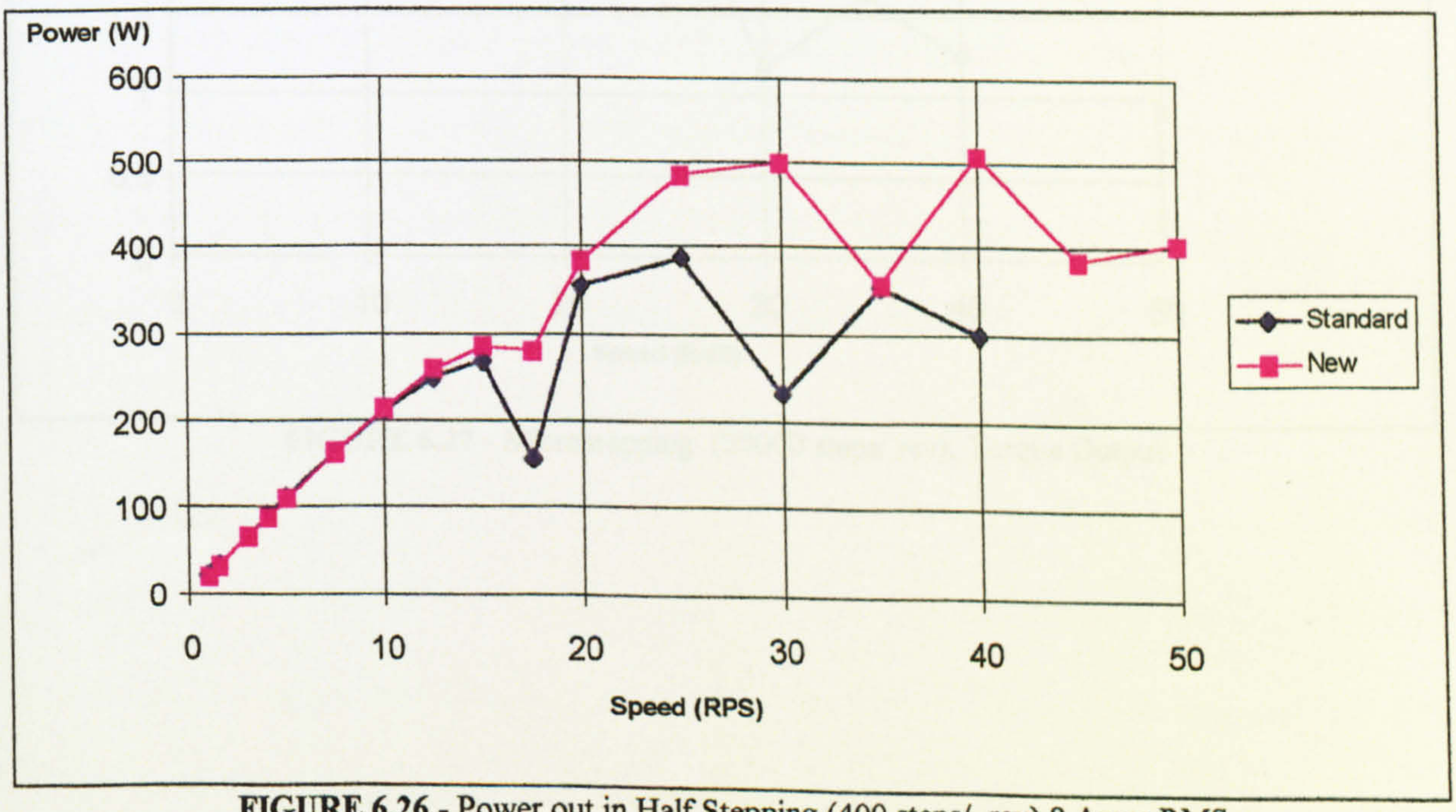


FIGURE 6.26 - Power out in Half Stepping (400 steps/ rev) 8 Amps RMS

Microstepping attempts to produce smoother rotation, less torque ripple, resistance to resonance, and an increased step resolution. The new motor has been found to have an excellent performance with high resolution drives. Figure 6.27 shows both motors running with a drive resolution of 20,000 steps per rev. The new motor produces a smooth torque versus speed curve with few deviations. Comparing the two motors shows that the new motor performs extremely well in the troublesome mid speed and high speed regions. The power out curve of figure 6.28 shows major improvements in the available shaft power. The new design of motor presents a very favourable microstepping performance.

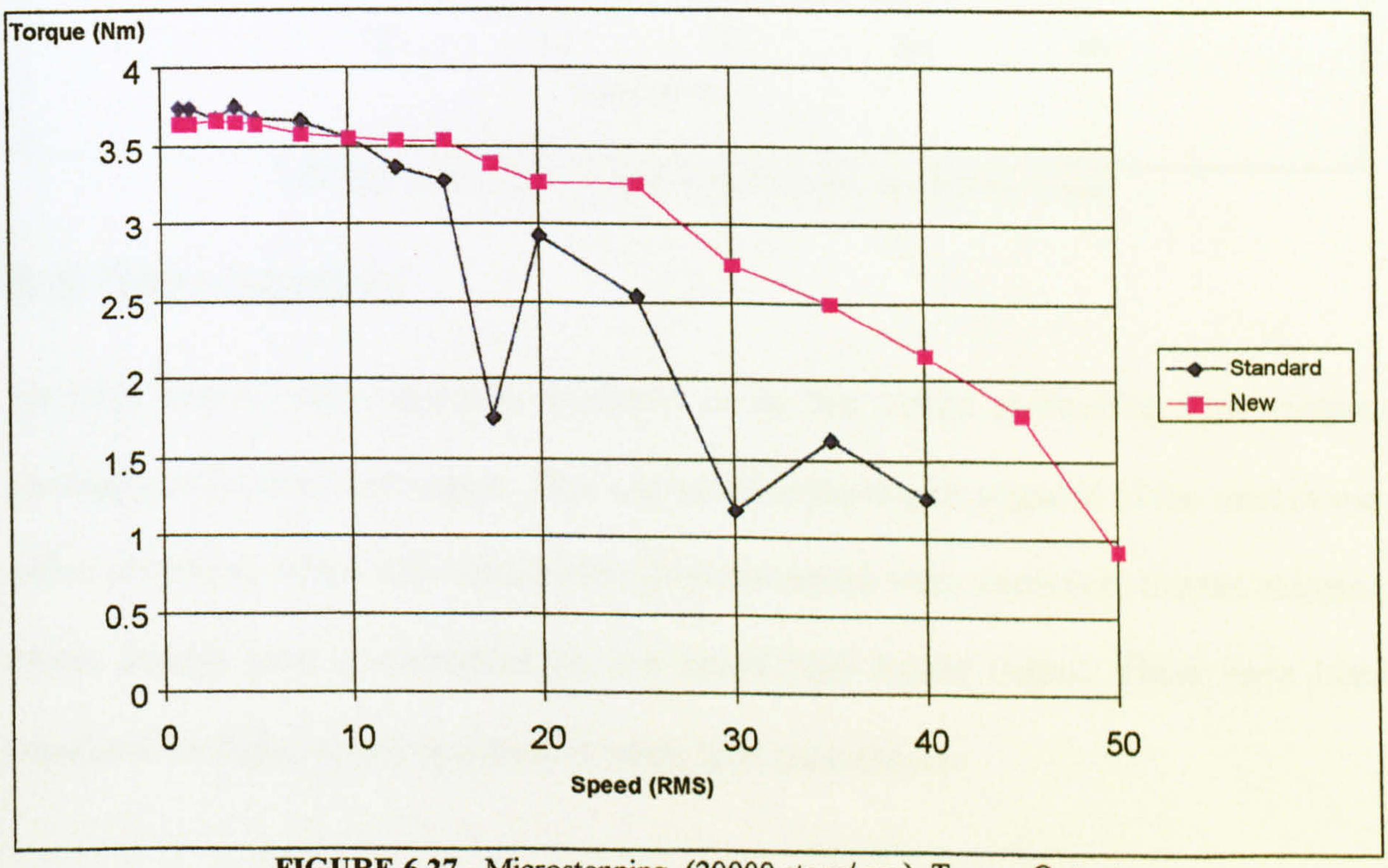


FIGURE 6.27 - Microstepping (20000 steps/ rev), Torque Output

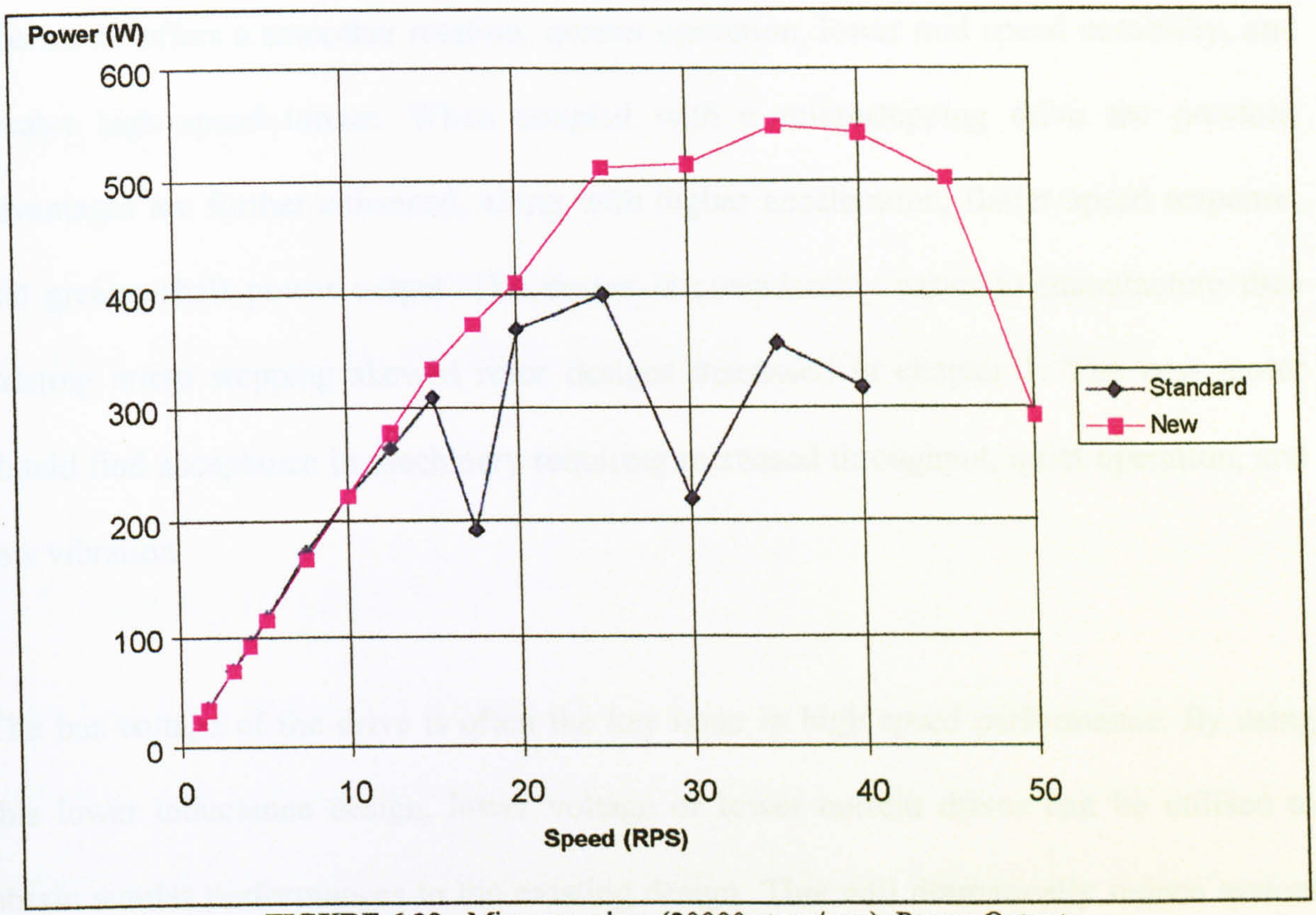


FIGURE 6.28 - Microstepping (20000 steps/ rev), Power Output

6.9 Discussion

Stepping motors were originally designed to be low speed positioning type motors, operating in the stop/ start region. This was because the largest segment of the market was office machinery where slow speed point to point moves were necessary. Recent stepping motor designs have concentrated on low speed high torque output. These have been penalised for higher speed operation by their high inductances.

A large sector of industrial machinery requires higher throughput, with increased torque for higher speed point to point moves. This market has increasingly been attacked by brushed DC servo motors. The stepping motor described in this chapter does not cost any more in material and yet offers an improved shaft power output, suitable for such jobs.

The motor offers a smoother rotation, quieter operation, lower mid speed instability, and greater high speed torque. When coupled with a microstepping drive the previous advantages are further enhanced, along with higher acceleration, flatter speed response, and greater shaft power output. The design is considerably easier to manufacture than existing micro stepping skewed rotor designs discussed in chapter 1. The new motor should find acceptance in machinery requiring increased throughput, quiet operation, and low vibration.

The bus voltage of the drive is often the key issue in high speed performance. By using this lower inductance design, lower voltage or lower current drives can be utilised to obtain similar performances to the existing design. This will dramatically reduce system prices, with little change in the motor. Alternatively copper losses can be reduced by turning down the number of turns in each phase, and increasing the turn gauge, to obtain a similar performance with the added benefits of lower resonance, and smoother and quieter movement.

This motor is planned to go into production with Stebon Ltd. and results have been verified by Parker Digiplan of Poole UK, who also found that the motor had significant improvements in settling time and overshoot.

CHAPTER 7

Application of Soft Magnetic Composite Materials in the Structure of the Hybrid Stepping Motor

7.1 Background

It has been reported that soft magnetic composite materials offer design advantages when compared with steel laminations due to their isotropic nature and the relative ease of producing complex shapes by the use of powder metallurgy methods [73-77]. Iron losses are low at elevated frequencies as the eddy current loss becomes a small proportion of the total output.

Because of the claims about the isotropic nature of powdered iron, this material may provide an improvement to the cross lamination flux patterns and iron losses at high rotational speeds in the hybrid stepping motor. This chapter is a study as to how such materials could be utilised in the hybrid stepping motor.

7.2 Laminations

Magnetic cores used for electric motors are usually made of strips of electric steel called laminations. These laminations are used to suppress the losses related to changing magnetisation. Typically the machines are produced with 0.35, to 0.5 mm laminations. The motors in this project have used 0.35 mm laminations. The material used is typically

a silicon iron. In motors with radial flux paths only, the higher the frequencies, the thinner the laminations that are used to reduce eddy current losses. Laminations are insulated one from another with coatings of non-conducting paint, on one or both sides. Alternatively, the lamination surface insulation can be a natural oxidation layer, an organic enamel, or an inorganic surface treatment. This breaks the continuity of the eddy-current-paths when the flux changes with time in a direction parallel to the surface plane of the lamination and reduces the losses. In general, the thinner the laminations, the lower the losses, since the eddy-current losses increase with the thickness of the lamination.

In laminated structures, eddy currents are minimised by constraining the flux to flow parallel to the surface plane of the lamination. Unfortunately in axial flux machines, such as the hybrid stepping motor, where flux is required to cross from one lamination to another, it has to travel through an inter-lamination boundary, figure 7.1. This boundary will act similar to a small air-gap, but with undefined and random dimensions. Therefore there will be an MMF drop at each inter lamination air-gap reducing flux and hence torque. Consequently if the areas of the motor that tend to experience cross lamination flux are replaced by an isotropic material, the MMF drop across the laminations could be reduced.

Electrical sheet steels have been the overwhelming choice for the soft iron structure in electrical machines, subject to time varying magnetic frequencies. Stebon Ltd. use TRANSIL a steel which was selected to reduce losses at higher frequencies. The grade of steel and lamination thickness is selected to reduce losses associated with its operating frequency and eddy currents. For applications (for example aerospace) where the

fundamental frequencies of operation may be high, laminations of 0.1 mm thickness silicon and cobalt steels may be found. As stated the hybrid stepper motor may operate with frequencies of over 1 KHz, where the iron losses become a large order of magnitude. Reducing the thickness of the lamination is not a viable option because of the losses associated with cross lamination fluxes. This is where the use of soft magnetic composites becomes attractive.

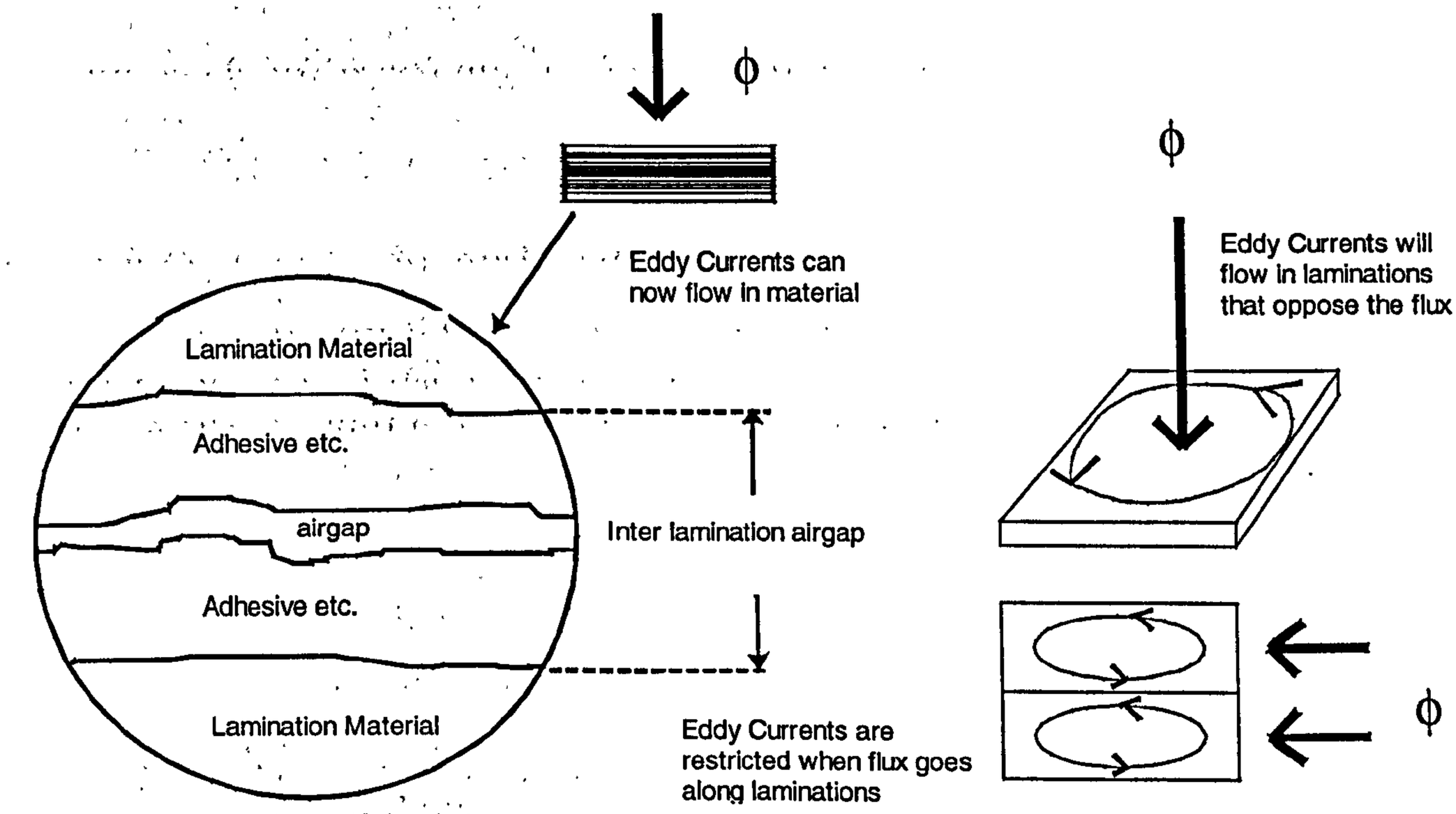


FIGURE 7.1 - Close-up Representation of Two Adjacent Laminations

7.3 Soft Magnetic Composite Materials

Soft magnetic composite materials (SMC), also commonly known as powdered irons, consist of metal powder particles of pure iron. The use of pure iron instead of alloying elements offers the highest saturation induction outside cobalt/ iron composites. Alloying elements are often introduced to minimise eddy current loss. In the case of soft magnetic materials, an insulating layer on the surface of the iron particle reduces the eddy current

loss in all directions. To obtain a high flux density at low field strengths, high compressibility iron powder is used with a high compacting pressure, in an attempt to achieve this. Production techniques require the addition of lubricants and resin to enhance the strength of the material. These therefore degrade the permeability and make the material less suitable for low frequency applications.

7.4 Magnetic Properties of Soft Magnetic Composites

7.4.1 Iron Loss

If the material is used at low frequency it is only competitive with low grade lamination steels. However at frequencies higher than 50 Hz, the laminated steel's performance is reduced, because of elevated frequency dependent losses. This makes the stepping motor an attractive option as it naturally operates at high electrical frequencies. However there is still a concern because of the relatively low unsaturated permeability. Figure 7.2 shows the B-H curve differences between a high grade steel and a high grade powdered iron. This curve is for a soft composite material that does not contain strengthening resin and machining lubricants.

7.4.2 Unsaturated Permeability

The unsaturated permeability is not as high as typical magnetic steels, and is approximately two thirds that of the magnetic steel. This can become a particular problem with small machines, which are magnetised from the armature, and where a large coil

current is found. This suggests that the material may only be suited for larger machines, where there is more flexibility for the choice of electric and magnetic loading.

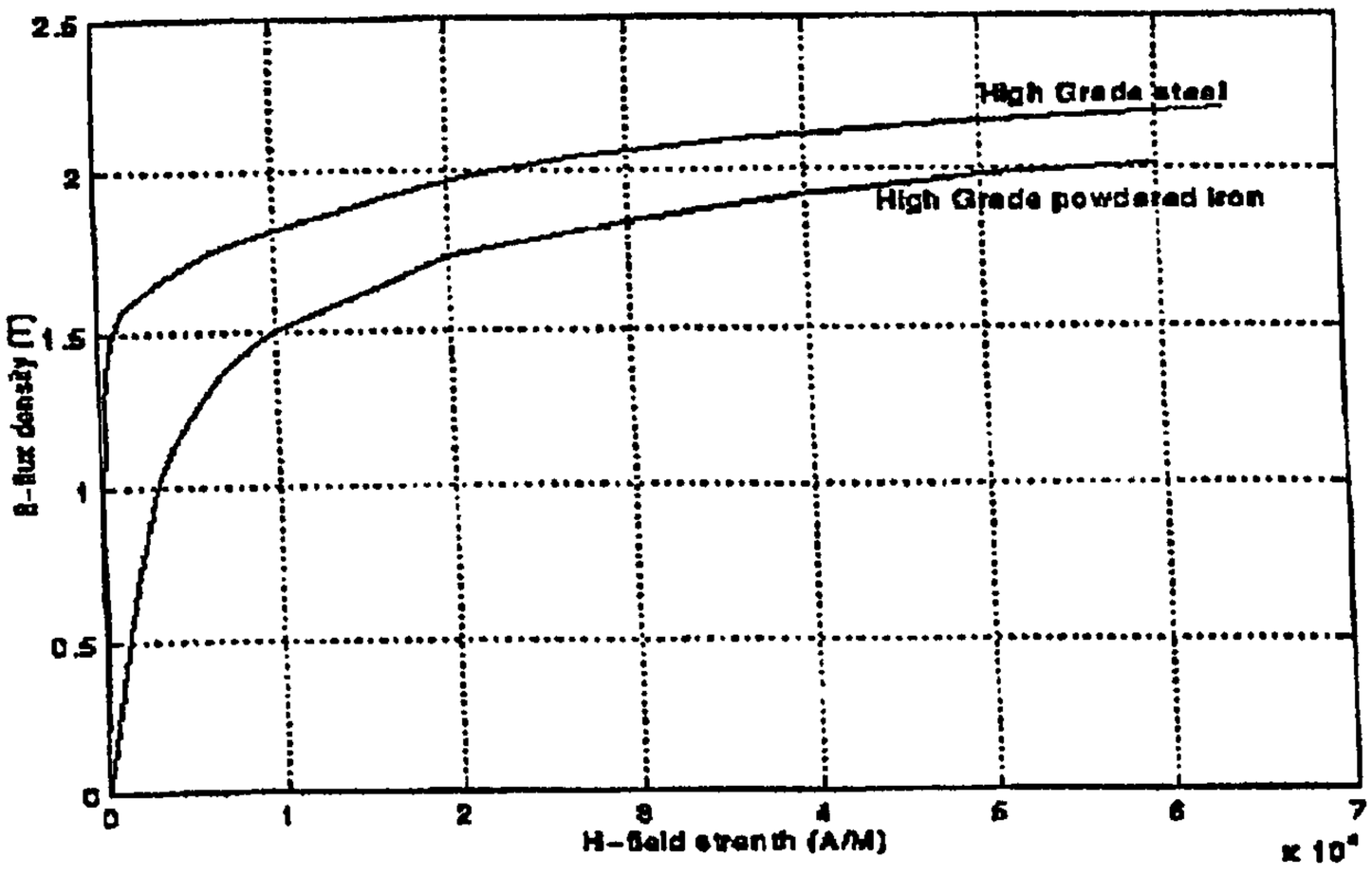


FIGURE 7.2 - Comparisons between the B-H curves of High Grade Powder Iron and Silicon Steels

7.4.3 Isotropy

As the material behaves equally in all directions, it is known as isotropic. This is important both thermally and magnetically. The bulk thermal capacity of the composite material is similar to laminated steel, in all directions, whereas the laminated steel has a greater thermal capacity in the plane of the lamination. This allows the heat transfer in a powdered iron to be isotropic, aiding cooling in all directions in the core.

The isotropic magnetic characteristics are more important. The machine types which appear to justify powdered iron consideration are those with three dimensional magnetic flux structures and those with high operating frequencies. Losses associated with field variations normal to the plane of steel laminations (combined with the back EMF effect) limit the maximum speed of a motor. With powdered iron the improvement of torque per unit volume has good potential, particularly at elevated flux frequencies.

7.5 Improving Characteristics of the Hybrid Stepping Motor with SMC materials

The iron losses of the stepping motor could be reduced for the higher frequencies of excitation. However at low speeds this is offset by the lower permeability of the materials, reducing the flux. The hybrid stepping motor which has the unusual structure of an axially magnetised magnet, placed within its centre core, causes the flux to cross laminations in the z-direction. As the motor has significant radial, axial and tangential components of flux, the motor is a truly three dimensional machine. If the reduced losses at high frequency are sufficient to overcome the lower permeability, a better high speed stepping motor may be produced. This all depends on the economics, ease of construction, and robustness of the materials to be used.

7.6 Analysis of Hybrid Stepping motors Using Finite Element Software

By using the static three dimensional finite element software, an investigation into the positive uses of soft material composites may be established. However three dimensional finite element software produces static analysis of the hybrid stepping motor. Unfortunately, dynamic non-linear finite element analysis of rotary machines would be extremely computationally expensive for each possible motor design. However if improvements in the static performance could be made by altering the machine design with powdered iron, benefits obtained in dynamic performance would naturally transpire. The following results are for a Stebon 1101 type motor.

7.6.1 Total Replacement of Steel by SMC Materials

The simplest investigation is to totally replace all the motor steel with SMC material. This involves switching from an anisotropic material with a packing factor to an isotropic type with the revised magnetisation characteristic. Table 7.1 shows the results of switching the steel to a SMC type P1171 manufactured by SMP [Appendices]. Referring to Table 7.1, it can be seen that at higher currents the torque is higher when using the laminated steels due to the increased permeability. At low current levels the SMC material produces better torque. The isotropic nature of the powdered iron is seen to be beneficial. With two phase excitation there is more flux flowing through the back iron, again the reduced permeability of the SMC is seen. This causes the difference in torque at low excitation to be of a lower margin. The motor is unlikely to run at low current levels and in single phase, and this makes SMC unsuitable to wholly replace the lamination steel. There may be some indication that changing either the rotor or stator by the soft magnetic composite, may produce improved results, however because the teeth may be operating with a flux density of around two Tesla, they would quickly saturate.

Some parts of the motor may be successfully replaced by SMC. The ideal locations are where the flux density does reach over 1.5 Tesla and has a high component of axial flux crossing the laminations normal to the surface plane. Places of low flux-density in the size 42 motor are the back iron ring and the centre region of the stator above the air-gap. The latter is because the flux does not fully fringe to the centre as described in figure 7.3. This effect particularly noticeable when the magnet length is a large percentage of the stator stack length.

	SINGLE PHASE TORQUE (Nm)	TWO PHASE TORQUE (Nm)
3.5 Amps - Silicon Steel Laminations	1.494	2.723
3.5 Amps - SMC	1.582	2.755
7 Amps - Silicon Steel Laminations	2.415	4.647
7 Amps - SMC	2.133	4.028

Table 7.1 -Laminated structure versus replacement by Powdered Iron for 1101 Motor

7.6.2 Partial Replacement of Steel by SMC Materials

Areas where the flux flows in an axial direction are found mainly in the rotor end-caps and in the centre of the stator core. Even though the flux density by be relatively low in this region, the MMF drop in these regions may be considerable as the flux has to travel across the laminations. Further research was undertaken to change the centre piece of the stator to SMC.

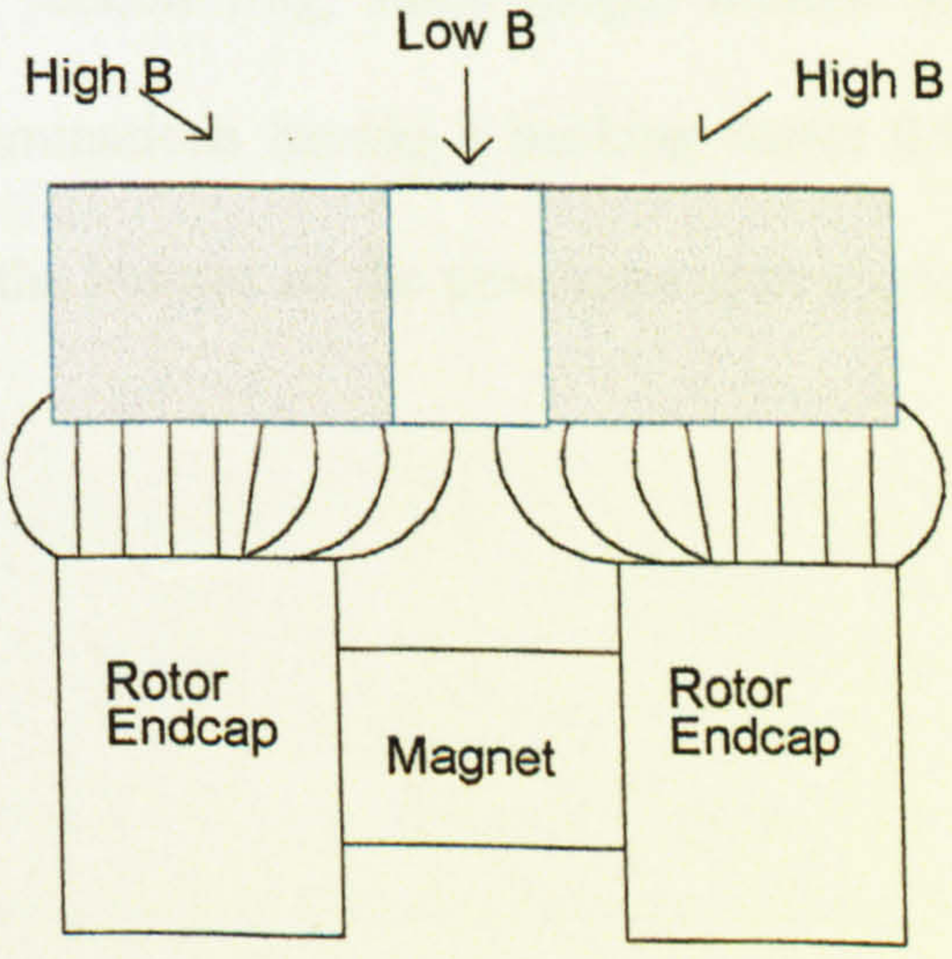


FIGURE 7.3 - Low area of Flux Density in the Stator due to rotor Fringing

7.6.3 Using a SMC Ring to Replace the Central Stator Laminations

If the central section of the motor was considered as a single section of the core pack, it may prove easy to replace this part with a 'thick' lamination of the new soft magnetic composite material. This would the aid the manufacturing process of the complete motor. Unfortunately the material to be considered would have to machined from standard slugs, if any complex design was to be included. This would include teeth. In a complete manufacturing process the design would be formed by pressing the SMC material.

Several designs were investigated to see the practicality of using a central ring of the SMC material in the centre of the stator core. These were based around the original dimensions of the Stebon 1101 motor. The stator pole representation is based upon figure 7.4.

Design 1; Stator central section ring, stack length section of 10 mm, which is equal to the magnet length. Packing factor on remaining stator laminations being 0.98. The geometry of the ring being identical to the standard steel lamination in the X-Y plane.

Design 2; Stator central section ring, stack length middle section of 10 mm width. Remaining steel stator laminations having a packing factor 0.98. Saliency is added by removing the teeth from the bottom of the powdered iron section. The triangular section above is still included.

Design 3; Stator central section ring, triangular pole section removed to achieve a straight pole similar to switched reluctance lamination. Width of section again 10 mm. Packing factor 0.98 for remaining steel laminations.

Design 4; Stator central section ring. Pole section removed. Only back iron ring kept. Again with 10 mm width, and remaining stator lamination packing factor of 0.98.

Design 5; Stator central section ring; As design 1 but with stack of length middle section increased to 20 mm width. Packing factor 0.98.

Design 6; All stator back iron and teeth of central section replaced with SMC; Section length of SMC being 10 mm. Steel laminations remaining being of packing factor 0.98. No castellations. Back iron throughout the stator is constructed of soft magnetic composite materials. Design difficult to construct.

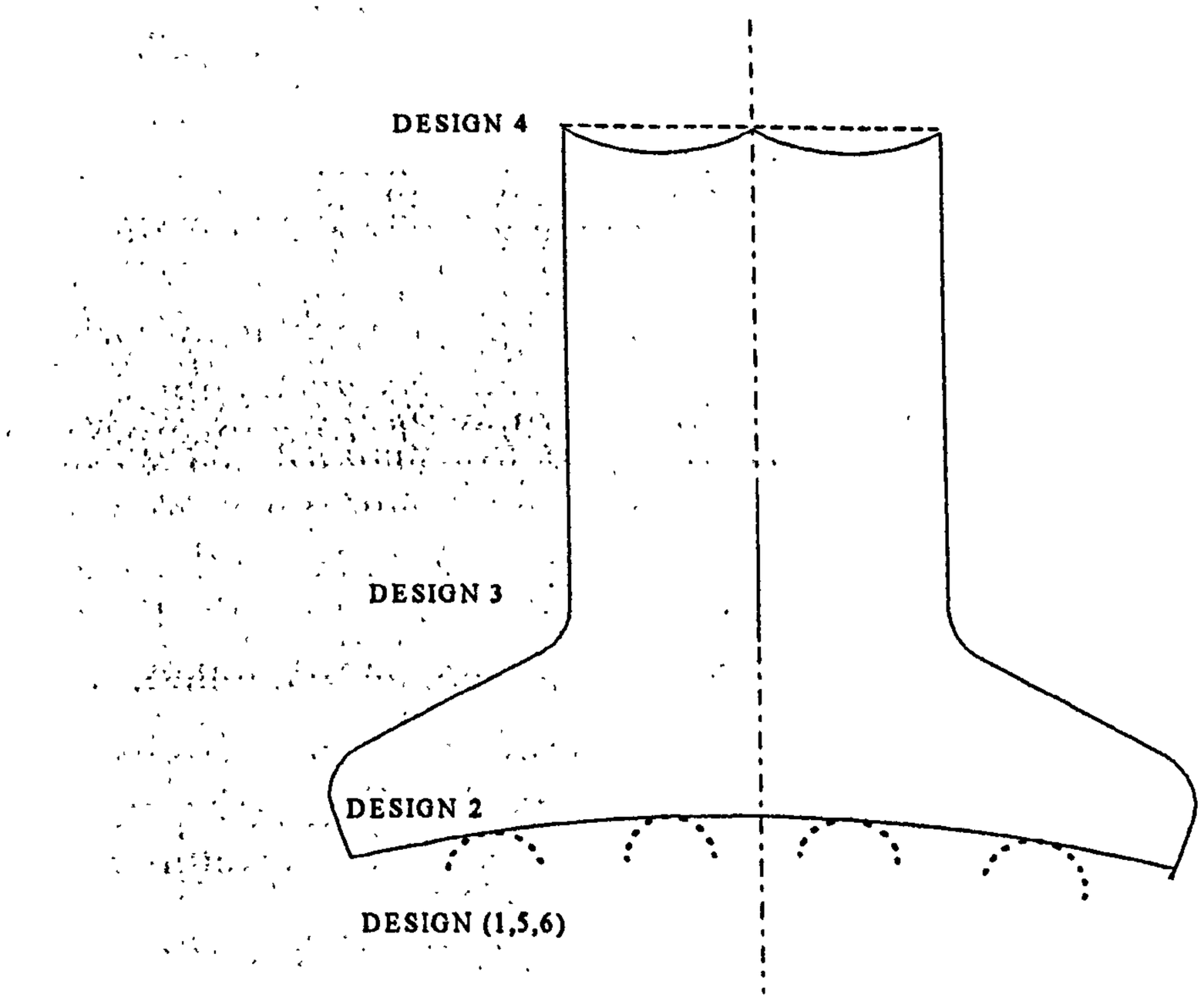


FIGURE 7.4 - New Pole Design for Powdered Iron Ring.
Note the position of the teeth arcs on the steel laminations.

The results for static torque are presented in table 7.2. The standard reference motor having a stator of steel laminations with a packing factor of 0.98.

The results show that with the exception of Design 6 most of the designs perform well at low excitation in a single phase capacity. However as the single phase excitation level increases only design 1 and 2 offer real improvements. Design 1 is aided by the central teeth not being pushed to higher levels of flux density in the central region of the motor's stator. Design 2 forces the flux to travel in an X-Y direction up into the teeth by adding saliency to the region of low flux-density. At higher levels of flux-density this may penalise design 2. When the excitation system is switched to two phase, again the only designs of promise are 1 and 2. Here at higher excitation, they are on par with the completely laminated steel design.

Design	Single Phase (3.5 Amps)	Single Phase (7 Amps)	Two Phase (3.5 Amps)	Two Phase (7 Amps)
Standard	1.494	2.415	2.723	4.647
Design 1	1.606	2.463	2.831	4.649
Design 2	1.613	2.535	2.867	4.652
Design 3	1.483	2.19	2.75	4.134
Design 4	1.433	2.028	2.434	3.772
Design 5	1.538	2.342	2.618	4.39
Design 6	1.18	1.690	1.759	2.193

Table 7.2 - Torque Results for Powdered Iron Sections Investigation

Having established that designs 1 and 2 offer the best static performance in relation to the standard design, the option of which type to further research has to be made. The two dominant factors are the strength of the structure, and the flux density limits of the material.

As the excitation rises, the fringing flux from the rotor end-caps travelling into the centre of the stator increases. This will eventually degrade the performance of design 2, below that of design 1. However within the normal operating range of the motor (7 Amps), a difference in performance would be minimal.

The soft magnetic materials have a notorious history of being extremely brittle. Though the addition of resins and lubricants aid machine processes they also degrade magnetic performance. With the powdered iron positioned between two packs of steel lamination they would act as a form of mechanical shield. However there is doubt about the ability of a set of SMC teeth on the stator to withstand honing. Therefore because of this, and the simplified structure of design 2, it seemed more suitable to develop this. A three dimensional model of design 2 is shown in figure 7.5.

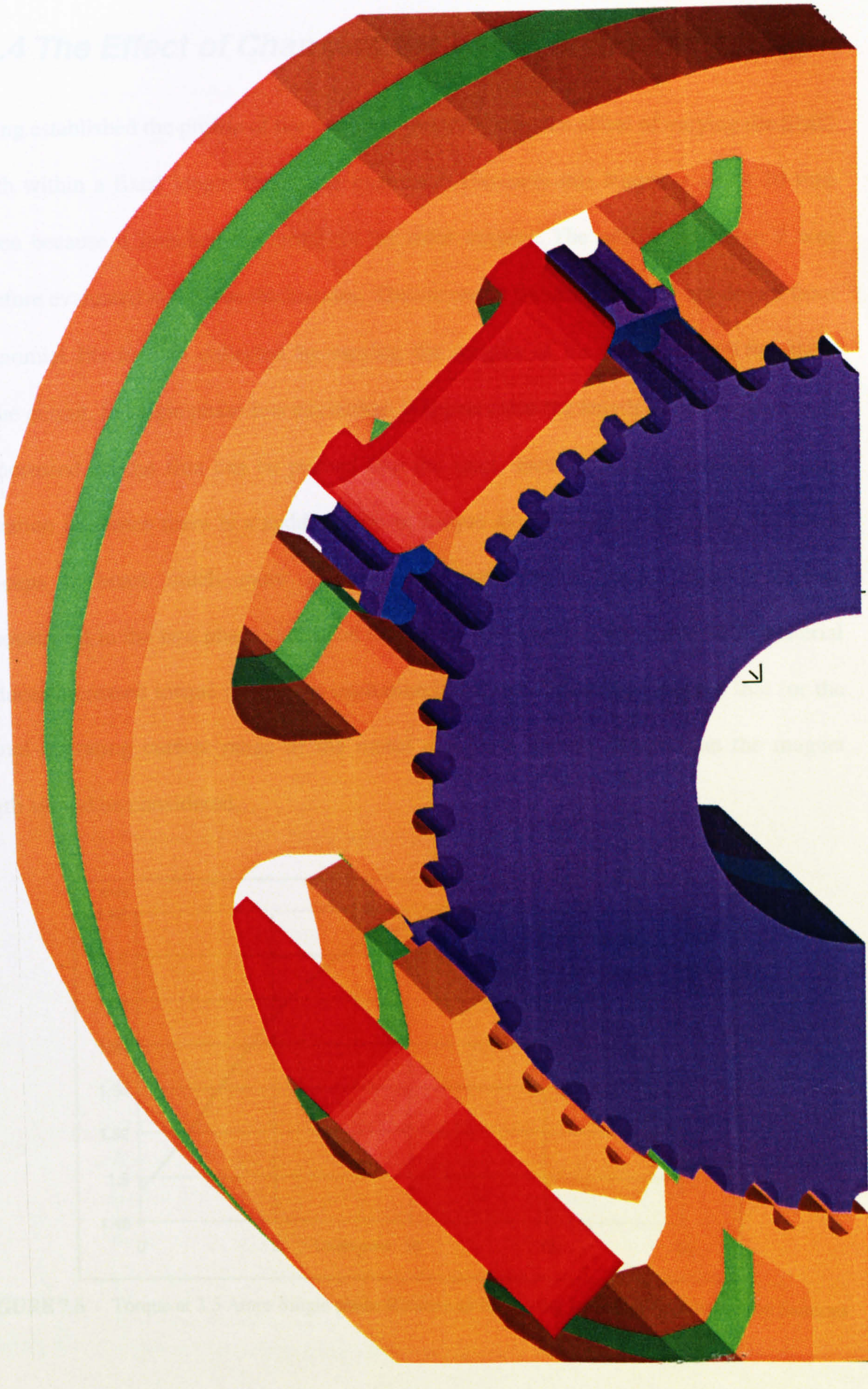


FIGURE 7.5 - Design 2 Ring Position in Stator

7.6.4 The Effect of Changing the Width of the SMC Ring

Having established the profile of the design to proceed with, the effect of varying the SMC length within a fixed length stator was examined. Presently this length is set at 10 mm chosen because it matched the length of the rotor magnet. The profile of design 2 was therefore evaluated at various thicknesses. The curves of figures 7.6 to 7.9 are drawn from polynomial fits to data collected by varying the lengths of the SMC to find the static torque output. In single phase low excitation, it appears that the length of the section could be increased, as the flux can be pushed into the steel teeth, without saturating. As the excitation increases there is a sudden drop of torque as the fringing flux from the rotor end-caps becomes much more abundant and saturates the iron. This is further demonstrated in the two phase excitation scheme as the length at which the SMC material produces increased torque reduces as excitation increases. It therefore appears that for the normal operating current range of the motor a slightly smaller length than the magnet length should be considered.

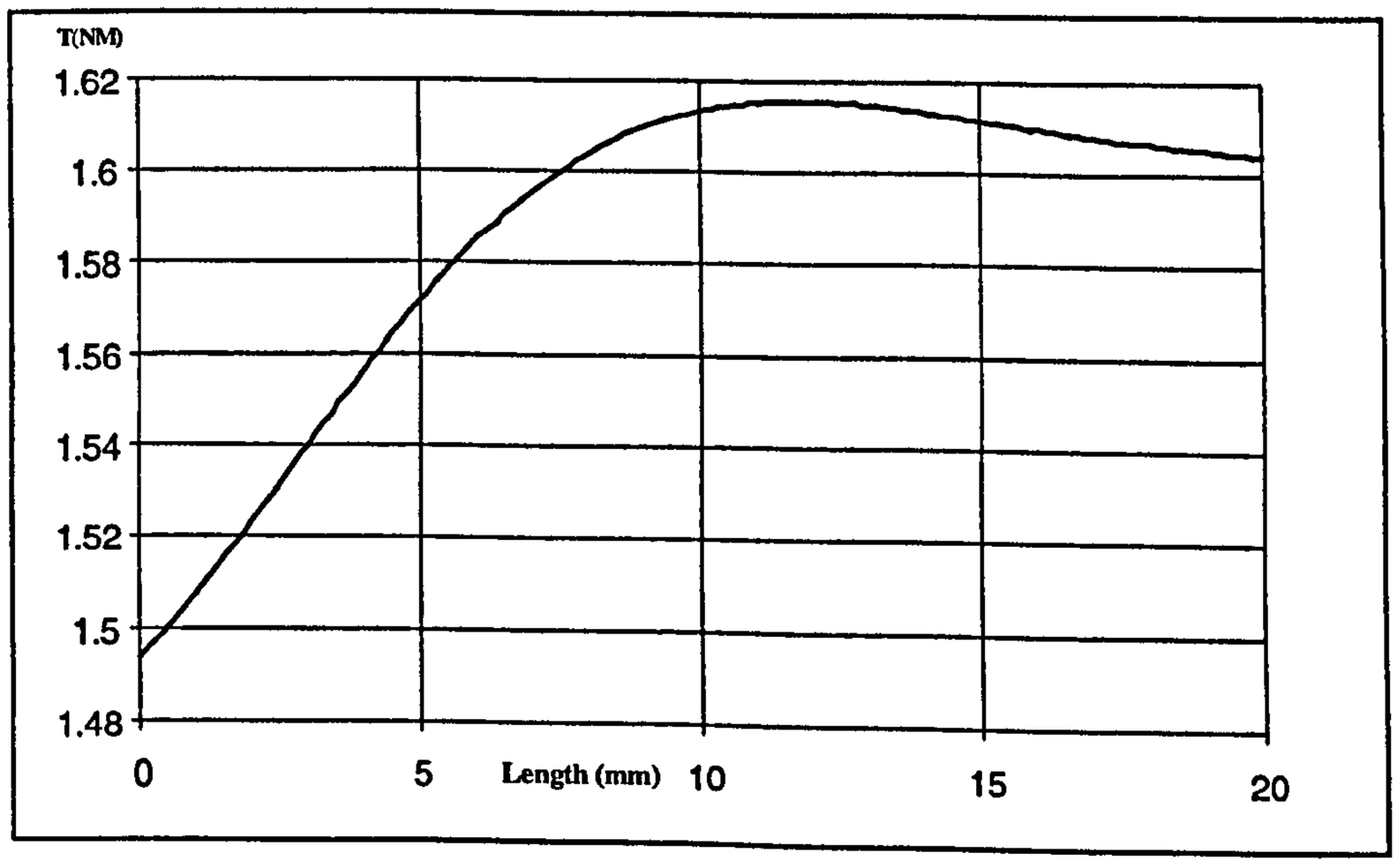


FIGURE 7.6 - Torque at 3.5 Amps Single phase Relative to Increasing Length of Powdered Iron Sections

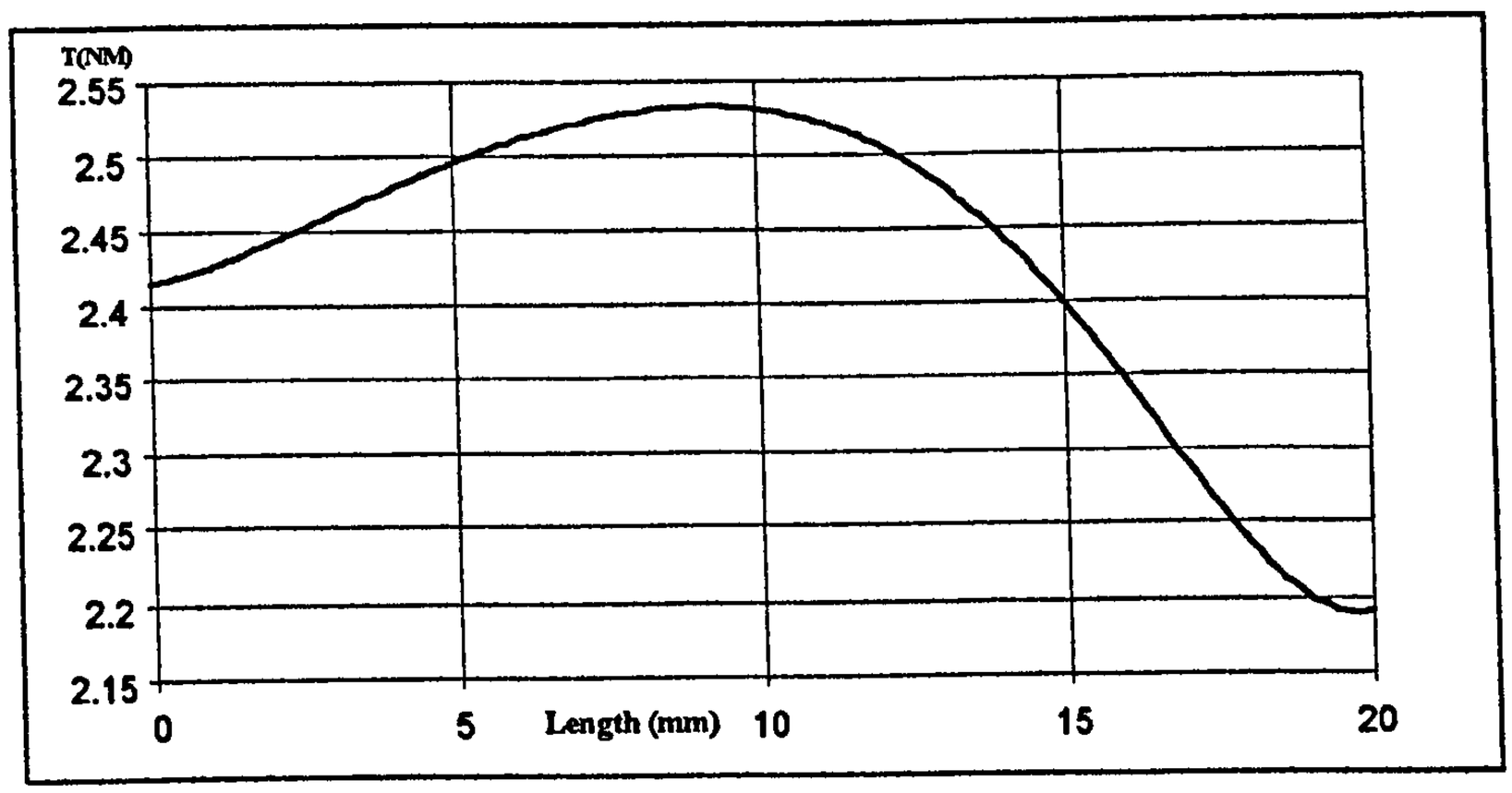


FIGURE 7.7 - Torque at 7 Amps Single phase Relative to Increasing Length of Powdered Iron Sections

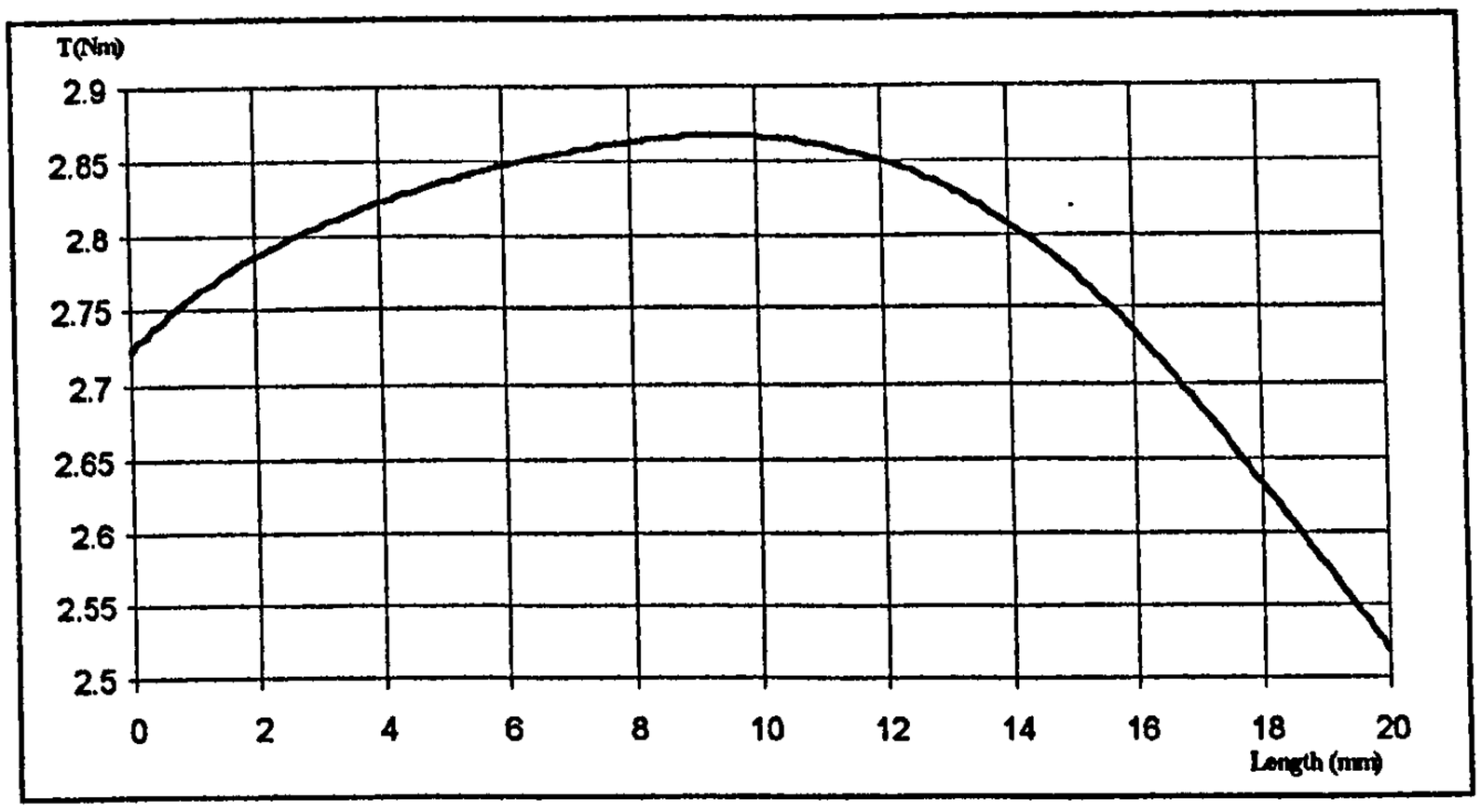


FIGURE 7.8 - Torque at 3.5 Amps Two phase Relative to Increasing Length of Powdered Iron Sections

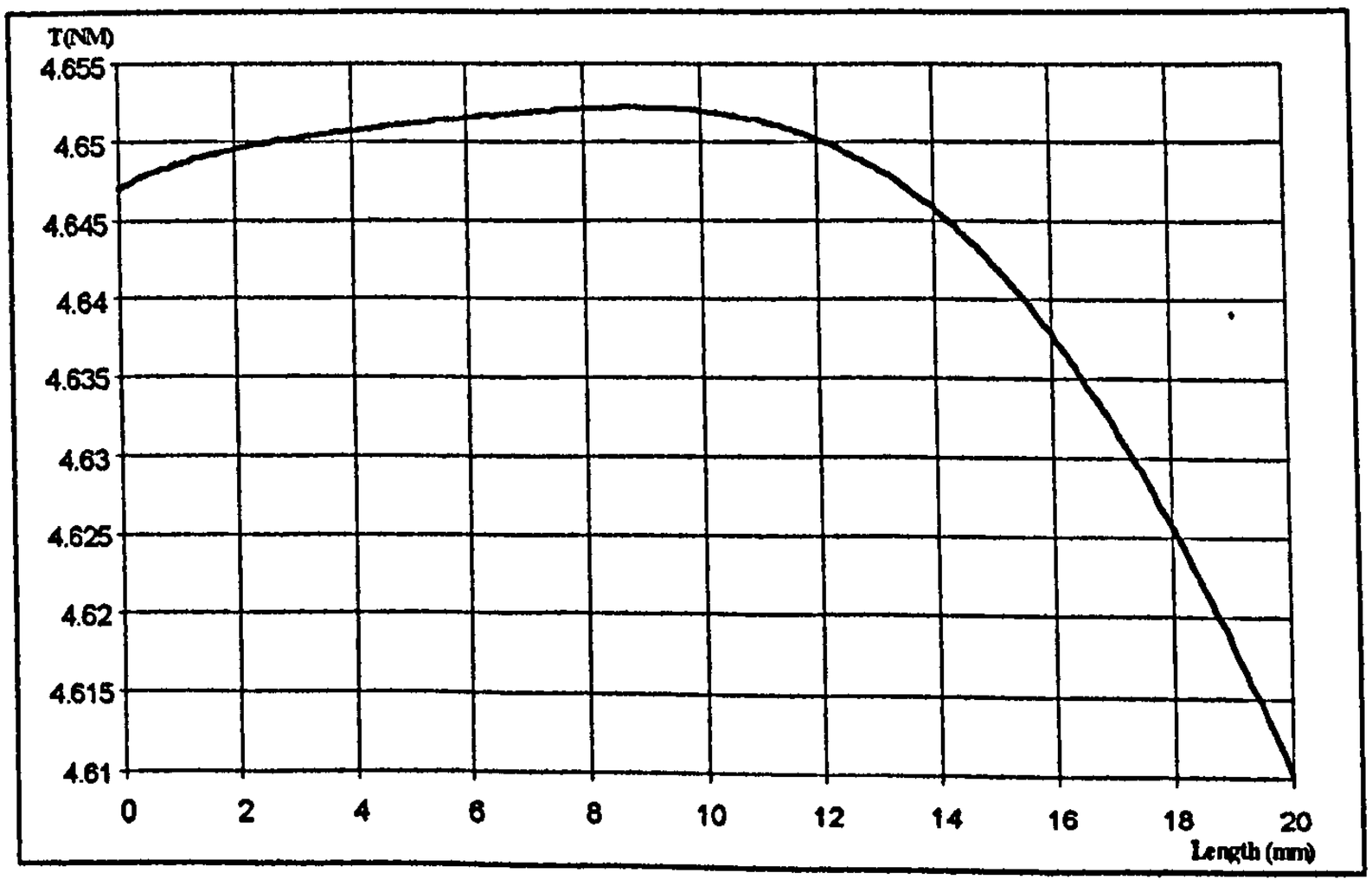


FIGURE 7.9 - Torque at 7 Amps Two phase Relative to Increasing Length of Powdered Iron Sections

7.6.5 Dynamic Advantages

The study presented has been on the static performance of the motor. The aim was to find the geometry of the SMC material which optimised the static performance despite the lower permeability of the SMC. The other key elements described previously were concerned with dynamic properties, such as improved thermal conductance and lower losses at higher frequencies. The finite element software used in the project unfortunately cannot model these dynamic effects, and thus the effects of reducing or increasing the section length cannot be found without experimental methods or new software. However it is anticipated that any improvements demonstrated in static conditions will be further enhanced under dynamic conditions.

7.7 Building a Stepping Motor with a Soft Magnetic Composite Section

There are only a few manufacturers of soft magnetic composite materials in the world. In Europe the two main suppliers are SMP of Germany, and Höganäs AB of Sweden [Appendicies]. SMP material data was used in the previous finite element analysis. Neither company could supply a slug the size of the outer diameter of the size 42 frame. Höganäs AB indicated that within the near future this slug would be available, when they moved to a new factory site. Höganäs AB could supply a soft magnetic material composite slug that could be machined for a size 34 motor. The decision was made to build a size 34 frame motor based on the same design as design 2, in an attempt to understand some of the dynamic effects of the new material. The details of this material are found the in the appendices.

The next stage was to obtain the slug from the Swedish plant. After obtaining the slug, it would need to be wire eroded to the appropriate shape. A company was found and the material was sent with the relevant drawing. The material was returned and unfortunately had not been packed well. The result was that the sections were broken, due to the brittle nature of the material. After trying to repair the sections it was decided that the process would have to begin again. Fortunately Höganäs AB were willing to supply the slug again and it was wire eroded again. However a heavy time penalty was incurred.

7.7.1 Revising the Design for a Size 34 Frame Motor

The plan was to build four 34-frame single stack motors. Again the lengths of the SMC sections were to be based around the magnet length. The basic dimensions were as follows: The rotor end-caps are of length 12 mm. The magnet being nominally 5.7 mm. With a stator core pack length of 30 mm. The lengths for the SMC material to be evaluated are 4, 5.7 and 10.5 mm. These represent a loss of 13, 19, and 35% of the effective tooth area from the stator core pack.

The motors were to be constructed to the same style as the larger 42 frame, design 2. However in the smaller motor the lower permeability may have a more significant effect, since there was a similar coil MMF in both the larger and smaller motors. Also because the SMC had a lubricant added for machining, the magnetic properties were further reduced.

At the same time as the smaller size 34 motors were being built, a new finite element model was created. This involved creating an 850 model from the beginning and solving for the useful average two phase torque prediction.

7.8 Experimental Results

The following results have been taken using a CD60 [3] drive operating under half stepping with a 85V DC bus voltage. The motors have been wound in parallel to allow testing at higher rotor speeds, where the benefits of the powdered iron should be more noticeable.

7.8.1 Normal Operating Current Results

The motors are rated at 4.5 Amps RMS two phase on. The drive was set to be just above this rating at 4.9 Amps. The results are shown in figure 7.10. All the motors produced lower torque than the standard laminated design at low speeds. The machine with the smallest length of SMC (4 mm), matches the torque output of the standard machine at around 12 rps. It then follows a slightly higher profile for the rest of the range. The motor with the middle length of material (5.7 mm), has a much flatter low speed curve, albeit with a decreased power out. At half the speed the motor produces power out which is equal to the standard motor and this continues to climb for the rest of the speed range. The motor with the largest section of powdered iron, produces far less torque, this is inherently due to loss of holding torque from the removal of teeth. From the power curve of figure 7.13 it does exhibit a very linear response. However this machine does exhibit the greatest power out at high speeds. The peak power out of this motor appears to be above 50 rps.

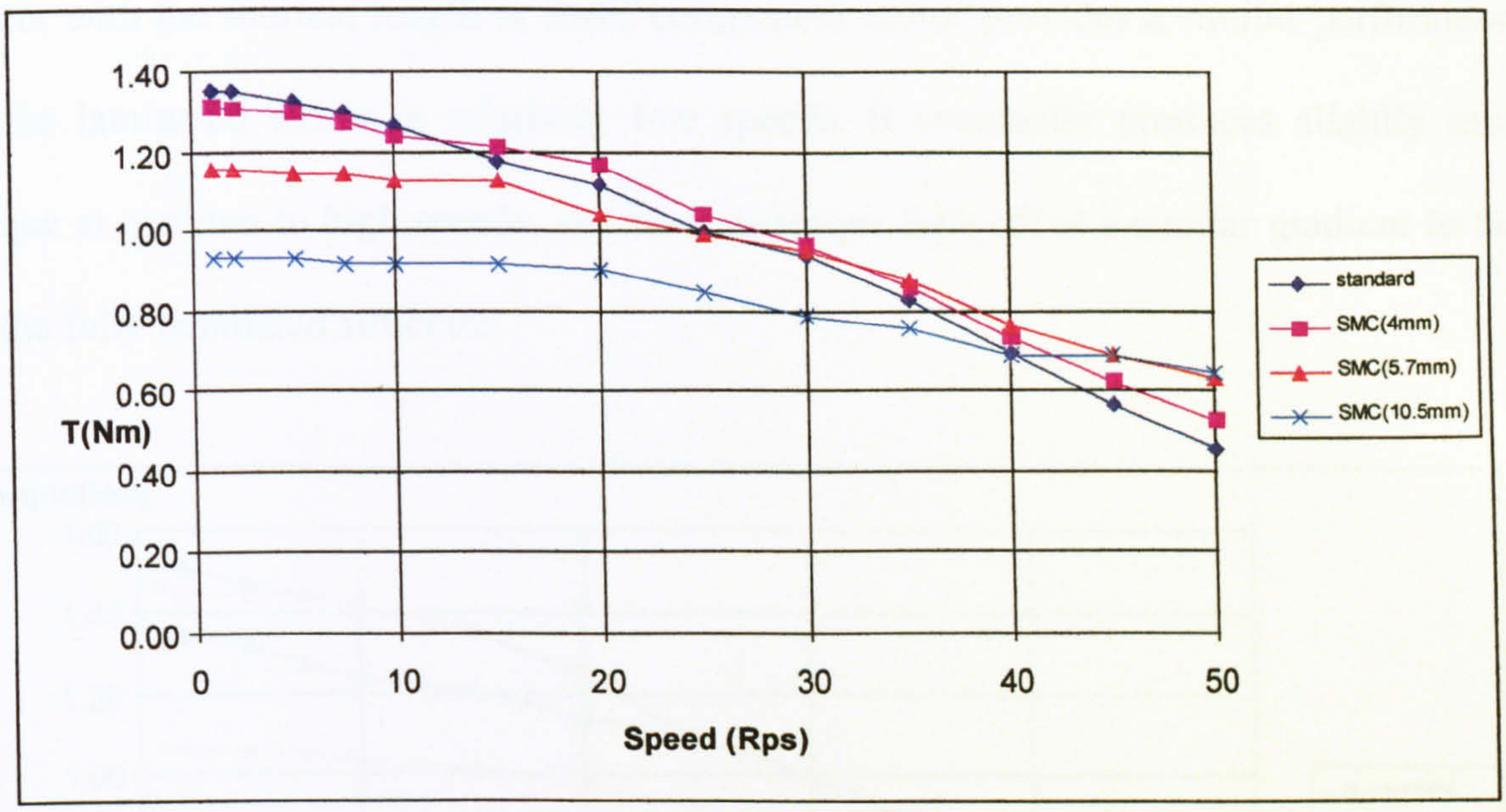


FIGURE 7.10 - Torque Speed for 4.9 Amps Excitation

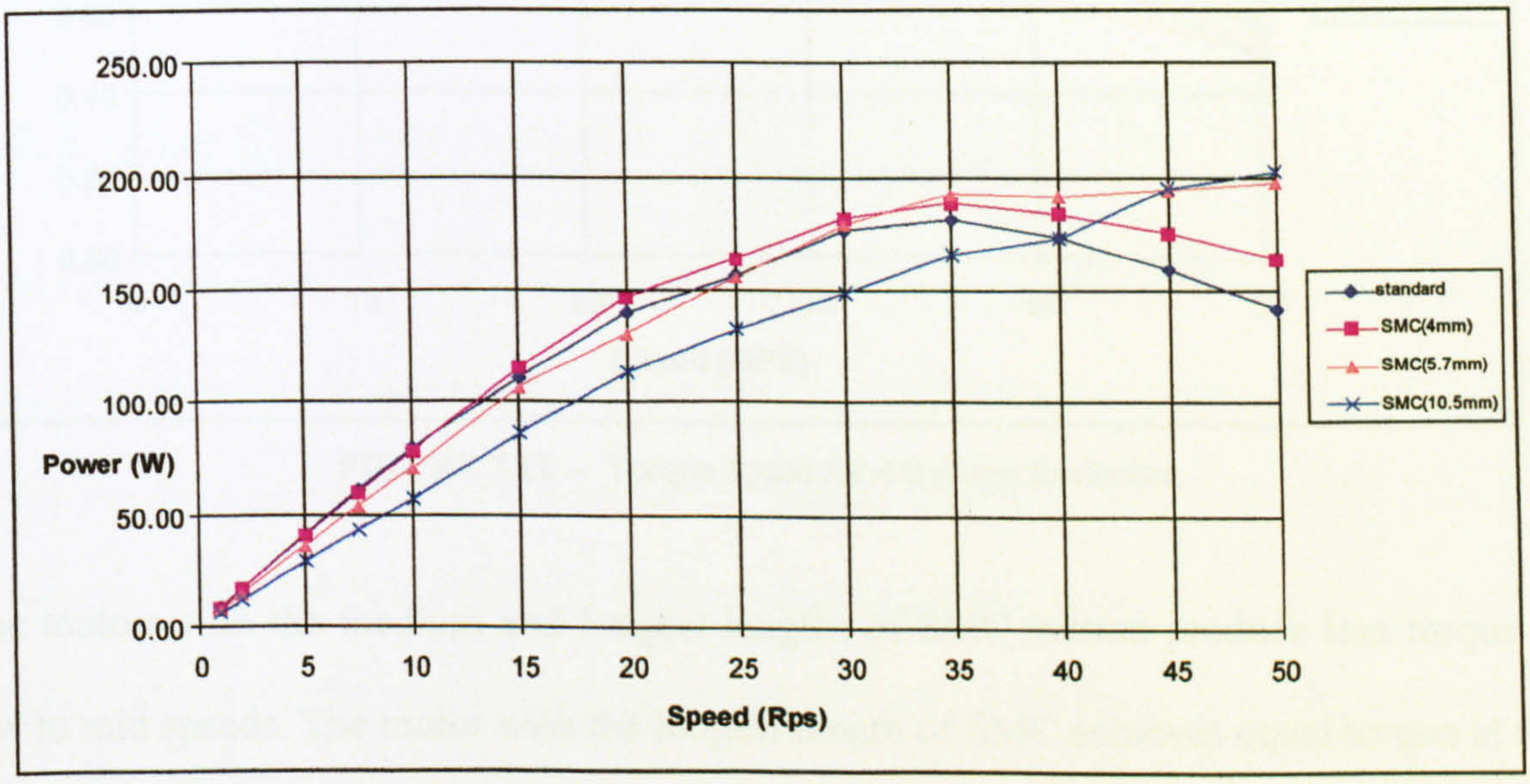


FIGURE 7.11 - Power for 4.9 Amps Excitation

7.8.2 High Current Results

In the last test all the motors produced less torque at low speeds than the normal motor. The same result occurs for a higher test current test of 6 Amp RMS (figure 7.12). The Hysteresis loss for the SMC lowers the efficiency of the motors at higher currents and lower speeds. As the speed increases and current is reduced due to the inductance of the winding and the reduced permeability of the SMC material becomes less dominant. The

motor with the shortest length of SMC component motor provides a similar performance to the laminated motor at relatively low speeds. It eventually produces slightly more torque at medium to high speeds, and then its torque falls off at a similar gradient to that of the fully laminated structure.

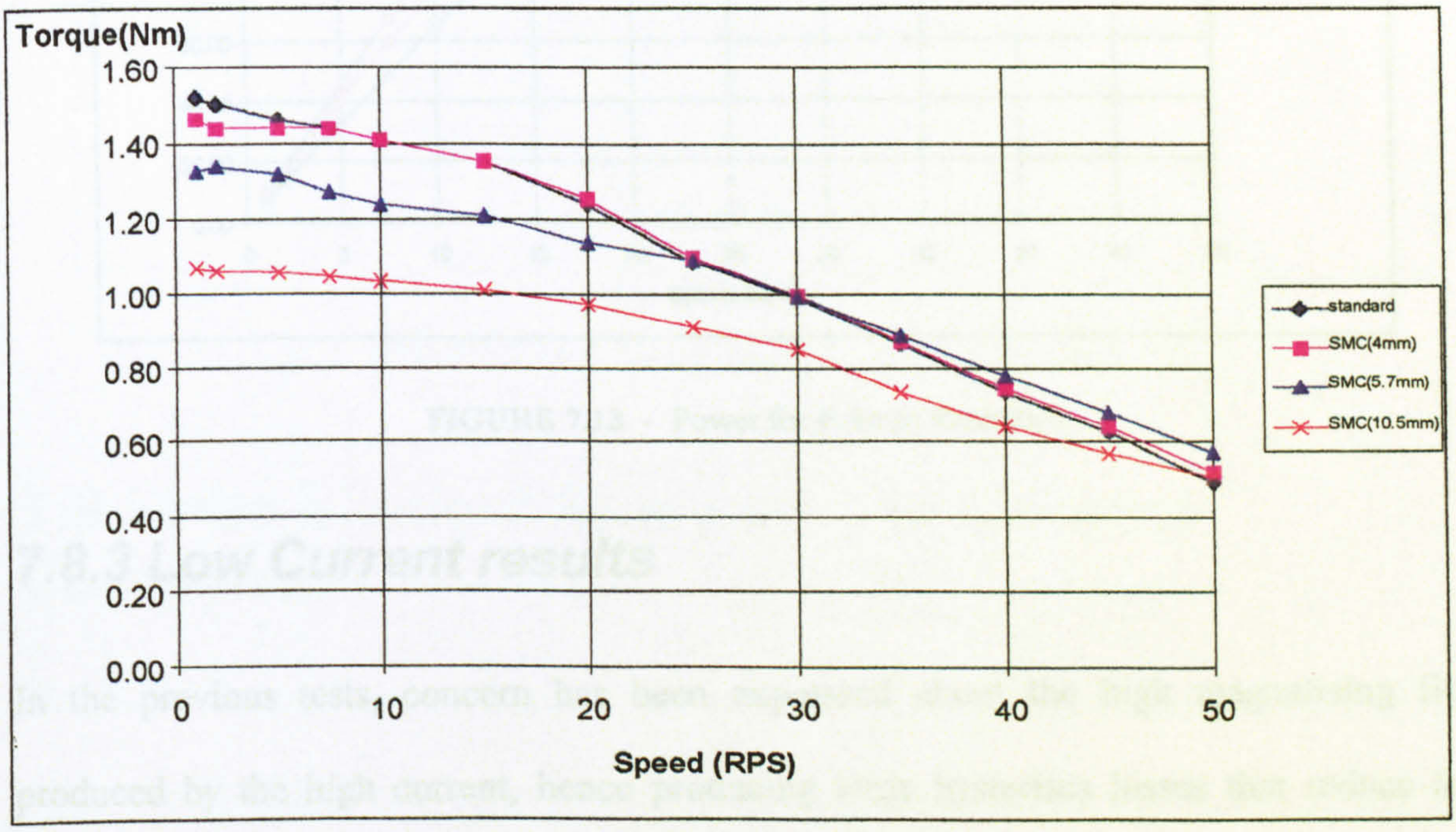


FIGURE 7.12 - Torque Speed for 4.9 Amps Excitation

The motors with the medium and longest lengths of SMC motors produce less torque at low to mid speeds. The motor with the longest length of SMC achieves equal torque at the maximum upper speed. The middle length of SMC material only gives equal torque to the standard design around mid speed. Referring to the shaft power graph of figure 7.13, the motor with the middle sized section gives an increased shaft power over the standard design. The largest section of SMC produced a flat power out curve, and at very high speeds may produce an enhanced output power.

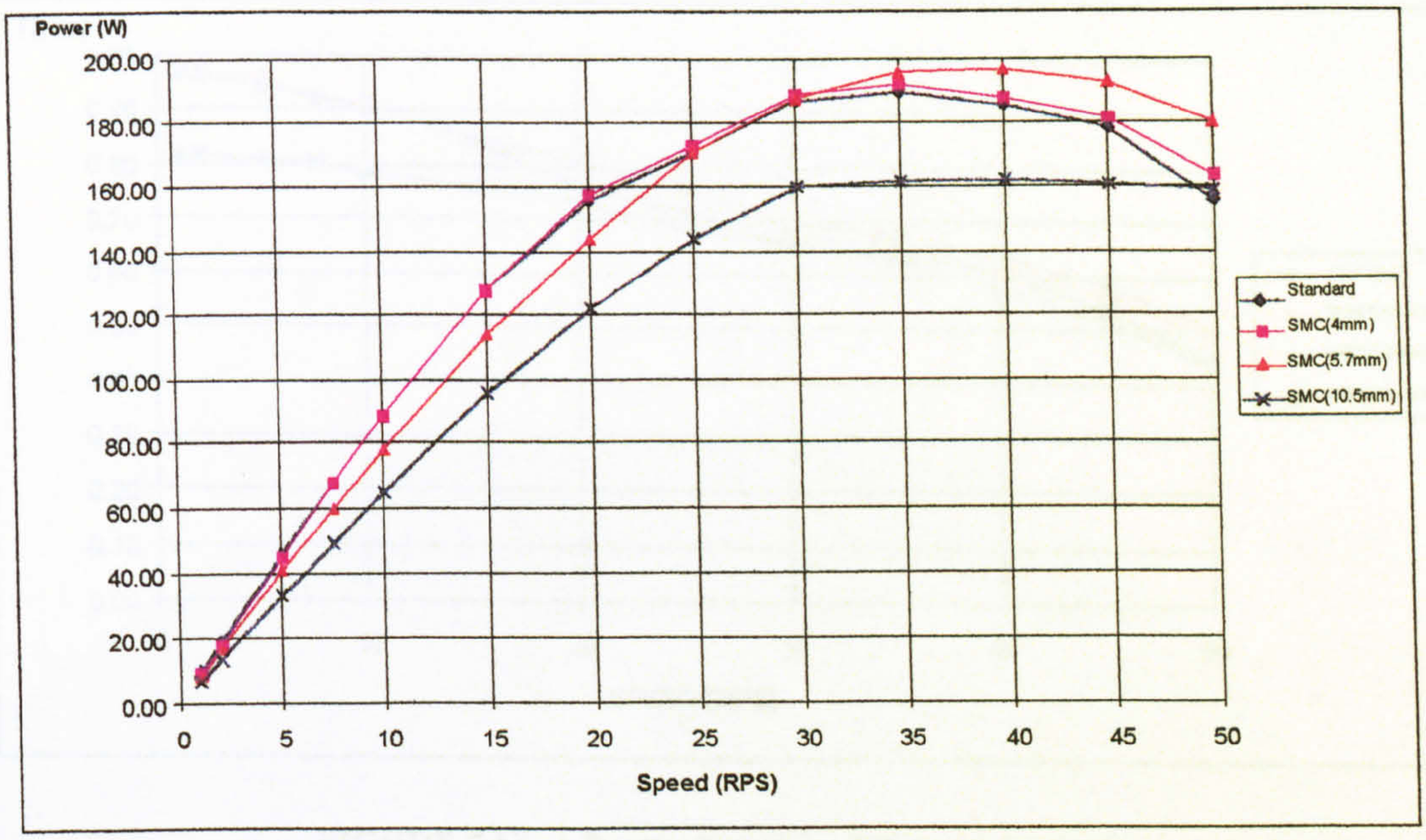


FIGURE 7.13 - Power for 6 Amps Excitation

7.8.3 Low Current results

In the previous tests, concern has been expressed about the high magnetising field produced by the high current, hence producing large hysterises losses that reduce low speed torque. The motors were now tested with a current of 3.8 Amps RMS. The low current experimental data (figure 7.14) is more favourable to the material for hysterises losses. The two shorter SMC component machines offer a similar low speed performance. The increased flux density of the smaller 34 frame motor is sufficient to offset the gain produced by the isotropic nature and the added saliency. They both produced a better mid speed torque output, which interprets as a greater shaft power output than the standard motor. The motor with the middle sized SMC section obtains a better high speed output due to the reduced eddy current losses. This has to be contrasted with the slightly lower holding torque output. The larger SMC sections offer a slightly lower performance, and do not appear to offer any advantage over the smaller sections. The resulting power out curves would be very similar.

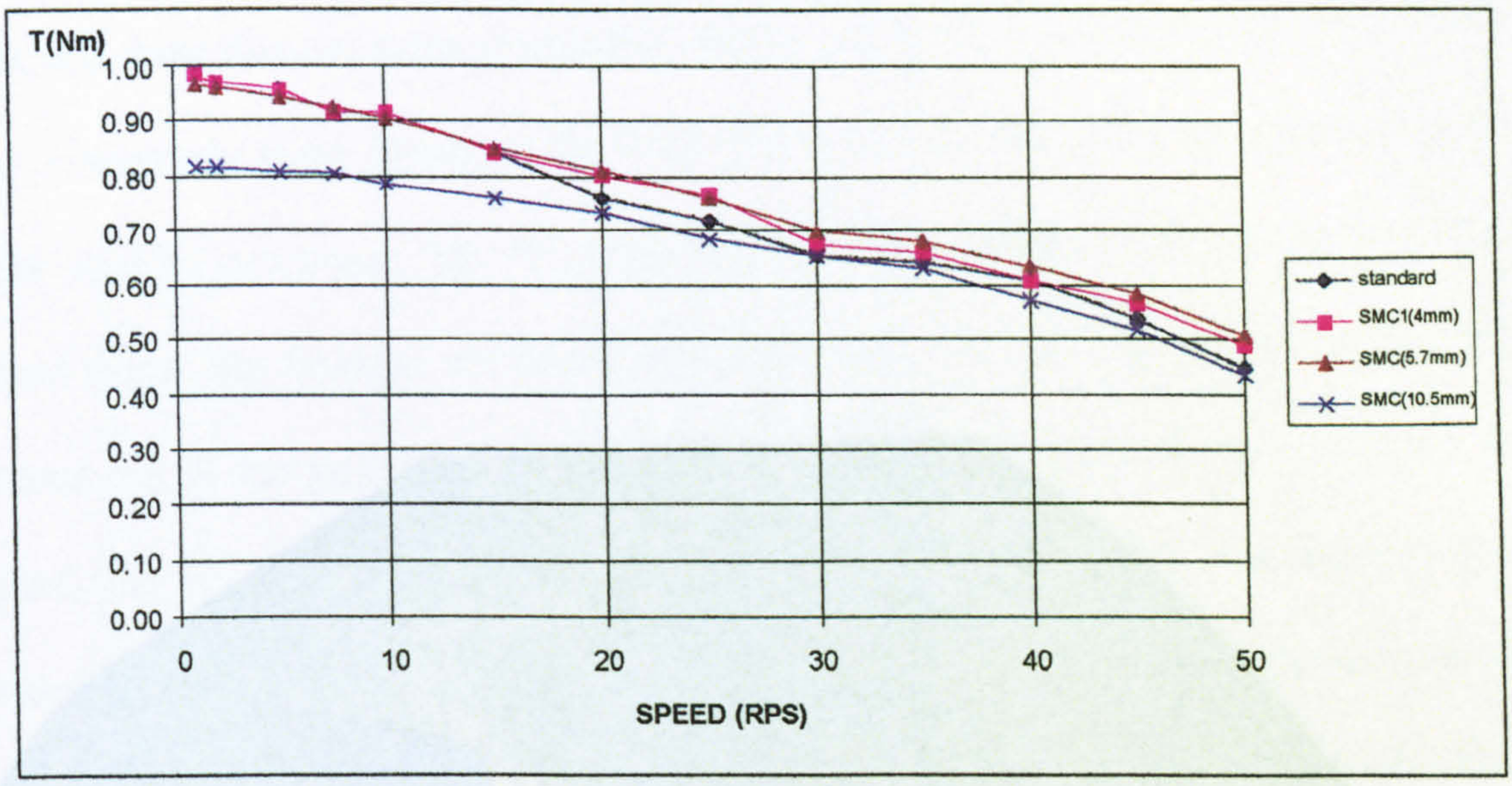


FIGURE 7.14 - Torque Speed for 3.8 Amps Excitation

7.9 Finite Element Study on 34-Frame for Comparison

The following section details observations made by a brief finite element study for the comparison of SMC component motors to the standard 34-frame type motor. Figure 7.15 shows a finite element representation of the 34-frame motor. The analysis used a SMC length of 5.7 mm, with the motor operating in two phase. The results are summarised in table 7.3.

	FE (Standard Model) Average Torque	FE (5.7 SMC Middle core) Average Torque	Experimental Standard Motor Maximum Low Speed Torque (Nm)	Experimental 5.7 mm Middle SMC Core Maximum Low Speed Torque.
Current (3.8 Amps RMS.)	1.00	1.13	0.99	0.97
Current (4.9 Amps RMS.)	1.29	1.31	1.35	1.16
Current (6.0 Amps RMS.)	1.56	1.44	1.52	1.32

TABLE 7.3 - FE to Experimental Data Results

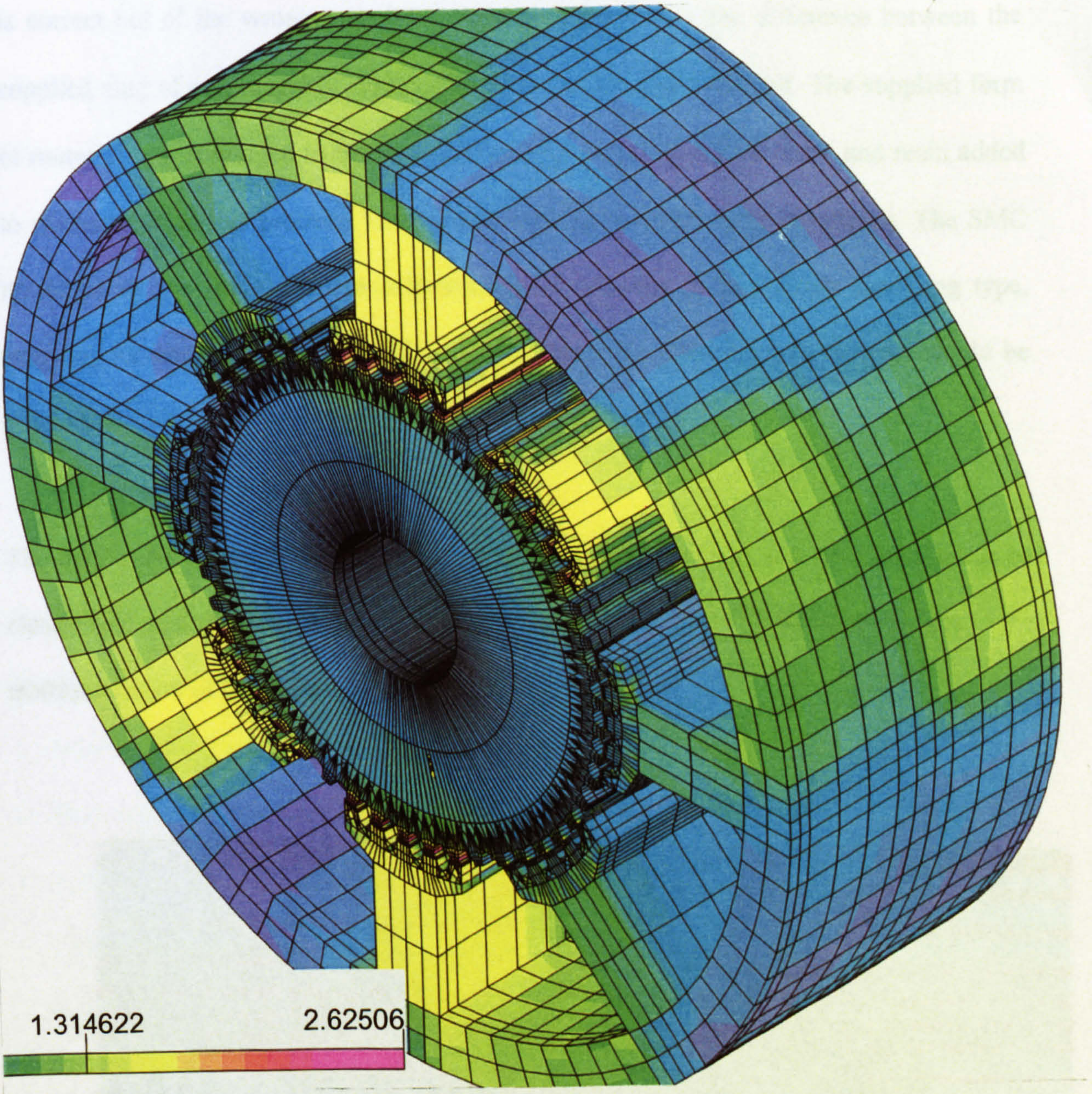


FIGURE 7.15 - FE Representation of the 34 - Frame Motor

The results show that the finite element prediction for the average torque of the standard motor is relatively good. However the FEA results for the SMC ring are overoptimistic. For the middle excitation, the 4.9 Amps FE predicts a higher performance than the standard motor, but this was not found in experimental data. At higher currents the trend is correct but of the wrong magnitude. One problem lies in the difference between the supplied slug of soft magnetic composite material and that modelled. The supplied form of material was envisaged to be machined and hence had some lubricant and resin added to protect it from this process. This would degrade its magnetic performance. The SMC modelled by the finite element software was considered to be a press moulding type, retaining its magnetic properties. If the motor was to be manufactured the part would be pressed.

The flow of flux through the back iron can be seen in figures 7.16 and 7.17. Here it can be clearly seen that the flux prefers to flow in the z-direction in the SMC machine, due to the isotropic nature of the material.

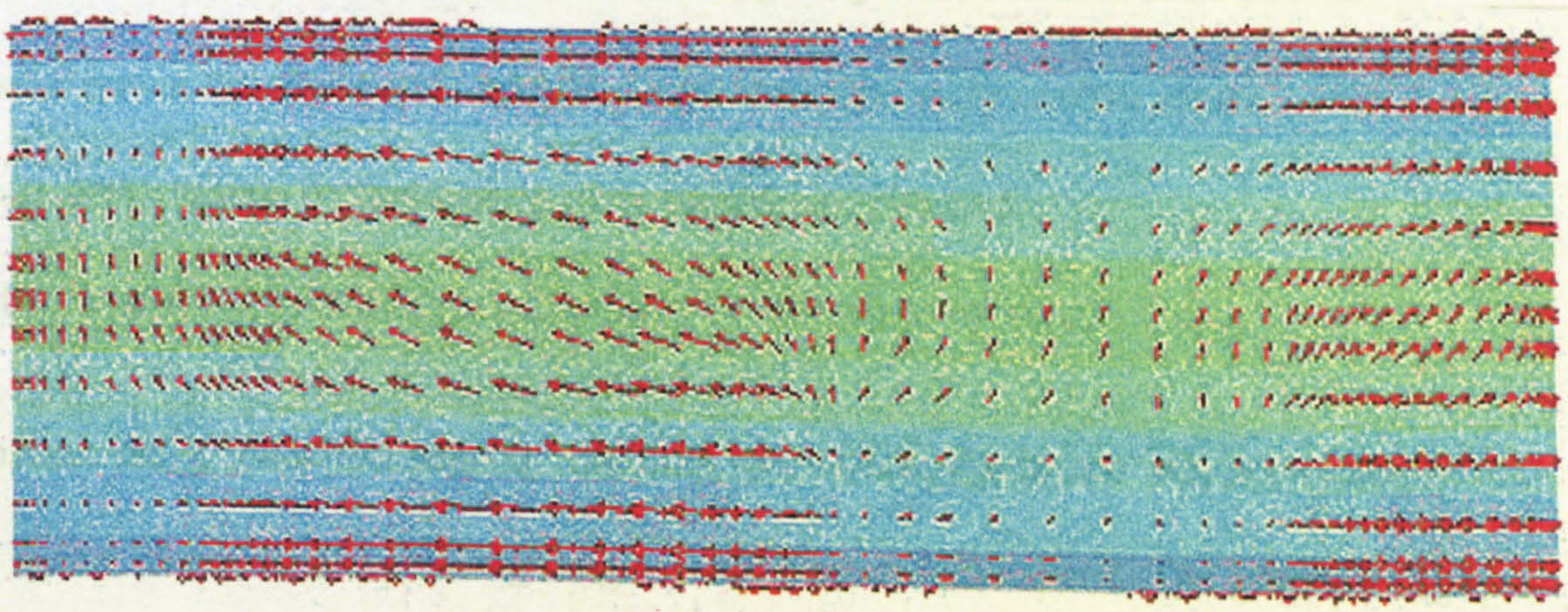


FIGURE 7.16 - Flux Flowing through Back iron of a Standard Motor

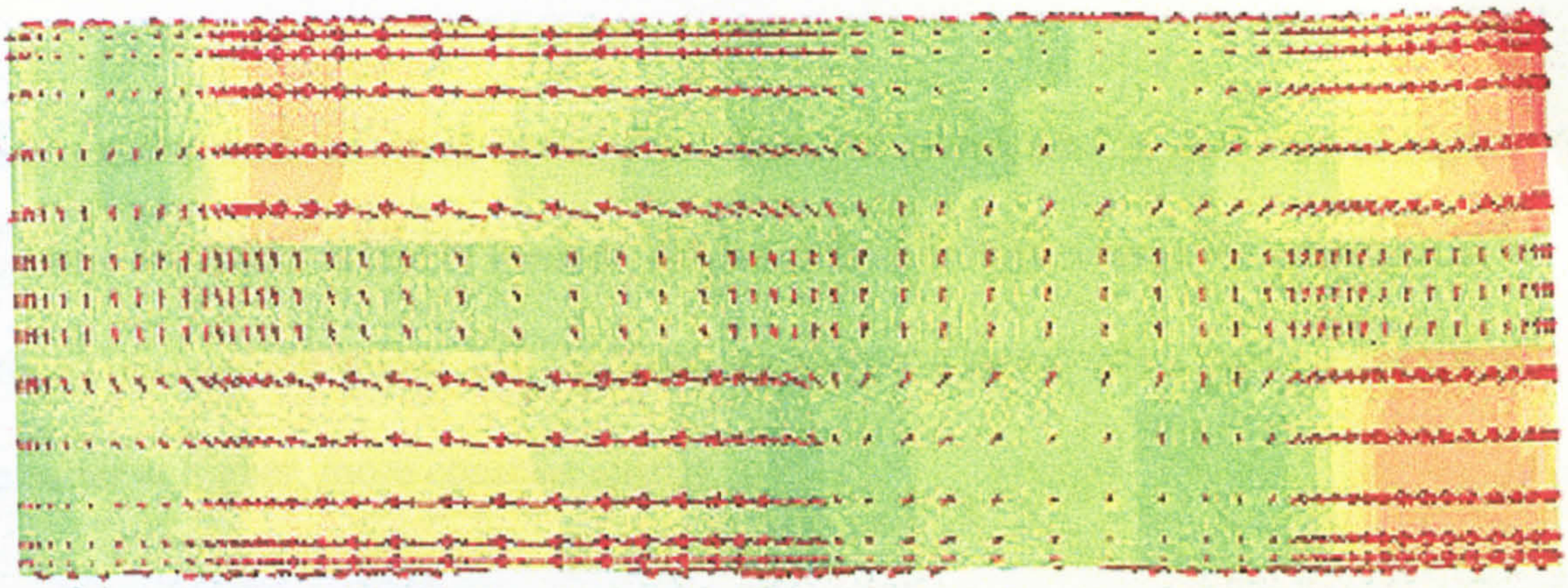


FIGURE 7.17 - Flux Flowing through Back iron of a SMC Composite Motor

7.10 Applications of Soft Magnetic Composite in Hybrid Stepping Motors

The key features of using these materials in the hybrid stepper motor which have been highlighted by this chapter are:

- Improved isotropic nature with press mouldings, giving the possibility of improvements in static and low speed torque.
- Smoother torque/ speed performance curves, giving more uniform shaft power.
- Greater power out at elevated speeds due to the lower eddy current losses.
- Lower performance by machining parts rather than pressing.
- Reduced performance at higher saturation levels due to hysteresis losses.

If the motor has a reduction gearing mounted to its shaft, the motor will not typically run at low speed. The normal operating range would be at middle speed. The graphs of torque

and power versus speed, indicate improvements in this particular region, making the motors with SMC suitable for adoption.

In pusher systems, such as wood sawing, the shaft power required is generally at higher speeds, where the motor is highly loaded. The lower speed range is used for accelerating a smaller load (such as the wood), and the motor requires the power when pushing the wood into the cutter at higher, more constant speeds.

The very flat (but much lower torque) torque/ speed response of the motor with the large section of SMC material could be utilised where a fast or simple control method of speed and acceleration was needed, such as 'on-the-fly' applications. If a packaging machine had different length products, the acceleration required to process the products would be different for each length. With a flat torque speed capability, the complete acceleration profile would only be set with one value that would match the linear power out curve of the motor. This would offer a simple control system and one that would not have to recalculate the acceleration for each product. The lower torque could possibly be used to protect the products from damage during a jam. The motor would stall before excessive force could be delivered.

A standard motor could have its performance rewound for a higher speed by reducing the number of phase turns and hence lowering the inductance. If the finite element predictions for a pressed section of soft magnetic material are accepted, then the SMC motor design would offer an improvement in performance over the complete torque range. However the gradient of shaft power to speed in the results shows a trend towards a flat power out

curve (figure 7.15). Once maximum power is reached, the torque falls off at a rate of $1/\omega$, whereas a standard rewound motor would have high speed torque fall off at a greater rate.

It can be concluded, therefore, that there are some distinct possibilities and advantages for the use of soft magnetic composite materials in hybrid stepping motors. The experimental results have shown that although, the designs have a lower low speed performance there is significant high speed improvement, due to reduced losses. The use of pressed parts would offer greater performance than machined parts.

CHAPTER 8

CONCLUSION

8.1 Conclusion

A stepping motor is a synchronous motor which is used in low cost positioning systems. Within their dynamic error limits, accurate path following and precise point to point control can be achieved. Their inherent holding torque and stiffness make them ideal for point to point positioning and for point to point motion applications. Stepping motors have been traditionally used in the office machinery market, and designs are still based upon types suitable for slow speeds, operating within in the stop start region.

The construction procedures of hybrid stepping motors can dictate the possible improvements of the motor due to machining processes and geometry tolerances such as air gap radius. This thesis includes a comprehensive study of the manufacturing process for these motors and the quality test procedures to ensure consistent motor performance. This study allowed the author to understand some of the 'real world' constraints on designing these motors. The verification of motor performance can be achieved by a series of tests that have been described that allow a designer to understand some of the fundamental performance elements in the motor.

Traditionally the hybrid stepping motor has been designed and analysed using a combination of experimental and empirical two dimensionalised studies. Two dimensional finite element software has been available for some time that allows a designer to study

the air gap tooth regions of the motor. However there has been little study into complete three dimensional modelling of the hybrid stepping motor, due to the complexity of the motor and the limited power of early 3-D FEA packages. The author has presented to his belief the first comprehensive study of the motor using three dimensional finite element analysis. This has allowed the author to describe key fundamental properties with a new insight.

From the study of the motor in 3-D, properties of the electromagnetic structure of the motor, in particular the back iron and teeth (which in the pasthave been widely disregarded) have been noted. These have been incorporated in to a new lumped element model that runs on a personal computer.

The mathematical models include derived equations for fitting the characteristics of highly permeable modern electromagnetic steels, and permanent magnets. These have been shown to perform better than the established Jiles-Atherton equations.

A key feature of the mathematical model is a technique of analytical solutions to the paths of flux within the motor using ellipses. The ellipse has been shown to be a more accurate representation of flux lines in certain parts of doubly salient machines, compared to circular paths assumed by previous authors.

Recent designs of stepping motors have looked to maximise low speed torque, without consideration of dynamic effects such as inductance and hence high speed performance has not been necessarily been improved. Industrial machinery demands faster throughput,

requiring increased torque at higher speeds, maximising shaft power. In a market sector coming increasingly under attack from brushless DC servo systems, stepping motor designs must retain their low cost, and improve their power output.

The software and modelling technique developed in this project has allowed development of two quite different conceptual designs of high shaft power motors. The first described in chapter 6 involved a redesign of the stator tooth profile to produce smoother operation, with low vibration and acoustic noise across a wide range of speeds. Whilst there was a small drop in holding torque there was a significant increase in high speed power out. When coupled with a microstepping drive a further improvement in performance has been demonstrated. Such designs should find acceptance in machines requiring increased throughput, quiet operation and low vibration.

The second new design described in chapter 7 used soft magnetic composite materials in place of some laminations in the stator. Two motors were analysed by finite element analysis (a size 42 and 34 frame size), and predicted that a pressed moulding of the new lamination would offer increased static torque, due to the composite isotropic effect in the motor back iron. Due to the material available in terms of diameter size the smaller 34 frame motor was built. This demonstrated lower eddy current losses and improved shaft power at medium to high speeds.

For high speed performance voltage is generally the most important factor. High speed performance can be attained by using low inductance windings, requiring high current for

low speed torque. The new designs should allow higher speeds with lower voltages, and lower currents to be utilised. These high speed improvements are not voltage related.

8.2 Author's Contribution to Knowledge

A large amount of work has been carried out by the author on the study of stepper motors using two dimensional and three dimensional finite element methods. This thesis offers the first comprehensive study of the hybrid stepping motor with three dimensional electromagnetic analysis. The effect of the laminations in parts of the motor has been demonstrated to be a hindrance in the torque production of the motor. The author has presented some possible ways of using soft magnetic composite materials to reduce the loss of energy due to eddy currents associated with axial flux paths. Design work on tooth profiling has been demonstrated that can produce greater shaft power and lower mid-speed resonance in the motor. Together these advantages will allow stepping motors to be manufactured for applications requiring higher speed positioning with low acoustic noise.

Completely new equations for modelling the new generation of highly permeable magnetic steels have been detailed. These new equations have also been used for modelling numerous characteristics of the motor. Elliptical path mapping of flux lines has been demonstrated to give a more accurate description of flux contours.

8.3 Future Work

It has been demonstrated in this project that two dimensional finite element methods are extremely useful for modelling the tooth / air gap region of the motor. With the recent development of two dimensional design environments which through parameterisation of

the geometry allow the rapid creation of FE meshes, it would be worthwhile to incorporate such a link into the HyStep program.

As the project neared its completion, soft magnetic materials of the size 42 frame diameter became available. It would be particularly worthwhile to manufacture a size 42 frame motor with a SMC ring, since the simulations were very encouraging and should produce a better improvement in performance than was achieved with size 34 frame motors. Design of a rotor incorporating powdered iron would also be attractive and should be investigated to find how SMC behave in physically rotating axial component.

This project has shown two very different ways in which stepping motors can be improved, each giving at least 10% more high speed power output. Further work should be done to determine what improvements can be gained by the design of a motor which incorporates both the revised tooth profile and appropriately placed powdered iron sections. The potential for a very significant increase in high speed power exists.

REFERENCES

1. W.R. Croon, K.F. Brodsky, and J.L. Lobsinger, 'Hybrid Versus Reluctance a Guide to Selection', Incremental Motion, Control, System and Device Conference 1986, Champaign, Illinois, Pg. 303-313.

2. A Michaelides, and C Pollock, 'A New Magnetic Flux Pattern to improve the Efficiency of the Switched Reluctance Machine', IEE Proc. Int. Conference on Electrical Machines, Sept. 1992, Vol. 2, Pg. 527-531.

3. Parker Compumotor Stepper Motor Application Guide 1995, Pg. 25-27.

4. 'Applications of the Closed Loop Motor', IEEE Transactions on Automatic Control, Vol. AC-13, No.5, October 1968, Pg.464-474.

5. S. Hunt, 'Motors take it a Step at a Time', Design Engineer Magazine, March 1991.

6. H.D. Chai, 'Permanence Based Step Motor Model Revisited', Incremental Motion, Control, System and Device Conference 1985, Champaign, Illinois, Pg. 399-410.

7. M. Jenkins, 'Design of Two Hybrid Motors', Report for Stebon Ltd. from Dept. of Engineering, University of Sheffield, Oct. 1989.

8. Berger Laher, '5-Phase Stepper Motors', Catalogue Publication 1991.

9. Positec/ Berger Lahr, '3-Phase Stepper Motors', Catalogue Publication 1994.

10. 'Three Phase Type Stepping Motor', United States Patent No. 5315192, May 24, 1994.

11. Sanyo, 'Step-Sync 5-Phase Stepper Motor Designs', Catalogue Publication 1996.

12. Pacific Scientific, 'PowerMax - Flux Focusing Hybrid Stepper Designs', Catalogue Publication 1995.

13. M. Juffer, 'Modelling and Analysis of High Performance Hybrid Stepper Motors', Incremental Motion, Control, System and Device Conference 1990, Pg. 220-225.

14. M. Kobori, and J.R. Hendershot. 'High Performance 2-Phase and 5-Phase Hybrid Stepper Motors', Incremental Motion, Control, System and Device Conference 1984, Champaign, Illinois, Pg. 241-252.

15. Vector Fields Ltd., 'The OPERA 3-D reference Manual', Publication 1994.

16. Vector Fields Ltd., 'The TOSCA reference Manual', Publication 1994.

17. D.P. Baker, 'Two Phase Hybrid Stepping Motors', IEE Colloquim on Stepping Motors and their Control, Digest 1994/017, January 1994, Pg. 2/1-2/3.

18. K.S. Kordik, 'Step Motor Inductance Measurement', Incremental Motion, Control, System and Device Conference 1975, Champaign, Illinois, Pg. C1.

19 M.R. Harris, A. Hughes, and P.J. Lawerenson, 'Static Torque Production in Saturated Doubly-Salient Machines', IEE Proceedings on Power, Vol. 122, No. 10, Pg. 1121-1125, Oct. 1975.

20. P.W. Lee, and C. Pollock, 'Flux Linkage Estimation in Electrical Machines', Conference Proceedings of 5th Interantional Conference on Electrical Machines, Paris, Vol.3, Pg. 463-467, Sept. 1994.

21. P.W. Lee, C. Jolliffe, and C. Pollock, 'Measurement of Flux Linkage in the Hybrid Stepper Motor', Conference Proceedings of 5th Interantional Conference on Power Electronics and Variable Speed Drives, London, Oct. 1994, Pg. 131-136.

22. F.T. Dewolf, 'Measurement of Inductances of DC Machines', IEEE Transactions on Power Apparatus and Systems. Vol. PAS-98, No.5, Sept./Oct. 1979, Pg. 1636-1644.

23. E.D. Ward and J.W. Reider, 'An Approach to Step Motor Understanding', Incremental Motion, Control, System and Device Conference 1986, Champaign, Illinois, Pg. 145-165.

24. Y.H. Wang, and J.R. Smith, 'The Lastest Study on CAD of Stepper Motors', International Conference on Electrical Machines 1992, Pg. 642-651.

25. U.Y. Huh, and B.C. Kuo, 'Hybrid Step Motor Simulation using Magnetic Circuit Model', Incremental Motion, Control, System and Device Conference 1986, Champaign, Illinois, Pg. 319-326.

26. K. Mizutani, S. Hayashi, and N. Matsui, 'Modelling and Control of Hybrid Stepping Motors', IAS IEEE 1993, Pg. 289-294.

27. B.C. Kuo, and D. Hunter, 'How to use the IBM PC for Step Motor Design', Incremental Motion, Control, System and Device Conference 1984, Champaign, Illinois, Pg. 187-213.

28. B.C. Kuo, P.J. West, W. Yeadon, S. Meyerson, F. Brodsky, 'Computer Aided Design of Step Motors', Incremental Motion, Control, System and Device Conference 1982, Champaign, Illinois, Pg. 241-254.

29. H.D. Chai, 'Permeance Between Toothed Structures', Incremental Motion, Control, System and Device Conference 1978, Champaign, Illinois, Pg. 45-54.

30. B.C. Kuo, and U.Y. Huh, 'Permeance Models and their Applications to Step Motor Design', Incremental Motion, Control, System and Device Conference 1986, Champaign, Illinois, Pg. 351-367.

31. P.A. Ward, and P.J. Lawerenson, 'Magnetic Permeance of Doubly-Salient Air-gaps', IEE Proceedings, Pt. B, Vol. 124, No.6, June 1977, Pg. 542-544.

32. H.D. Chai, 'Permeance Model and Reluctance. Force Between Structures', Chapter from 'Theory and Applications of Step Motors', West Publishing Company, 1974.

33. M. Jufer and G. Heine, 'Hybrid Stepper Motor Torque and Inductance Characteristics with Saturation Effects', Incremental Motion, Control, System, and Device Conference 1982, Champaign, Illinois, Pg. 207-214.

34. M.P. Materu and R. Krishan, 'Analytical Prediction of SRM Inductance Profile and Steady State Average Torque', Conference Proceedings IEEE IAS Annual Meeting 1990, Pg. 214-223.

35. J. Sabonnadiere, 'Computing EM Fields', IEEE Spectrum Magazine, November 1992, Pg. 52-56.

36. W. Cole, and H.D. Chai, 'An Assessment of Various Magnetic Simulation Tools', Incremental Motion, Control, System, and Device Conference 1989, Champaign, Illinois, Pg. 1-5.

37. B. Forghani, and S. Rao, 'PC FEM Software', Incremental Motion, Control, System, and Device Conference 1990, Champaign, Illinois, Pg. 8-12.

38. B. Forghani, '3-D Finite Element Solution of a Hybrid Stepper Motor', Power Converters and Intelligent Motion, Vol. 16, Pg. 19-22.

39. E.C.T. So, and S.J. Yang, 'Calculation of Stator Radial Vibration for a Hybrid Stepper Motor using 30D Finite Element Method', Journal of Applied Physics, Vol. 73 Pt. B, Iss. 10, May 1993, Pg. 6799-6801.

40. C.M. Jolliffe, A.M. Michaelides, and C. Pollock, 'Three Dimensional Finite Element Analysis for Hybrid Stepping Motors', Proceedings of IEEE Conference on Magnetics, Okyama, Japan. 1996.

41. K. Richardson, and C.M. Jolliffe, 'Design of Hybrid Stepper Motors and Switched Reluctance Machines', Proceedings of Vector Fields User Meeting, Blenheim Palace, Oxford, September 1994.

42. K. Richardson, and C.M. Jolliffe, 'The use of Finite Element Analysis in Machine Design at the University of Warwick', Vector Fields Electromagnetics Journal, Vol. 11, No. 1, 1995, Pg. 3-4.

43. J.C. Maxwell, 'Treatise in Electricity and Magnetism', 1881.

44. Vector Fields Ltd., ' The OPERA 2-D Reference Manual', Publication 1995.

45. S.R. Huard, ' A Finite Element Based Lumped Parameter Model for 1.8° Stepping Motors', Incremental Motion, Control, System, and Device Conference 1990, Champaign, Illinois, Pg. 62-74.

46. S.R. Huard, ' A Finite Element Based Lumped Parameter Model for a Hybrid Stepping Motors', Incremental Motion, Control, System, and Device Conference 1992, Champaign, Illinois, Pg. 220-233.

47. J. Simkin, and C.W. Trowbridge, 'Three-dimensional Non-linear Electromagnetic Computations using Scalar Potentials', IEE Proc. Pt. B, Vol. 127, No. 6, Nov. 1980, Pg. 368-374.

48. Vector Fields Ltd., ' The OPERA 3-D Training Manual', Publication 1993.

49. A. Michaelides and C. Pollock, 'The Effect of End-Core Flux on the Performance of the Switched Reluctance Motor', IEE Proc on Electric Power Applications, Vol. 141, No. 6, Nov. 1994, Pg. 308-316.

50. Borland INC. , 'C++ Development Language V4.5 User Manual', Publication 1995.

51. D.C. Jiles and D.L. Atherton, 'Theory of Ferromagnetic Hysteresis', Journal of Magnetism and Magnetic Materials, Vol. 61, 1986, Pg. 48-60.

52. E.G. Cullwick, 'The Fundamentals of Electro-Magnetism', Cambridge University Press, 3rd Edition, 1996, Pg. 199-201.

53. A. Cornish-Bowden, and D.E. Koshland, 'Diagnostic uses of the Hill Plot', Journal of Molecular Biology, Vol. 95, 1975, Pg. 201-21.

54. Saber Optional Template Library Manual, Ch. 2, 1993, Pg. 1-30.

55. C.M. Jolliffe, C. Pollock, and M.J. Chappell, 'A Novel Algebraic Expression for Modelling Non-linear Magnetic Steels with Direct uses for Electric Motor Design Software Packages', Proceedings of IEEE Conference on Magnetics, Okyama, Japan. 1996.

56. Journal of Facsimile Applications, AEA Technology, Harwell, Didcot, Oxfordshire, UK.

57. H.D. Chai, and K. Konechy, 'Effects of Saturation on Step Motor Permeance and Force', Incremental Motion, Control, System, and Device Conference 1982, Champaign, Illinois, Pg. 232-240.

58. A. Michaelides, C.M. Jolliffe, and C. Pollock, 'Analytical Computation of Minimum and Maximum Inductances in Single and Two Phase Switched Reluctance Machines', Proceedings of IEEE Conference on Magnetics, Okyama, Japan, 1996.

59. A. Michaelides, C.M. Jolliffe, and C. Pollock, 'Analytical Computation of Minimum and Maximum Inductances in Single and Two Phase Switched Reluctance Machines', IEEE Transactions on Magnetics, 1997.

60. J Corda, and J M Stephenson, 'Analytical Estimation of the Minimum and Maximum Inductances of a Double-Salient Motor', International Conference on Stepping Motors and Systems, Leeds, Sept. 1979. Pg. 50-59.

61. M.K. Jenkins, T.S. Birch, and D. Howe, 'Static Torque Production in Hybrid Stepper Motors: The Influence of Magnet MMF', Proceedings of Third Int. Conference on Electrical Machines 1987, Vol. 282, Ch. 76, Pg. 270-274.

62. M.K. Jenkins, D. Howe, and T.S. Birch, 'An Improved Design Procedure for Hybrid Stepper Motors', IEEE Transactions on Magnetics, Vol. 26, No. 5, Sept. 1987, Pg. 2535-2537.

63. T. Kenjo, 'Stepping Motors and their Microprocessor Controls',, Oxford Science Publications, Claredon Press, Oxford, 1990.

64. P.P. Arcarnly, 'Stepping Motors: A Guide to Modern Theory and Practice', IEE Control Engineering, Series 19, Revised 2nd Edition, 1984.

65. J.N. Chiasson and R.T. Novotnak, 'Non-linear Speed Observer for the PM motor', IEEE Transactions on Automatic Control, Vol. 38, No. 10. October 1993, Pg. 1584-1588.

66. S. C. Chapra and R P. Canale, 'Numerical Methods for Engineers with Personal Computer Applications', McGraw-Hill, 1985, Pg. 594-617.

67. G. Singh, 'Mathematical Model of a Stepper Motor', Chapter from 'Theory and Application of Step Motors',, West Publishing Company, 1974.

68. M.R. Harris and J.W. Finch, 'Estimation of Static Characteristics in the Hybrid Stepping Motor', Incremental Motion, Control, System, and Device Conference 1979, Champaign, Illinois, Pg. 293-306.

69. J. Wagner, 'Determination of Step Motor Parameters with Magnetic Field Solution', COMPEL Journal, Vol. 13, No. 1, 1994, Pg. 137-139.

70. J. Wagner, 'Analysis of the Magnetic Field of a DC Machine', COMPEL Journal, Vol. 13, No. 1, 1994, Pg. 145-148.

71. D. Tormey, D. Torrey, and P. Levin, 'Minimum Air-gap Permeances Data for Doubly Slotted Pole Structures', Conference Proceeding for IEEE IAS, Seattle, Oct. 1990, Pg. 196-200.

72. T. Akiyam and T. Sueki, 'Torque Generation Mechanism of Stepping Motors', International Conference of Stepping Motors, Leeds, 1979.

73. K. Fukui, I. Wantanabe, and M. Morita, 'Compressed Iron Powder Core for Electric Motors', IEEE Transactions on Magnetics, Sept. 1972, Pg. 682-684.

74. M. Alakula, T. Cedell, M. Persson, and Lars Sjoberg, 'An Iron Composite Based Switched Reluctance Machine', IEEE Power Technical Conference, Sweden, June 1995, Pg. 251-255

75. P. Jansson, M. Persson, A.G. Jack, and B.C. Mecrow, 'Composites Pave the Way to the Electrical Machine Designs of the Future',, Euro PM '95, Birmingham, Oct. 1995, Pg. 1-8.

76. M. Persson and P. Jansson, 'Advances in Powder Metallurgy Soft Magnetic Composite Materials for Electrical Machines', IEE Colloquium on 'The Impact of New Materials on Designs', Dec. 1995, Pg. 4/1-4/6.

77. P. Jansson, M. Persson, A.G. Jack, and B.C. Mecrow, 'Powdered Soft Magnetic Material for Medium Frequency Applications. Soft Magnetic Materials '96, Gorham/ Intertech Conference, San Fransisco, Feb. 1996.

APPENDIX A

Motor Specifications

Details of Stebon SDT-1101-250-70 Motor Referred to in this thesis.

-Hybrid Design

-1.8° Step Angle (200 Steps for Revolution)

-Two phase Bipolar Winding

-Class F insulation to IEC 85 and BS2757

-Nema 42 Frame and Flange Size

-Bearing Loading, axial 180N maximum

-Bearing Loading, radial 330N maximum

-Maximum Detent Torque 0.12Nm

-Parallel Resitance per Phase 0.26 ohms

-Parallel Inductance per Phase 2.8 mH.

-Bi-polar Current per Phase 7 A

- Rotor Inertia 3.65 Kgm^2

-Motor Weight 4.8 Kg

-IP44 Rating

Details of Stebon SDL-851-250-45 Motor Referred to in this thesis.

-Hybrid Design

-1.8° Step Angle (200 Steps for Revolution)

-Two phase Bipolar Winding

-Class F insulation to IEC 85 and BS2757

-Nema 34 Frame and Flange Size

-Bearing Loading, axial 160N maximum

-Bearing Loading, radial 250N maximum

-Maximum Detent Torque 0.05 Nm

-Parallel Resitance per Phase 0.24 ohms

-Parallel Inductance per Phase 1.5 mH.

-Bi-polar Current per Phase 4.5 A

-Rotor Inertia 0.6 Kgm^2

-Motor Weight 2.2 Kg

-IP22 Rating.

APPENDIX B

Three Dimensional Algorithm (Opera-3-D / Tosca) [15,16]

Stationary magnetic fields consist of both solenoidal and rotational components. The field produced by electric currents has a rotational component inside the volumes where current flows. In the exterior space the field is solenoidal but the scalar potential is multi-valued. The field produced by magnetised volumes is solenoidal. It is convenient to separate the total field into two parts in order to obtain a description of the field in terms of the scalar potential. The total field intensity H may be expressed as the sum of the source field intensity H_s and the reduced field intensity H_m.

$$H = H_s + H_m \qquad (A.1)$$

The source field can be obtained directly from Biot-Savart law by integration over the region Ω_j containing the current

$$H_s = \int_{\Omega_s} \frac{J \times R}{|R|^3} \delta\Omega_s . \qquad (A.2)$$

The field satisfies

$$\nabla \times H_s = J , \qquad (A.3)$$

so that

$$\nabla \times \mathrm{H}_{\mathrm{M}} = 0. \qquad \text{(A.3)}$$

The reduced field intensity can now be represented using the reduced scalar potential, ϕ,

$$\mathrm{H}_{\mathrm{M}} = -\nabla\phi. \qquad \text{(A.4)}$$

The divergence of the flux density B is always zero. Introducing the permeability tensor, μ, and combining the expressions for the source and reduced field intensity, gives the partial differential equation for the reduced scalar potential.

$$\nabla \cdot \mu\nabla\phi - \nabla \cdot \mu\left(\int_{\Omega_s} \frac{\mathrm{J} \times \mathrm{R}}{|\mathrm{R}|^3} d\Omega_s\right) = 0 \quad \text{(A.5)}.$$

This equation can be solved using the finite element method. However, in magnetic materials the two parts of the field $\mathrm{H_M}$ and $\mathrm{H_s}$ tend to be of similar magnitude but opposite direction. Therefore, cancellation occurs in computing the field intensity H, that results in a loss of accuracy. The errors can be completely avoided by combining the total and reduced scalar potential representations. Hence, exterior to the volumes where currents flow the total field can be represented using the total scalar potential Ψ.

$$\mathrm{H} = -\nabla\Psi, \qquad \text{(A.6)}$$

where the total scalar potential satisfies

$$\nabla \cdot \mu \nabla \Psi = 0. \qquad \text{(A.7)}$$

The minimal combination consists of using the reduced scalar potential only inside volumes where current flows and the total potential everywhere else.

APPENDIX C

The following pages are the .CPP files from the HyStep modelling program. The .H header files are not included due to space restrictions.

```
//////////////////////////////////////////////
//**BH dialog window function ***//

DEFINE_RESPONSE_TABLE1(TBhDialog, TDialog)
                EV_COMMAND(104, EnterButton),
                EV_COMMAND(107, DefaultButton),
                EV_COMMAND(108, ClearButton),
END_RESPONSE_TABLE;

//** dialog defaults ***
TBhDialog::TBhDialog(TWindow* parent, const char* name)
:TDialog(parent, name),
                TWindow(parent)
{

//steel BH data default curve
DEFpoint=27;

DEF[0].b=0;
DEF[1] b=.64;
DEF[2].b=.92;
DEF[3].b=1.01;
DEF[4] b=1.1;
DEF[5] b=1.2;
DEF[6].b=1.3;
DEF[7] b=1.4,
DEF[8] b=1.45;
DEF[9] b=1.5;
DEF[10].b=1.55;
DEF[11].b=1.575;
DEF[12].b=1 6;
for(int bh=13, bh<27, bh++)
DEF[bh].b=DEF[bh-1].b+0.05;


DEF[0].h=0;
DEF[1] h=OSAm;
DEF[2].h=OSAm*1.7;
DEF[3].h=OSAm*2;
DEF[4].h=OSAm*2 4;
DEF[5].h=OSAm*3;
DEF[6] h=OSAm*4;
DEF[7].h=OSAm*6.2;
DEF[8].h=OSAm*8.1;
DEF[9] h=OSAm*11;
DEF[10].h=OSAm*16;
DEF[11].h=OSAm*20,
DEF[12].h=OSAm*27;
DEF[13] h=OSAm*42;
DEF[14].h=OSAm*60;
DEF[15].h=OSAm*82;
DEF[16].h=OSAm*115;
DEF[17].h=OSAm*150;
DEF[18].h=OSAm*190;
DEF[19].h=OSAm*233;
DEF[20].h=OSAm*280;
DEF[21] h=OSAm*345;
DEF[22].h=OSAm*450,
DEF[23].h=OSAm*600;
DEF[24].h=OSAm*800;
DEF[25].h=OSAm*1180;
DEF[26] h=OSAm*1600;

/////////** magnet bh data default **
DEFmagpoint=17;

DEFmag[0].b=0;
DEFmag[1].b=.155;
DEFmag[2].b=.27;
DEFmag[3] b=.35;
DEFmag[4] b=.43;
DEFmag[5].b=.5;
DEFmag[6].b=.57;
DEFmag[7].b=.62,
DEFmag[8].b=.66,
DEFmag[9].b=.7;
DEFmag[10].b=.74;
DEFmag[11].b=.775;
DEFmag[12] b= 8,
DEFmag[13].b=.84;
DEFmag[14].b=.87;
DEFmag[15] b= 88;
DEFmag[16].b=.9;


DEFmag[0].h=-1600*OSAm;
for(bh=1;bh<17; bh++) DEFmag[bh].h=DEFmag[bh-1].h+(100*OSAm);

}
//*** end of default data ***

//**FUNCTIONS CONTAINED WITHIN DIALOG WINDOWS***///
// DEFAULT BUTTON//
void
TBhDialog::DefaultButton() //LOADS DEFAULT CURVE INTO CURRENT CURVE SPACE
{
```

```
// "" HEADERS ""
#include <osl\geometry.h>
#include <owl\gdiobjec.h>
#include "com.h"
#include "bh.h"
#include "classla h"
#include "mast h"
#include "wchild.h"
#include "drawgeo.h"
#include "windmain.h"


//////////////////////
double BImag(NOST [5],NOST [5], FluxMmf [DIS],GEO,classBH);

FluxLink get(Curve,Curve,Curve, Curve ,double,Curve,double,double,double,
float,int,int );

FluxLink getDENT(Curve ,Curve ,Curve , Curve ,double);

double coENG(float,int,double,double,double,double,int,double);

//"" MENU COMMANDS LINKED to Functions""
DEFINE_RESPONSE_TABLE1(TDbWindow, TFrameWindow)
        EV_WM_PAINT,
        EV_COMMAND(201, CmBH),                          //call BH steel points input
        EV_COMMAND(601, CmBHcurve),     //call BH steel equation input
        EV_COMMAND(701, CmMAG),                         //call BH magnet points input
        EV_COMMAND(801, CmMAGcurve),    //call BH magnet points input
        EV_COMMAND(52, CmBHsave),       //save steel bh data
        EV_COMMAND(51, CmBHload),       //load steel bh data
        EV_COMMAND(301, CmDim),  //calls dimensions input window
        EV_COMMAND(401, CmDen),  //call flux density window
        EV_COMMAND(402, CmPf),   //call packing factor window
        EV_COMMAND(501, CmPath),  //starts paths calculation
        EV_COMMAND(901, CmGraph),
        EV_COMMAND(959, CmStat),
        EV_COMMAND(350, CmDrawGeo),

        END_RESPONSE_TABLE;


//***** initialisation of main window *****//
TDbWindow::TDbWindow(TWindow* parent, const char far* title)
:TFrameWindow(parent, title),
        TWindow(parent, title)
{
Attr.X=0;
Attr.Y=0,

//timer
CursorCounter = 0;
SystemCursors[0] = IDC_WAIT,
//

        AssignMenu(1);          //assign the menu to the screen
        MclassGEO.stringblank(); //clears the strings in class GEO
        BhCHK=0; //bh data is not entered
        DimCHK=1; //sets dimension check to 'dimensions not entered'
        FdCHK=0;  // flux density limit is entered
        PfCHK=0;  //pf set
        MclassGEO.PF.n=0.98; //packfactor default

        graphyes=1;// set flag so no flux/mmf graph is displayed until results

        MclassGEO.maxFD.n=2.1;
        gcvt(MclassGEO.maxFD.n,7,MclassGEO.maxFD.s);

        MalFM.clearhead();
        MmdFM.clearhead();
        MmFM.clearhead();
        Mclassbh.clearhead();
        Mclassmag clearhead();
        }

        /////
        //declare child window for graph
        DEFINE_RESPONSE_TABLE1(TGraphFrame, TFrameWindow)
        END_RESPONSE_TABLE;

        TGraphFrame::TGraphFrame(TWindow* parent, const char far * title)
        :TFrameWindow(parent, title)
        {
        Attr.X=100;
        Attr.Y=100;
        Attr.W=350;
        Attr.H=350;

        Attr.Style=WS_VISIBLE | WS_CAPTION | WS_BORDER | WS_SYSMENU | WS_MINIMIZEBOX
                                                        |WS_CLIPSIBLINGS | WS_THICKFRAME;
        }
        ////////////////////
```

```
/////
//declare child window for graph
DEFINE_RESPONSE_TABLE1(TGeoFrame, TFrameWindow)
END_RESPONSE_TABLE,

TGeoFrame::TGeoFrame(TWindow* parent, const char far * title)
:TFrameWindow(parent, title)
{
Attr.X=25;
Attr Y=25,
Attr W=450,
Attr H=450,
Attr Style=WS_VISIBLE | WS_CAPTION | WS_BORDER | WS_SYSMENU | WS_MINIMIZEBOX
                                                    |WS_CLIPSIBLINGS | WS_THICKFRAME;
}
//////////////////

//** DECLARE FUNCTIONS FOR TDbWindow (main window) class ****//
void
TDbWindow::EvPaint()
{

TPaintDC paintDC(HWindow); //paint main window
//**
if (Mclassbh.MbhFlag==1 && Mclassmag.MbhFlag==1) {BhCHK=1;}
else if (Mclassbh MbhFlag==2 && Mclassmag MbhFlag==2) {BhCHK=2,}
else {BhCHK=0,}
//***
char blank[]=" ";                  //** charcters to tell user what curves have been selected**//
char cur[]=" (CURVE)";                        // with what method
char poi[]=" (POINTS)";
char methodbh[10];
char methodmag[10];
//**              steel BH curve**
if (Mclassbh.MbhFlag==0) {strcpy(methodbh,blank);}       //no points entered no method
else if (Mclassbh.MbhFlag==1) {strcpy(methodbh,poi),}    //points method
else {strcpy(methodbh,cur);}                                        //equation method
//**              magnet BH curve
if (Mclassmag.MbhFlag==0) {strcpy(methodmag,blank);}   //no points entered no method
else if (Mclassmag.MbhFlag==1) {strcpy(methodmag,poi);} //points method
else {strcpy(methodmag,cur);}                                                  //equation method
//********************

//** display names of steel and magnet
char title[]="BH curve in current use :";
paintDC TextOut(0,10, title, strlen(title));
char holdbh[25];
strcpy(holdbh,Mclassbh.MbhName);
strcat(holdbh,methodbh);
paintDC.TextOut(175,10, holdbh, strlen(holdbh));

char mtitle[]="Magnet in current use  :";
paintDC.TextOut(0,25, mtitle, strlen(title));
char holdmag[25];
strcpy(holdmag,Mclassmag MbhName);
strcat(holdmag,methodmag);
paintDC.TextOut(175,25, holdmag, strlen(holdmag));
//*** - check to see if bh have been entered **
if (BhCHK==0){
char chkTMP[]="BH data missing, or of differing techniques!";
paintDC.TextOut(1,46, chkTMP, strlen(chkTMP));}
else{char chkTMP[]="BH data Entered";
paintDC.TextOut(1,46, chkTMP, strlen(chkTMP)),}

//*** - check to see if dimensions have been entered **
if (DimCHK!=0){
char chkTMP[]="Dimensions not completed !";
paintDC.TextOut(1,70, chkTMP, strlen(chkTMP));}
else{char chkTMP[]="Dimensions Entered";
paintDC.TextOut(1,70, chkTMP, strlen(chkTMP));}

// *** maximum flux density ***
char MXfd[]="Maximum Flux Density is set at : ";
strcat(MXfd,MclassGEO.maxFD.s);
paintDC.TextOut(1,85, MXfd, strlen(MXfd));

}
//////////////////////////////////

//**** main window function *****//

// ** SAVE BH **
void
TDbWindow::CmBHsave()
{
//** SAVE DIALOG ** looks for .bh files
TOpenSaveDialog::TData data(
OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,
"BH Files (*.BH)|*.BH|",
0,
"",
"BH"
);

//** if file can be saved holds filename in data.filename
if ((new TFileSaveDialog(this, data))->Execute()==IDOK){
```

```
                ofpstream out(data FileName);
                 if(!out) MessageBox("Cannot Open file", "Error", MB_OK);
                                        else{
                                                        out<<Mclassbh, //because class is streamed
                                        }

                }
}

//** load BH **
void
TDbWindow:.CmBHload()
{
static TOpenSaveDialog:.TData data ( //displays only .BH files
OFN_HIDEREADONLY,
"BH Files (*.BH)|*.BH|",
0,
"",
"BH"
);

if ((new TFileOpenDialog(this, data))->Execute()==IDOK){
                ifpstream in(data.FileName);
                 if(!in) MessageBox("Cannot Open file", "Error", MB_OK);
                                else{
                                                in>>Mclassbh, //load data back into class
                                 }
                }
}
////// FUNCTIONS FOR BH FILES ENDED //////

//// BH STEEL MANUPLATION ///
void
TDbWindow::CmBH()
{
 TBhDialog* the_pad=new TBhDialog(this, TResId(1)); //calls the Dialog window (ID 1)

//*** loads data from main window ***///

                the_pad->steel_mag=0; // informs dialog window we are dealing with steel data

                for (int loop=0; loop<100; loop++) { //loads bh data to dialog window
 the_pad->bh[loop] b=Mclassbh.Mbh[loop].b;the_pad->bh[loop] h=Mclassbh.Mbh[loop].h;}

                strcpy(the_pad->name,Mclassbh.MbhName);//copies the bh name to Dialog
                the_pad->point=Mclassbh.MbhPoint; //copies the number of points

//** when the user presses enter** - return the data back to the main window
if (the_pad->Execute()==IDOK){

                for(int j=0,j<100,j++){ Mclassbh.Mbh[j].b=the_pad->bh[j].b,
                                                        Mclassbh.Mbh[j].h=the_pad->bh[j].h;}
                                                        strcpy(Mclassbh.MbhName,the_pad->name);
                                                        Mclassbh.MbhPoint=the_pad->point;
                                                        Mclassbh.MbhFlag=1;
                                }
}

//////////////////////////////////////////////
//** function to deal with steel equation data **
void
TDbWindow::CmBHcurve()
{

// creates a transfer buffer to put data directly into windows //
                struct TransferBuffer{
                char a[15],b[15],n[15],e[15],k[15],name[20];}tb;

                strcpy(tb.a,Mclassbh.MbhCurve[0].s);
                strcpy(tb.b,Mclassbh.MbhCurve[1].s);
                strcpy(tb.n,Mclassbh.MbhCurve[2].s);
                strcpy(tb.e,Mclassbh.MbhCurve[3].s);
                strcpy(tb.k,Mclassbh.MbhCurve[4].s);
                strcpy(tb.name,Mclassbh.MbhName);

TBhCurDialog* the_pad=new TBhCurDialog(this, TResId(4)); //creates new dialog window
the_pad->steel_mag=0; //steel data is being dealt with
the_pad->SetTransferBuffer(&tb);
if (the_pad->Execute()==IDOK){
                                                        //return values from dialog to main window data
strcpy(Mclassbh.MbhCurve[0] s,tb.a);
strcpy(Mclassbh.MbhCurve[1] s,tb.b);
strcpy(Mclassbh.MbhCurve[2].s,tb.n);
strcpy(Mclassbh.MbhCurve[3].s,tb.e);
strcpy(Mclassbh.MbhCurve[4] s,tb.k);
strcpy(Mclassbh.MbhName,tb.name);

Mclassbh.MbhCurve[0].n=atof(Mclassbh.MbhCurve[0] s); //atof function converts string data to
Mclassbh.MbhCurve[1].n=atof(Mclassbh.MbhCurve[1].s), //numerical
Mclassbh.MbhCurve[2].n=atof(Mclassbh MbhCurve[2].s); //dialog data is string form
Mclassbh MbhCurve[3] n=atof(Mclassbh.MbhCurve[3].s);
Mclassbh.MbhCurve[4].n=atof(Mclassbh.MbhCurve[4].s);

Mclassbh.MbhFlag=2;         //flag set to equation
```

```
 }

}

////
//**** magnet data **** same as CmBH and CmMAGcurve

//////////////////////////////////////////
void
TDbWindow::CmMAG()
{
 TBhDialog* the_pad=new TBhDialog(this, TResId(1));
 for (int loop=0, loop<100, loop++) {
 the_pad->steel_mag=1;
 the_pad->bh[loop].b=Mclassmag Mbh[loop] b,the_pad->bh[loop] h=Mclassmag Mbh[loop] h;}
 strcpy(the_pad->name,Mclassmag MbhName);
 the_pad->point=Mclassmag MbhPoint;
  if (the_pad->Execute()==IDOK){

                for(int j=0,j<100;j++){ Mclassmag Mbh[j] b=the_pad->bh[j].b;
                                        Mclassmag Mbh[j].h=the_pad->bh[j].h;}
                                        strcpy(Mclassmag.MbhName,the_pad->name);
                                        Mclassmag.MbhPoint=the_pad->point;
                                        Mclassmag.MbhFlag=1;
                        }
}

//////////////////////////////////////////
void
TDbWindow CmMAGcurve()
{

                struct TransferBuffer{
                char a[15],b[15],n[15],e[15],k[15],name[20];}tb,

                strcpy(tb.a,Mclassmag.MbhCurve[0].s);
                strcpy(tb.b,Mclassmag MbhCurve[1].s);
                strcpy(tb.n,Mclassmag MbhCurve[2].s);
                strcpy(tb.e,Mclassmag.MbhCurve[3] s);
                strcpy(tb.k,Mclassmag MbhCurve[4] s);
                strcpy(tb.name,Mclassmag MbhName);


  TBhCurDialog* the_pad=new TBhCurDialog(this, TResId(4));
  the_pad->steel_mag=1;
  the_pad->SetTransferBuffer(&tb);
  if (the_pad->Execute()==IDOK){
  strcpy(Mclassmag MbhCurve[0].s,tb.a);
  strcpy(Mclassmag MbhCurve[1].s,tb b);
  strcpy(Mclassmag MbhCurve[2].s,tb.n);
  strcpy(Mclassmag.MbhCurve[3].s,tb.e);
  strcpy(Mclassmag.MbhCurve[4].s,tb.k);
  strcpy(Mclassmag MbhName,tb.name);

  Mclassmag MbhCurve[0].n=atof(Mclassmag.MbhCurve[0].s);
  Mclassmag.MbhCurve[1].n=atof(Mclassmag.MbhCurve[1].s);
  Mclassmag.MbhCurve[2].n=atof(Mclassmag.MbhCurve[2].s);
  Mclassmag MbhCurve[3].n=atof(Mclassmag MbhCurve[3].s);
  Mclassmag.MbhCurve[4].n=atof(Mclassmag.MbhCurve[4].s);

  Mclassmag MbhFlag=2;
  }
  ///get offset
  struct TransferBufferr{
                char a[15],}cb;

                strcpy(cb a,Mclassmag.offset.s);

                TBhOffDialog* the_padd=new TBhOffDialog(this, TResId(5));

the_padd->SetTransferBuffer(&cb);
if (the_padd->Execute()==IDOK){
strcpy(Mclassmag offset.s,cb a);
Mclassmag offset.n=atof(Mclassmag.offset.s);
}

}
//////////////////////////////////////////
///***function to deal with dimension input window***//
void
TDbWindow::CmDim()
{
 int loop;
 do ( loop=0;
  struct TransferBuffer{
  char a[10],b[10],c[10],d[10],e[10],f[10],g[10],h[10],
                                i[10],j[10],k[10],l[10],m[10],n[10],o[10],p[10],
                                q[10],r[10],s[10],t[10];
  }tb;

  strcpy(tb.a,MclassGEO.nsDIM[0].s);strcpy(tb.b,MclassGEO.nsDIM[1].s);
  strcpy(tb.c,MclassGEO.nsDIM[2].s);strcpy(tb.d,MclassGEO.nsDIM[3].s);
  strcpy(tb.e,MclassGEO.nsDIM[4].s);strcpy(tb f,MclassGEO.nsDIM[5] s);
```

```
strcpy(tb.g,MclassGEO.nsDIM[6].s);strcpy(tb.h,MclassGEO.nsDIM[7].s);
strcpy(tb.i,MclassGEO.nsDIM[8].s);strcpy(tb.j,MclassGEO.nsDIM[9].s);
strcpy(tb.k,MclassGEO.nsDIM[10].s),strcpy(tb.l,MclassGEO.nsDIM[11].s);
strcpy(tb.m,MclassGEO.nsDIM[12].s);strcpy(tb.n,MclassGEO.nsDIM[13].s);
strcpy(tb.o,MclassGEO.nsDIM[14].s),strcpy(tb.p,MclassGEO.nsDIM[15].s);
strcpy(tb.q,MclassGEO.nsDIM[16].s),strcpy(tb.r,MclassGEO.nsDIM[17].s);
strcpy(tb.s,MclassGEO.nsDIM[18].s);strcpy(tb.t,MclassGEO.nsDIM[19].s);


TDimDialog* dim_pad=new TDimDialog(this, TResId(2)); //ready to call dialog

                                                                                    //window with ID 2
dim_pad->SetTransferBuffer(&tb);
if (dim_pad->Execute()==IDOK){          //if OK pressed

MclassGEO.transfer(tb.a,tb.b,tb.c,tb.d,tb.e,tb.f,
                   tb.g,tb.h,tb.i,tb.j,tb.k,tb.l,tb.m,tb.n,tb.o,tb.p,tb.q,tb.r,tb.s,tb.t);}
                    //transfer function of GEO class

//check to see if any dimensions missing or zero value
int checka=MclassGEO.zerochk();//call function to check
DimCHK=checka, //Main window check
if (checka!=0){                 //if dimensions missing
MessageBeep(-1);//call error box and makes user re-enter dialog

MessageBox("Dimension missing or of zero value!");
loop=1;
            }
}while (loop!=0);

}

//////////////////////////////////////////
///***function to deal with flux density input window***//
void
TDbWindow::CmDen()
{
int loop;
do {  loop=0;
 struct TransferBuffer{  //set up transfer buffer
 char fd[10];
 }tb;
 //put function which copies
 strcpy(tb.fd,MclassGEO.maxFD.s),

 TDenDialog* dim_pad=new TDenDialog(this, TResId(3));
 dim_pad->SetTransferBuffer(&tb);

 if (dim_pad->Execute()==IDOK){ //when ok pressed
 strcpy(MclassGEO.maxFD.s,tb.fd);}
 MclassGEO.maxFD.n=atof(MclassGEO.maxFD.s);
 //check to see if flux density missing or zero value
 int checka=MclassGEO.zerochkFD(MclassGEO.maxFD); FdCHK=checka;
 if (checka!=0){
 MessageBeep(-1);
 MessageBox("Flux Density value missing or of zero value!");
 loop=1;
            }
 }while (loop!=0);

}
//////////////////////////////////////////
//////////////////////////////////////////
///***function to deal with packing factor input window***//
void
TDbWindow::CmPf()
{
int loop;
 do {  loop=0;
  struct TransferBuffer{  //set up transfer buffer
  char pf[10];
  }tb;
  //put function which copies
  strcpy(tb.pf,MclassGEO.PF.s);

  TDenDialog* dim_pad=new TDenDialog(this, TResId(8));
  dim_pad->SetTransferBuffer(&tb);

  if (dim_pad->Execute()==IDOK){ //when ok pressed
  strcpy(MclassGEO.PF.s,tb.pf);}
  MclassGEO.PF.n=atof(MclassGEO.PF.s);
  //check to see if flux density missing or zero value
  int checka=MclassGEO.zerochkFD(MclassGEO.PF); PfCHK=checka;
  if (checka!=0){
  MessageBeep(-1);
  MessageBox("Packing factor value missing or of zero value!");
  loop=1;
             }
  }while (loop!=0);

}
//////////////////////////////////////////
void TDbWindow::CmPath()
{
if (FdCHK!=0 || DimCHK!=0 || BhCHK==0){ //is data ready
MessageBox("Dimension, BH data, or maximum Flux Density value missing!");
```

```
}
else{ graphyes=1;
MessageBox(
"Path lengths and areas calculations started\n This may take a little time"
,"PATHS",
MB_ICONINFORMATION);
MessageBeep(-1);
//timer
CursorCounter %= 1;
                SetCapture();
                ::SetCursor(LoadCursor(NULL, SystemCursors[CursorCounter]));
//

minAREA=condtm.mdata(MclassGEO); //calls function to convert data to be read by

//paths
if (minAREA==0)
MessageBox("minarea","PATHS",MB_ICONINFORMATION);

int DESTROY=0, //run position calculation in loop at an attempt to save memory
do {////////////*********************////////////////////
//////////// ***  calculate aligned position ***///////
path1.A alA,alB,alC,alD,alE; //create 5 paths
;
alA.Apath1(condtm);alB.Apath2(condtm);alC.Apath3(condtm),
alD.Apath2(condtm);alE.Apath1(condtm);
//calculate area and lengths of steel paths

alA.Atooth1(condtm);alB Atooth2(condtm);alC.Atooth3(condtm);
alD.Atooth4(condtm);alE.Atooth5(condtm);
//calculate area and lengths of airgap paths
MessageBeep(-1);
alA airgapREL();alB.airgapREL(),alC.airgapREL();alD.airgapREL();
alE.airgapREL();

if (alA.AGrel==0 || alB.AGrel==0 || alC.AGrel==0
|| alD.AGrel==0 || alE.AGrel==0)
MessageBox("air gap","PATHS",MB_ICONINFORMATION);
//calculate reluctance of airgap paths

if (MclassGEO.maxFD.n==0)
MessageBox("flux density","PATHS",MB_ICONINFORMATION);

//fill flux table of class MASTER MalFM
MalFM.fluxset(5,minAREA,MclassGEO.maxFD);
if (MalFM.table[DIS-1].f==0)
MessageBox("flux table","PATHS",MB_ICONINFORMATION);

if (BhCHK==1) { //calculate mmf by tables method
 double ALarray[5];
                ALarray[0]=alA.calmmf(Mclassbh.Mbh, MalFM.table,Mclassbh.MbhPoint);
                ALarray[1]=alB.calmmf(Mclassbh Mbh, MalFM.table,Mclassbh.MbhPoint);
                ALarray[2]=alC.calmmf(Mclassbh.Mbh, MalFM.table,Mclassbh.MbhPoint);
                ALarray[3]=alD.calmmf(Mclassbh.Mbh, MalFM.table,Mclassbh.MbhPoint);
                ALarray[4]=alE.calmmf(Mclassbh.Mbh, MalFM table,Mclassbh.MbhPoint);
//find average
                double AVE=(ALarray[0]+ALarray[1]+ALarray[2]+ALarray[3]+ALarray[4])/5,



                if (AVE==0)
MessageBox("AVE","PATHS",MB_ICONINFORMATION);

//send average to compute mmf table
                MalFM.mmfset(AVE);
if (MalFM.table[DIS-1].m==0)
MessageBox("mmf table","PATHS",MB_ICONINFORMATION);
//////////(error in mmf table)

                alA.newflux(MalFM.table,MalFM.tmpflux);//calculates the flux
                alB.newflux(MalFM.table,MalFM.tmpflux);//in each path at the
                alC.newflux(MalFM.table,MalFM.tmpflux);//final mmf, and sums
                alD.newflux(MalFM.table,MalFM tmpflux);//the values in tmpflux
                alE.newflux(MalFM.table,MalFM.tmpflux);

//final table, replace table.f by tmpflux readings.
MalFM finalupdate();
if (MalFM.table[1] f==0)
MessageBox("error","PATHS",MB_ICONINFORMATION);

MessageBeep(-1);
}
else{//calculate mmf by curve fitting
double store[DIS]; //store for passing arrays to curve fitting
for (int s=0; s<DIS; s++) store[s]=MalFM.table[s].f;
double ALarray[5];
ALarray[0]=alA.eqummf(Mclassbh MbhCurve, MalFM.table);
//alA.Pathmmf=hart(alA.MMF,store,0);
ALarray[1]=alB.eqummf(Mclassbh.MbhCurve, MalFM.table);
//alB.Pathmmf=hart(alB.MMF,store,0);
ALarray[2]=alC.eqummf(Mclassbh.MbhCurve, MalFM.table);
//alC.Pathmmf=hart(alC.MMF,store,0);
ALarray[3]=alD.eqummf(Mclassbh.MbhCurve, MalFM.table);
//alD.Pathmmf=hart(alD.MMF,store,0);
ALarray[4]=alE eqummf(Mclassbh MbhCurve, MalFM.table);
```

```
//alE.Pathmmf=hart(alE MMF,store,0);

//find average
                //by putting maximum flux value in each of the paths equations and then
//averaging the results
double AVE=(ALarray[0]+ALarray[1]+ALarray[2]+ALarray[3]+ALarray[4])/5;

//send average to compute mmf table
                MalFM.mmfset(AVE);

/*              alA.eqnewflux(MalFM.table,MalFM.tmpflux),//calculates the flux
                alB.eqnewflux(MalFM.table,MalFM.tmpflux);//in each path at the
                alC.eqnewflux(MalFM.table,MalFM.tmpflux),//final mmf, and sums
                alD.eqnewflux(MalFM.table,MalFM.tmpflux);//the values in tmpflux
                alE.eqnewflux(MalFM.table,MalFM.tmpflux);
 */
                alA.newflux(MalFM.table,MalFM.tmpflux);//calculates the flux
                alB.newflux(MalFM table,MalFM.tmpflux);//in each path at the
                alC.newflux(MalFM table,MalFM tmpflux);//final mmf, and sums
                alD.newflux(MalFM.table,MalFM.tmpflux);//the values in tmpflux
                alE.newflux(MalFM.table,MalFM.tmpflux),

//final table, replace table f by tmpflux readings
MalFM.finalupdate();
//need an equation to model relationship
static double storem[DIS],
for (int kk=0; kk<DIS; kk++){
                                        storem[kk]=MalFM.table[kk].m;
                                        store[kk]=MalFM table[kk].f;}
MalFM.equ=hart(storem,store,0);
//int poo=store[0];

}

}while (DESTROY!=0);
do {//////////////**********////////////////////
//////////// *** calculate mid position ***///////
pathLA mA,mB,mC,mD,mE; //create 5 paths

mA.Mpath1(condim);mB.Mpath2(condim);mC.Mpath3(condim);
mD.Mpath2(condim);mE.Mpath1(condim);
//calculate area and lengths of steel paths

mA.Mtooth1(condim),mB.Mtooth2(condim);mC.Mtooth3(condim);
mD.Mtooth5(condim);mE.Atooth4(condim);
//calculate area and lengths of airgap paths
MessageBeep(-1);
mA.airgapREL();mB.airgapREL();mC.airgapREL();mD.airgapREL();
mE airgapREL(),
//calculate reluctance of airgap paths

//fill flux table of class MASTER MalFM
MmidFM.fluxset(6,mmAREA,MclassGEO.maxFD);

if (BhCHK==1) { //calculate mmf by tables method
 double ALarray[5];
                ALarray[0]=mA.calmmf(Mclassbh.Mbh, MmidFM.table,Mclassbh.MbhPoint);
                ALarray[1]=mB.calmmf(Mclassbh Mbh, MmidFM.table,Mclassbh.MbhPoint);
                ALarray[2]=mC.calmmf(Mclassbh.Mbh, MmidFM.table,Mclassbh.MbhPoint);
                ALarray[3]=mD.calmmf(Mclassbh.Mbh, MmidFM.table,Mclassbh.MbhPoint);
                ALarray[4]=mE.calmmf(Mclassbh.Mbh, MmidFM.table,Mclassbh.MbhPoint);


                //find average
double AVE=(ALarray[0]+ALarray[1]+ALarray[2]+ALarray[3]+ALarray[4])/5;

//send average to compute mmf table
                MmidFM.mmfset(AVE);

                mA.newflux(MmidFM.table,MmidFM.tmpflux);//calculates the flux
                mB.newflux(MmidFM.table,MmidFM.tmpflux);//in each path at the
                mC.newflux(MmidFM.table,MmidFM.tmpflux);//final mmf, and sums
                mD.newflux(MmidFM.table,MmidFM.tmpflux),//the values in tmpflux
                mE.newflux(MmidFM.table,MmidFM.tmpflux);

//final table, replace table.f by tmpflux readings.
MmidFM.finalupdate();
//MessageBeep(-1);
}
else{//calculate mmf by curve fitting
double store[DIS]; //store for passing arrays to curve fitting
for (int s=0; s<DIS, s++) store[s]=MmidFM.table[s].f;
double ALarray[5];
ALarray[0]=mA.eqummf(Mclassbh.MbhCurve, MmidFM.table);
//mA.Pathmmf=hart(mA.MMF,store,1);
ALarray[1]=mB.eqummf(Mclassbh.MbhCurve, MmidFM.table);
//mB.Pathmmf=hart(mB.MMF,store,1);
ALarray[2]=mC.eqummf(Mclassbh.MbhCurve, MmidFM.table);
//mC.Pathmmf=hart(mC.MMF,store,1);
ALarray[3]=mD eqummf(Mclassbh.MbhCurve, MmidFM.table);
//mD.Pathmmf=hart(mD.MMF,store,1);
ALarray[4]=mE.eqummf(Mclassbh.MbhCurve, MmidFM.table);
//mE.Pathmmf=hart(mE.MMF,store,1);

//find average
```

```
                //by putting maximum flux value in each of the paths equations and then
//averaging the results
double AVE=(ALarray[0]+ALarray[1]+ALarray[2]+ALarray[3]+ALarray[4])/5;

//send average to compute mmf table
                MmidFM.mmfset(AVE);

 //             mA.eqnewflux(MmidFM.table,MmidFM.tmpflux);//calculates the flux
                //mB.eqnewflux(MmidFM.table,MmidFM.tmpflux);//in each path at the
                //mC.eqnewflux(MmidFM.table,MmidFM.tmpflux);//final mmf, and sums
                //mD.eqnewflux(MmidFM table,MmidFM.tmpflux);//the values in tmpflux
                //mE.eqnewflux(MmidFM.table,MmidFM.tmpflux);


 mA.newflux(MmidFM.table,MmidFM.tmpflux);//calculates the flux
                mB.newflux(MmidFM table,MmidFM tmpflux);//in each path at the
                mC.newflux(MmidFM.table,MmidFM.tmpflux);//final mmf, and sums
                mD.newflux(MmidFM.table,MmidFM.tmpflux);//the values in tmpflux
                mE.newflux(MmidFM.table,MmidFM.tmpflux);

//final table, replace table.f by tmpflux readings.
MmidFM.finalupdate();
//need an equation to model relationship
static double storem[DIS];
for (int kk=0; kk<DIS, kk++){
                                        storem[kk]=MmidFM.table[kk].m;
                                        store[kk]=MmidFM table[kk] f,}
MmidFM equ=hart(storem,store,1);
//int poo=store[0],

}
}while (DESTROY!=0),
MessageBeep(-1);
do {//////////////**********////////////////
////////// *** calculate un-aligned position ***//////
pathLA unA,unB,unC,unD,unE,unF,unG,unH,unI,unJ; //create 10 paths

unA.Upath1(condim);unB.Upath2(condim);unC.Upath3(condim);unD.Upath4(condim);
unE Upath5(condim);

unF.Upath6(condim);unG.Upath7(condim);unH.Upath8(condim);
unI.Upath9(condim);unJ.Upath10(condim);
//calculate area and lengths of steel paths

unA.Utooth3(condim);unB Utooth4(condim);unC.Utooth1(condim);unD.Utooth2(condim);
unE.Utooth1(condim);

unF.Utooth2(condim);unG.Utooth1(condim);unH.Utooth2(condim);
unI.Utooth6(condim);unJ.Utooth5(condim);
//calculate area and lengths of airgap paths
 MessageBeep(-1);
unA.airgapREL();unB.airgapREL();unC.airgapREL();unD.airgapREL();
unE.airgapREL();

unF.airgapREL();unG.airgapREL(),unH.airgapREL();
unI.airgapREL();unJ.airgapREL();
//calculate reluctance of airgap paths

//fill flux table of class MASTER MalFM
MunFM.fluxset(10,minAREA,MclassGEO.maxFD);

if (BhCHK==1) { //calculate mmf by tables method
  double UNarray[10],

UNarray[0]=unA.calmmf(Mclassbh.Mbh,MunFM table,Mclassbh.MbhPoint);
UNarray[1]=unB calmmf(Mclassbh.Mbh,MunFM table,Mclassbh.MbhPoint);
UNarray[2]=unC.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[3]=unD.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[4]=unE.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);

UNarray[5]=unF.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[6]=unG.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[7]=unH.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[8]=unI.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[9]=unJ.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);

//find average
                double AVE=(UNarray[0]+UNarray[1]+UNarray[2]+UNarray[3]+UNarray[4]
                +UNarray[5]+UNarray[6]+UNarray[7]+UNarray[8]+UNarray[9])/10;


//send average to compute mmf table
                MunFM.mmfset(AVE);

                unA.newflux(MunFM.table,MunFM.tmpflux);//calculates the flux
                unB.newflux(MunFM.table,MunFM.tmpflux);//in each path at the
                unC.newflux(MunFM.table,MunFM.tmpflux);//final mmf, and sums
                unD.newflux(MunFM.table,MunFM.tmpflux);//the values in tmpflux

                unE.newflux(MunFM.table,MunFM.tmpflux);
                unF.newflux(MunFM.table,MunFM.tmpflux);
                unG.newflux(MunFM.table,MunFM tmpflux);
                unH.newflux(MunFM table,MunFM tmpflux);
                unI.newflux(MunFM.table,MunFM.tmpflux);
                unJ.newflux(MunFM.table,MunFM tmpflux);
```

```
//final table, replace table.f by tmpflux readings.
MunFM.finalupdate();
MessageBeep(-1);
}
else{//calculate mmf by curve fitting
double store[DIS]; //store for passing arrays to curve fitting
for (int s=0; s<DIS; s++) store[s]=MunFM.table[s] f;
double unarray[10];
unarray[0]=unA.eqummf(Mclassbh.MbhCurve, MunFM.table);
//unA.Pathmmf=hart(unA.MMF,store,2);
unarray[1]=unB.eqummf(Mclassbh.MbhCurve, MunFM.table);
//unB.Pathmmf=hart(unB.MMF,store,2);
unarray[2]=unC.eqummf(Mclassbh.MbhCurve, MunFM.table);
//unC.Pathmmf=hart(unC.MMF,store,2),
unarray[3]=unD.eqummf(Mclassbh.MbhCurve, MunFM.table);
//unD.Pathmmf=hart(unD.MMF,store,2);
unarray[4]=unE.eqummf(Mclassbh.MbhCurve, MunFM.table);
//unE.Pathmmf=hart(unE.MMF,store,2);
unarray[5]=unF.eqummf(Mclassbh.MbhCurve, MunFM table);
//unF.Pathmmf=hart(unF.MMF,store,2);
unarray[6]=unG.eqummf(Mclassbh.MbhCurve, MunFM.table);
//unG.Pathmmf=hart(unG.MMF,store,2);
unarray[7]=unH.eqummf(Mclassbh MbhCurve, MunFM.table);
//unH Pathmmf=hart(unH.MMF,store,2);
unarray[8]=unI.eqummf(Mclassbh.MbhCurve, MunFM.table);
//unI Pathmmf=hart(unI.MMF,store,2);
unarray[9]=unJ.eqummf(Mclassbh.MbhCurve, MunFM.table);
//unJ.Pathmmf=hart(unJ.MMF,store,2);

//find average
        //by putting maximum flux value in each of the paths equations and then
//averaging the results
double AVE=(unarray[0]+unarray[1]+unarray[2]+unarray[3]+unarray[4]
                                        +unarray[5]+unarray[6]+unarray[7]+unarray[8]+unarray[9])/10;

//send average to compute mmf table
        MunFM.mmfset(AVE);

 /*     unA.eqnewflux(MunFM.table,MunFM.tmpflux);//calculates the flux
        unB eqnewflux(MunFM.table,MunFM.tmpflux);//in each path at the
        unC.eqnewflux(MunFM.table,MunFM.tmpflux);//final mmf, and sums
        unD.eqnewflux(MunFM table,MunFM.tmpflux);//the values in tmpflux
        unE eqnewflux(MunFM.table,MunFM.tmpflux);
        unF.eqnewflux(MunFM.table,MunFM.tmpflux);//calculates the flux
        unG.eqnewflux(MunFM.table,MunFM.tmpflux);//in each path at the
        unH.eqnewflux(MunFM.table,MunFM.tmpflux);//final mmf, and sums
        unI.eqnewflux(MunFM.table,MunFM tmpflux);//the values in tmpflux
        unJ.eqnewflux(MunFM.table,MunFM.tmpflux);
        */
        unA.newflux(MunFM.table,MunFM tmpflux);//calculates the flux
        unB.newflux(MunFM.table,MunFM.tmpflux);//in each path at the
        unC.newflux(MunFM table,MunFM.tmpflux);//final mmf, and sums
        unD.newflux(MunFM.table,MunFM tmpflux);//the values in tmpflux

        unE.newflux(MunFM.table,MunFM.tmpflux);
        unF.newflux(MunFM table,MunFM.tmpflux);
        unG.newflux(MunFM.table,MunFM.tmpflux);
        unH.newflux(MunFM.table,MunFM.tmpflux);
        unI.newflux(MunFM table,MunFM.tmpflux);
        unJ.newflux(MunFM.table,MunFM.tmpflux);


//final table, replace table f by tmpflux readings.
MunFM.finalupdate();
//need an equation to model relationship
static double storem[DIS];
for (int kk=0; kk<DIS; kk++){
                                        storem[kk]=MunFM table[kk].m;
                                        store[kk]=MunFM table[kk].f;}
MunFM.equ=hart(storem,store,2);
//int poo=store[0];

}
}while (DESTROY!=0);

//timer
        ReleaseCapture();
        ::SetCursor(LoadCursor(NULL, IDC_ARROW));


MessageBox(
"Airgap Reluctance is computed \n          Arrays Built"
,"PATHS",
MB_ICONINFORMATION);
graphyes=0;
 }
}
////////////////////////////////////////////
void TDbWindow::CmDrawGeo(){
drawgeo pic;
pic.draw(MclassGEO);


        TGeoFrame * ThisChild;
                ThisChild = new TGeoFrame(this, "GEO");
```

```
//backiron.cpp
#include "com.h"
#include "geo.h"
#include "bh.h"

const float MU0=1.2566e-6;  //permability of free space

double BImag(NOST bh[5],NOST mag[5],FluxMmf FMbimag[DIS],GEO MclassGEO,
classBH Mclassmag)
{
//get dimensions
//length of stack
//outer diameter
//inner diameter

//maximum flux density in back iron=1 T
double AREAback=1e-6*M_PI*(
(MclassGEO.nsDIM[18].n/2)*(MclassGEO.nsDIM[18].n/2)-
(MclassGEO.nsDIM[19].n/2)*(MclassGEO.nsDIM[19].n/2)),
double kper=0.75;
double LenCorr=0.5e-3*(MclassGEO.nsDIM[15].n+(2*kper*(MclassGEO.nsDIM[17].n-
                                        MclassGEO.nsDIM[15].n)));
double LENmag=1e-3*MclassGEO.nsDIM[15].n;
double AREAmag=1e-6*MclassGEO.nsDIM[16].n;
double OFFSETmag=Mclassmag.offset.n;

double MXBKflux=1*AREAback;
float PF=MclassGEO.PF.n;

FMbimag[0].f=0;
FMbimag[0].m=-1*OFFSETmag*LENmag;
double BTMP;
double con;
for (int i=1; i<DIS; i++){

FMbimag[i].f=i*(MXBKflux/(DIS-1));

BTMP=FMbimag[i].f/AREAback;
////////////////////////////////////
double xi=0;
double xu=128e3,
double xr;
double xk,
double yi;
double yr;
int count=0,
do{
        xr=(xi+xu)/2;
    yi=(
                (bh[0].n*pow(xi,bh[2].n)+bh[4].n*pow(xi,bh[2].n+1))/
                        (bh[1].n+pow(xi,bh[2].n)+bh[3].n*pow(xi,bh[2].n+1)))-BTMP;


    yr=(
                (bh[0] n*pow(xr,bh[2].n)+bh[4].n*pow(xr,bh[2].n+1))/
                        (bh[1].n+pow(xr,bh[2].n)+bh[3].n*pow(xr,bh[2].n+1)))-BTMP;

if (yi*yr<0) xu=xr; else xi=xr;
xk=(xi+xu)/2;
con=fabs(xk-xr)/xk;
if (con<1e-3) count=350;

}while (count<300);
////////////////////////////
double mui=BTMP/xk;
double mucorr=(MU0*mui)/(PF*MU0+(1-PF)*mui);
double Hcorr=BTMP/mucorr;

FMbimag[i].m=Hcorr*LenCorr;


////////////////////////////////////
///magnet
BTMP=FMbimag[i].f/AREAmag;
////////////////////////////////////
double mxi=0; double mxu=128e3; double mxr;
double myi,myr;double mxk;
count=0;
do{
        mxr=(mxi+mxu)/2;

        myi=(
                (mag[0].n*pow(mxi,mag[2].n))/
                        (mag[1].n+pow(mxi,mag[2].n)))-BTMP;


        myr=(
                (mag[0].n*pow(mxr,mag[2].n))/
                        (mag[1].n+pow(mxr,mag[2].n)))-BTMP;

if (myi*myr<0) mxu=mxr; else mxi=mxr;
mxk=(mxi+mxu)/2;
```

```
con=fabs(mxk-mxr)/mxk;
if (con<1e-3) count=350;

}while (count<300);
//////////////////////////
double mHcorr=mxk-OFFSETmag;

                                        FMbimag[i].m+=mHcorr*LENmag+fabs(FMbimag[0].m);

////////////////////////////////
}
                                        double value=fabs(FMbimag[0].m);
                                        FMbimag[0].m=0;
                                        return value;
}

//find H from equation

//convert to flux
//use equation to generate new mu and hence mmf
//obtain mmf

//generate equation   //done


//generate magnet mmf/flux charcteristics


//merge backiron and magnet

//create new equation for relation ship
```

```
#include"com.h"
#include"geo.h"
#include"bh.h"
#include"mast.h"
//** non windows classes
void classBH::clearhead() {
                for(int i=0;i<100;i++) {Mbh[i].b=Mbh[i].h=0;}
                MbhPoint=0;
                char blank[]=" ";
                for (i=0; i<5; i++) {MbhCurve[i].n=0;
                strcpy(MbhCurve[i].s,blank);}
                MbhFlag=0;
                strcpy(MbhName,blank);
                strcpy(offset.s,blank);
  offset.n=0;
                }

IMPLEMENT_CASTABLE( classBH );
IMPLEMENT_STREAMABLE( classBH );

void *classBH::Streamer::Read( ipstream& in, uint32 ) const          //replacing default read
{                                                                                                                    //stores all of class in stream
                for (int i=0; i<100; i++) {
                in >>GetObject()->Mbh[i].b;
                in >>GetObject()->Mbh[i].h,} // BH curve in use
                in >> GetObject()->MbhName[20],//BH name to printed on main window;
                in >> GetObject()->MbhPoint; //number of points
                in >> GetObject()->MbhFlag,// 0-no data 1-tables 2-coefficients
                for (i=0; i<5; i++) {
                                in >> GetObject()->MbhCurve[i].n;
                                in >> GetObject()->MbhCurve[0].s[20];}

                return GetObject();
}

//replacing default write command
void classBH::Streamer::Write( opstream& out ) const
{

                for (int i=0; i<100; i++) {
                out << GetObject()->Mbh[i] b;
                out << GetObject()->Mbh[i].h;} // BH curve in use
                out << GetObject()->MbhName[20],//BH name to printed on main window;
                out << GetObject()->MbhPoint; //number of points
                out << GetObject()->MbhFlag,// 0-no data 1-tables 2-coefficients
                for (i=0; i<5; i++) {
                                out << GetObject()->MbhCurve[i].n;
                                out << GetObject()->MbhCurve[0].s[20];}
}

//************** END OF classBH ***************************************


void GEO::transfer              //copy transfer buffer strings to main window
(char a[10],char b[10],char c[10],char d[10],
char e[10],char f[10],char g[10],char h[10],char i[10],char j[10],
char k[10],char l[10],char m[10],char n[10],char o[10],char p[10]
,char q[10],char r[10],char s[10],char t[10])

{
strcpy(nsDIM[0] s,a),strcpy(nsDIM[1].s,b),
strcpy(nsDIM[2].s,c),strcpy(nsDIM[3] s,d),
strcpy(nsDIM[4].s,e),strcpy(nsDIM[5].s,f),
strcpy(nsDIM[6].s,g);strcpy(nsDIM[7].s,h);
strcpy(nsDIM[8].s,i);strcpy(nsDIM[9].s,j);
strcpy(nsDIM[10].s,k);strcpy(nsDIM[11] s,l);
strcpy(nsDIM[12].s,m);strcpy(nsDIM[13].s,n),
strcpy(nsDIM[14].s,o),strcpy(nsDIM[15].s,p),
strcpy(nsDIM[16].s,q);strcpy(nsDIM[17].s,r);
strcpy(nsDIM[18].s,s);strcpy(nsDIM[19].s,t);

for (int kk=0; kk<20; kk++)
nsDIM[kk].n=atof(nsDIM[kk].s);
}

void GEO::stringblank(){  //clear strings at startup
char tmp[]="50";
 strcpy(nsDIM[0] s,tmp),char tmpa[]="59.27";
 strcpy(nsDIM[1].s,tmpa);char tmpb[]="20";
 strcpy(nsDIM[2].s,tmpb);char tmpc[]="1.35";
 strcpy(nsDIM[3].s,tmpc); char tmpd[]="0.285";
 strcpy(nsDIM[4].s,tmpd); char tmpe[]="1.17";
 strcpy(nsDIM[5].s,tmpe); char tmpf[]="1.8";
 strcpy(nsDIM[6] s,tmpf); char tmpg[]="1.94";
 strcpy(nsDIM[7].s,tmpg); char tmph[]="0.6";
 strcpy(nsDIM[8] s,tmph); char tmpi[]="0.94";
 strcpy(nsDIM[9].s,tmpi);char tmpj[]="4.243";
 strcpy(nsDIM[10].s,tmpj);char tmpk[]="7.95";
 strcpy(nsDIM[11].s,tmpk);char tmpl[]="3";
 strcpy(nsDIM[12].s,tmpl); char tmpm[]="4.5";
 strcpy(nsDIM[13].s,tmpm); char tmpn[]="0 1125";
 strcpy(nsDIM[14].s,tmpn); char tmpo[]="10";
```

```
strcpy(nsDIM[15].s,tmpo); char tmpp[]="2065 4";
strcpy(nsDIM[16].s,tmpp);
strcpy(nsDIM[17] s,tmp); char tmpq[]="101.52";
strcpy(nsDIM[18] s,tmpq); char tmpr[]="88 92";
strcpy(nsDIM[19].s,tmpr), char tmps[]="2.1";
strcpy(maxFD.s,tmps);char tmpt[]="0 98";
strcpy(PF.s,tmpt);
}

int GEO::zerochk() //sees if any dimensions missing
{
int check=0;
for (int k=0, k<20; k++){
if (nsDIM[k].n>0) {check=0;}else {check=1,}
}
return check;
}

int GEO::zerochkFD(NOST maxFD) //sees if flux density (or pf) has been entered
{
int check=0;
if (maxFD.n>0) {check=0;}else {check=1,}

return check;
}


///master functions
void master::clearhead()
{for (int k=0; k<DIS,k++)table[k].f=table[k].m=tmpflux[k]=0;}

void master::fluxset(float path, double minarea, NOST FD)
//(no. of paths,min area,max flux density)
{
maxflux=(FD.n*minarea)/(path*1.1);
table[0].f=0;
for (int i=1;i<DIS; i++)
table[i].f=(maxflux/(DIS-1))*i;
}

void master::mmfset(double avmmf)//average flux
{

table[0].m=0;
for (int i=1;i<DIS; i++)
table[i].m=(avmmf/(DIS-1))*i;
}
//function to set up mmf array

void master::finalupdate(){//set up flux in final table
table[0].f=0;
for (int i=1;i<DIS; i++) table[i].f=tmpflux[i];}
```

```
//headers
#include "com.h"
#include "classla.h"
//const float OSAm=79.57747155;              //Oersteds to A per metre

double pathLA::eqummf(NOST bh[5],FluxMmf MFLUX[DIS]) //calculate mmf using curve
{
//varvbles
double maxmmf;
int md=0;//position of path
int mc=1;//flux position;
MMF[0]=0;
double FHi, //function intial
double fHi=1; //H initial
double Hi;//H mutal
double Hn; // H new
double Hkeep[7]; //keeps values of H

 double dFHi; //derivative of FHi
 double BTMP; //flux density tmp
 int Fcon; //convergence flag
double con;//convergence percent

int safe=0;
do{
                        MMF[mc]=(MFLUX[mc].f*AGrel); //air gap mmf

                        do {
                    if (AL[md].A!=0)     //if area not equal to zero
                                                {BTMP=(MFLUX[mc].f/AL[md].A);}  //find flux density
                                                                                else {BTMP=0,}
                                Hi=fHi;
                               if (BTMP!=0){
                    do{
                               Fcon=0;
                               FHi=(
                               (bh[0].n*pow(Hi,bh[2].n)+bh[4].n*pow(Hi,bh[2].n+1))/
                                           (bh[1].n+pow(Hi,bh[2].n)+bh[3].n*pow(Hi,bh[2].n+1)))-BTMP;

                               dFHi=(((bh[0].n*bh[2] n*pow(Hi,bh[2].n)/Hi)+((bh[2].n+1)*bh[4].n*pow(Hi,bh[2].n+1)/Hi))/
                                           (bh[1].n+pow(Hi,bh[2].n)+bh[3].n*pow(Hi,bh[2].n+1)))
                                                       -
                                           ((bh[0].n*pow(Hi,bh[2].n)+bh[4].n*pow(Hi,bh[2].n+1))*
                                           ((bh[2].n*pow(Hi,bh[2].n)/Hi)+((1+bh[2].n)*bh[3].n*pow(Hi,bh[2].n+1)/Hi))/
                                           (pow(bh[1].n+pow(Hi,bh[2].n)+bh[3] n*pow(Hi,bh[2].n+1),2)));

                               Hn=Hi-(FHi/dFHi);
                               con=(fabs((Hn-Hi)/Hn))*100;
                               Hi=fabs(Hn);
                               safe++;
                                           if (con<=1e-4){
                                           Fcon=1;
                                           MMF[mc] +=Hi*AL[md] L;
                                           Hkeep[md]=Hi;}
                                           }
                                           while (Fcon==0 && safe<300);
                                           }
                                if (BTMP==0) Hkeep[md]=Hkeep[md-1];
                                safe=0,Fcon=0,
                                md++; //if ok move to next position

                                }while (md<7); //do until seven section complete
                                 md=0;       //reset to first position
                                 mc++;       //increase flux
                        //find lowest value of H for last round
                                fHi=Hkeep[0];
                                for (int z=1; z<7;z++) {if(Hkeep[z]<fHi)fHi=Hkeep[z],}
                                if (fHi==0) fHi=1;
                        }
                        while (mc<DIS ); //do until all fluxs have mmf calculated

                        maxmmf=MMF[DIS-1];//returns highest value of mmf;
                        return maxmmf;
}

//alA.newflux(MalFM.table,MalFM.tmpflux);
void pathLA::eqnewflux(FluxMmf Mmmf[DIS],double flux[DIS])
{
for (int mc=0; mc<DIS; mc++){
                                                flux[mc] +=
                                                            (Pathmmf.a*pow(Mmmf[mc].m,Pathmmf.n))
                                                            /(Pathmmf.b+pow(Mmmf[mc].m,Pathmmf.n));
                                                            }

}
```

```
#include "com.h"

#include "pathdim.h"

#include "drawgeo.h"
const float DtoR=0.017453292;
float circle(float, float);

float circle(float x, float r)
{
float b=sqrt(r*r-x*x);
return b;
}

void drawgeo..draw(GEO dim)
{
float ALPHA=360/dim.nsDIM[0].n,

OS.x=225,
OS y=750;
k=18;
int kag=2,
//stator

SR=dim.nsDIM[9].n;
float a=(0.5*dim.nsDIM[1].n+kag*dim.nsDIM[14].n);
float b=0.5*dim nsDIM[7].n;

float x=a*sin(DtoR*2*ALPHA);
float x1=(a+dim.nsDIM[8].n)*sin(DtoR*2*ALPHA);
float x2=(a+dim nsDIM[12].n)*sin(DtoR*2*ALPHA);
float xx=b*cos(DtoR*2*ALPHA);
float y=a*cos(DtoR*2*ALPHA);
float y1=(a+dim.nsDIM[8] n)*cos(DtoR*2*ALPHA);
float y2=(a+dim.nsDIM[12].n)*cos(DtoR*2*ALPHA);

float yy=b*sin(DtoR*2*ALPHA);
pA.x=OS.x-k*(x+xx);
pA1.x=OS x-k*(x2+xx);
pB.x=OS.x-k*(x-xx);
pB1.x=OS.x-k*(x1-xx);

pJ.x=OS x+k*(x+xx);
pJ1.x=OS.x+k*(x2+xx);

pI1.x=OS.x+k*(x1-xx);

pI.x=OS.x+k*(x-xx);

pA y=pJ.y=OS y-k*(y-yy);
pA1 y=pJ1.y=OS y-k*(y2-yy);
pB y=pI y=OS.y-k*(y+yy),
pB1 y=pI1.y=OS y-k*(y1+yy);


x=a*sin(DtoR*1*ALPHA);
x1=(a+dim nsDIM[8].n)*sin(DtoR*1*ALPHA);

xx=b*cos(DtoR*1*ALPHA);
y=a*cos(DtoR*1*ALPHA),
y1=(a+dim.nsDIM[8] n)*cos(DtoR*1*ALPHA);

yy=b*sin(DtoR*1*ALPHA);
pC.x=OS.x-k*(x+xx);
pC1.x=OS.x-k*(x1+xx);


pD.x=OS.x-k*(x-xx);
pD1.x=OS.x-k*(x1-xx);

pG.x=OS.x+k*(x-xx),
pG1.x=OS.x+k*(x1-xx);

pH.x=OS.x+k*(x+xx);
pH1.x=OS.x+k*(x1+xx);

pC.y=pH.y=OS.y-k*(y-yy);
pD.y=pG.y=OS y-k*(y+yy);
pC1.y=pH1.y=OS.y-k*(y1-yy);
pD1.y=pG1 y=OS y-k*(y1+yy);


pE.x=OS.x-k*b;
pF.x=OS.x+(k*b);
pE1.x=OS.x-k*b;
pF1.x=OS.x+(k*b);


pE.y=pF.y=OS.y-k*(circle(b,a));
pE1.y=pF1 y=OS.y-k*(circle(b,a+dim nsDIM[8].n));

p1.x=OS.x-k*(0.5*dim nsDIM[11] n);
p2.x=OS.x+k*(0.5*dim.nsDIM[11].n);
p1.y=p2.y=OS y-k*(a+dim.nsDIM[8] n+dim.nsDIM[9].n+dim.nsDIM[13].n);
p3.x=p1.x;
p4.x=p2.x;
```

```
p3.y=p4.y=p1.y-k*(dim.nsDIM[10].n);
s1.x=OS.x;
s1.y=p1.y-k*(0.5*dim.nsDIM[10].n);

//rotor
RR=dim.nsDIM[5].n;
a=a-kag*dim.nsDIM[14].n; //rotor od
b=0.5*dim.nsDIM[3].n; //half teeth width;

x=a*sin(DtoR*3*ALPHA);
x1=(a-dim.nsDIM[4].n)*sin(DtoR*3*ALPHA);
x2=(a-dim.nsDIM[4].n-RR-dim.nsDIM[6].n)*sin(DtoR*3*ALPHA);
y=a*cos(DtoR*3*ALPHA);
y1=(a-dim.nsDIM[4].n)*cos(DtoR*3*ALPHA);
y2=(a-dim.nsDIM[4].n-RR-dim.nsDIM[6].n)*cos(DtoR*3*ALPHA);
xx=b*cos(DtoR*3*ALPHA);
yy=b*sin(DtoR*3*ALPHA);

rA.x=OS.x-k*(x+xx);
rB.x=OS.x-k*(x-xx);
rA1.x=OS.x-k*(x1+xx),
rB1.x=OS.x-k*(x1-xx);
rN.x=OS.x+k*(x+xx);
rM.x=OS.x+k*(x-xx);
rN1.x=OS.x+k*(x1+xx);
rM1.x=OS.x+k*(x1-xx),

rA.y=rN.y=OS.y-k*(y-yy);
rB.y=rM.y=OS.y-k*(y+yy);
rA1.y=rN1.y=OS.y-k*(y1-yy);
rB1.y=rM1.y=OS.y-k*(y1+yy);
///
p5.x=OS.x-k*(x2+xx);
p5.y=OS.y-k*(y2-yy),
p6.y=p5.y;
p6.x=OS.x+k*(x2+xx),

//
x=a*sin(DtoR*2*ALPHA);
x1=(a-dim.nsDIM[4].n)*sin(DtoR*2*ALPHA);
y=a*cos(DtoR*2*ALPHA);
y1=(a-dim.nsDIM[4].n)*cos(DtoR*2*ALPHA);
xx=b*cos(DtoR*2*ALPHA);
yy=b*sin(DtoR*2*ALPHA);

rC.x=OS.x-k*(x+xx);
rD.x=OS.x-k*(x-xx);
rC1.x=OS.x-k*(x1+xx);
rD1.x=OS.x-k*(x1-xx);
rL.x=OS.x+k*(x+xx);
rK.x=OS.x+k*(x-xx);
rL1.x=OS.x+k*(x1+xx);
rK1.x=OS.x+k*(x1-xx);

rC.y=rL.y=OS.y-k*(y-yy);
rD.y=rK.y=OS.y-k*(y+yy);
rC1.y=rL1.y=OS.y-k*(y1-yy);
rD1.y=rK1.y=OS.y-k*(y1+yy);
//
x=a*sin(DtoR*1*ALPHA),
x1=(a-dim.nsDIM[4].n)*sin(DtoR*1*ALPHA);
y=a*cos(DtoR*1*ALPHA);
y1=(a-dim.nsDIM[4].n)*cos(DtoR*1*ALPHA);
xx=b*cos(DtoR*1*ALPHA);
yy=b*sin(DtoR*1*ALPHA);

rE.x=OS.x-k*(x+xx);
rF.x=OS.x-k*(x-xx);
rE1.x=OS.x-k*(x1+xx);
rF1.x=OS.x-k*(x1-xx);
rJ.x=OS.x+k*(x+xx);
rI.x=OS.x+k*(x-xx);
rJ1.x=OS.x+k*(x1+xx);
rI1.x=OS.x+k*(x1-xx);

rE.y=OS.y-k*(y-yy);
rF.y=OS.y-k*(y+yy);
rE1.y=OS.y-k*(y1-yy);
rF1.y=OS.y-k*(y1+yy);
rJ.y=OS.y-k*(y-yy);
rI.y=OS.y-k*(y+yy);
rJ1.y=OS.y-k*(y1-yy);
rI1.y=OS.y-k*(y1+yy);

//
rG.x=rG1.x=OS.x-k*b;
rH.x=rH1.x=OS.x+k*b;
rH.y=rG.y=OS.y-k*(circle(b,a));
rH1.y=rG1.y=OS.y-k*(circle(b,a-dim.nsDIM[4].n));


}
```

```
FluxLink getDENT

{
double mmfR;
double mmfl=10;
double mmfu=10.


double fluxbR,fluxdR,fluxR09,fluxR27;


int count=0; int erflag=0;
double con=1;
do{
//BRANCH A (magnet and back-iron relationship)
double MMFcorr=MbimagMMF+(-1*mmfl);
double fluxMAGl=flux(McurveBIMAG,MMFcorr);
double MMFcorr=MbimagMMF+(-1*mmfu);
double fluxMAGu=flux(McurveBIMAG,MMFcorr);

double fluxbl=flux(MalFMe,mmfl);double fluxbu=flux(MalFMe,mmfu);
double fluxcl=flux(MmidFMe,mmfl);double fluxcu=flux(MmidFMe,mmfu);
double fluxdl=flux(MunFMe,mmfl),double fluxdu=flux(MunFMe,mmfu);

double A0l=(fluxbl+2*fluxcl+fluxdl)/4;
double A1l=0.5*(fluxbl-fluxdl);
double A2l=0 25*(fluxbl-fluxcl+fluxdl-fluxcl);

double Fl09=A0l+A1l*cos(2*M_PI*0 9/7.2)+A2l*cos(4*M_PI*0 9/7.2);
double Fl27=A0l+A1l*cos(2*M_PI*2.7/7.2)+A2l*cos(4*M_PI*2.7/7.2);

double A0u=(fluxbu+2*fluxcu+fluxdu)/4;
double A1u=0.5*(fluxbu-fluxdu);
double A2u=0.25*(fluxbu-fluxcu+fluxdu-fluxcu);

double Fu09=A0u+A1u*cos(2*M_PI*0 9/7.2)+A2u*cos(4*M_PI*0.9/7.2);
double Fu27=A0u+A1u*cos(2*M_PI*2 7/7.2)+A2u*cos(4*M_PI*2.7/7.2);

//chk to see with in range;

double Fl=4*Fl09+4*Fl27-fluxMAGl;
double Fu=4*Fu09+4*Fu27-fluxMAGu,

if (erflag==1){
        if(Fl*Fu<0) erflag=0; else mmfu+=50,}

if (erflag!=1){
mmfR=mmfu-(Fu*(mmfl-mmfu))/(Fl-Fu);
MMFcorr=MbimagMMF+(-1*mmfR);
double fluxMAGR=flux(McurveBIMAG,MMFcorr);

fluxbR=flux(MalFMe,mmfR);
double fluxcR=flux(MmidFMe,mmfR);
 fluxdR=flux(MunFMe,mmfR);

double A0R=(fluxbR+2*fluxcR+fluxdR)/4;
double A1R=0.5*(fluxbR-fluxdR);
double A2R=0.25*(fluxbR-fluxcR+fluxdR-fluxcR);

fluxR09=A0R+A1R*cos(2*M_PI*0.9/7.2)+A2R*cos(4*M_PI*0.9/7.2);
fluxR27=A0R+A1R*cos(2*M_PI*2.7/7.2)+A2R*cos(4*M_PI*2.7/7.2);

double FR=4*FR09+4*FR27-fluxMAGR;

double conM;
if (Fl*FR<0) {conM=mmfu; mmfu=mmfR;} else {conM=mmfl; mmfl=mmfR,}

con=fabs((mmfR-conM)/mmfR))*100,

}
count++;
}
while (count<100 && con>1e-5);

FluxLink detent;
detent.al=fluxbR;
detent.un=fluxdR;
detent.mid1=fluxR09;
detent.mid2=fluxR27;
detent.mag=mmfR;

return detent;
}
```

```
// ** HEADERS **
#include "com.h"
#include "bh.h"
#include "classla.h"
#include "mast.h"

#include "windmain.h"
const float OSAm=79.57747155;                //Oersteds to A per metre
//////////////////////
//function
Curve hart(double xx[DIS],double yy[DIS],int path)
{
const int no=10,
double x[no];x[0]=xx[5];x[1]=xx[10];x[2]=xx[15];x[3]=xx[20];x[4]=xx[23];
x[5]=xx[25],x[6]=xx[29],x[7]=xx[33],x[8]=xx[36];x[9]=xx[39];
double y[no];y[0]=yy[5];y[1]=yy[10];y[2]=yy[15],y[3]=yy[20];y[4]=yy[23];
y[5]=yy[25];y[6]=yy[29];y[7]=yy[33],y[8]=yy[36];y[9]=yy[39];
int COUNT;
 //rows columns

// intial guess may do work on intial guess for frolich function!!
Curve RESULT;
double iALP;//yy[DIS-1]*1.1;//y[no-1]*1e4;  cout<<endl<<"Alpha"<<iALP;
double iBET;//xx[DIS-1]/6; //cout<<endl<<"beta"<<iBET;
double iN=1;
double SumSQ=0;
int ErroR=0,

double NEWALP=0;
double NEWBET=0;
double NEWN=0;

static double Sum[4];
static double T1[no][4];
static double T2[3][4],

static double COEFF[3][2][4];//alpha beta
COEFF[0][0][0]=1.05*y[9],COEFF[0][1][0]=1.5*x[9];
COEFF[0][0][1]=2*y[9];COEFF[0][1][1]=3.8*x[9];
COEFF[0][0][2]=1.5*y[9];COEFF[0][1][2]=3.3*x[9];
COEFF[0][0][3]=1.25*y[9];COEFF[0][1][3]=2.5*x[9];

COEFF[1][0][0]=1 05*y[9];COEFF[1][1][0]=1.5*x[9];
COEFF[1][0][1]=2.3*y[9],COEFF[1][1][1]=4*x[9],
COEFF[1][0][2]=1.6*y[9];COEFF[1][1][2]=3.5*x[9];
COEFF[1][0][3]=1.3*y[9];COEFF[1][1][3]=2.5*x[9];

COEFF[2][0][0]=1.05*y[9],COEFF[2][1][0]=1.5*x[9];
COEFF[2][0][1]=1.25*y[9],COEFF[2][1][1]=3.7*x[9];
COEFF[2][0][2]=1.05*y[9],COEFF[2][1][2]=3 1*x[9];
COEFF[2][0][3]=1*y[9],COEFF[2][1][3]=2.5*x[9];

Curve keeper[4];
int move=0,
do {
iALP=COEFF[path][0][move]; iBET=COEFF[path][1][move];iN=1;COUNT=0;
/////
do {
T2[0][0]=T2[0][1]=T2[0][2]=T2[0][3]=T2[1][0]=
T2[1][1]=T2[1][2]=T2[1][3]=T2[2][0]=T2[2][1]=
T2[2][2]=T2[2][3]=0;

for (int i=0; i<no; i++){

                                                T1[i][0]=pow(x[i],iN)/(iBET+pow(x[i],iN));
                                                T1[i][1]=(-iALP*pow(x[i],iN))/pow((iBET+pow(x[i],iN)),2);
                                                T1[i][2]=
                                                (
                                                (iALP*pow(x[i],iN)*log(x[i]))/
                                                (iBET+pow(x[i],iN)))
                                                -
                                                (
                                                (iALP*pow((pow(x[i],iN)),2)*log(x[i]))/
                                                (pow((iBET+pow(x[i],iN)),2))
                                                );
                                                T1[i][3]=y[i]-(
                                                (iALP*pow(x[i],iN))
                                                /(iBET+pow(x[i],iN)));
                                                T2[0][0]+=pow(T1[i][0],2);
                                                T2[0][1]+=T1[i][0]*T1[i][1];
                                                T2[0][2]+=T1[i][0]*T1[i][2];
                                                T2[0][3]+=T1[i][0]*T1[i][3];

                                                T2[1][0]+=T1[i][0]*T1[i][1];
                                                T2[1][1]+=pow(T1[i][1],2);
                                                T2[1][2]+=T1[i][1]*T1[i][2];
                                                T2[1][3]+=T1[i][1]*T1[i][3];

                                                T2[2][0]+=T1[i][0]*T1[i][2];
                                                T2[2][1]+=T1[i][2]*T1[i][1];
                                                T2[2][2]+=T1[i][2]*T1[i][2];
                                                T2[2][3]+=T1[i][2]*T1[i][3];
                                                }

// solve systems of equations for D.
double EQ1[4], EQ2[4];
for ( i=0, i<4, i++) {
                                                EQ1[i]=(T2[1][0]/T2[0][0])*T2[0][i];
```

```
                                        EQ2[i]=T2[1][i]-EQ1[i];}

double EQ3[4];
for (i=0; i<4; i++) {
                                        EQ1[i]=(T2[2][0]/T2[0][0])*T2[0][i];
                                        EQ3[i]=T2[2][i]-EQ1[i];}

double EQ4[4];
for ( i=0; i<4; i++) {
                                        EQ1[i]=(EQ3[1]/EQ2[1])*EQ2[i];
                                        EQ4[i]=EQ3[i]-EQ1[i];}
double D[3];
D[2]=EQ4[3]/EQ4[2];
D[1]=(EQ2[3]-(D[2]*EQ2[2]))/EQ2[1];
D[0]=(T2[0][3]-(D[2]*T2[0][2])-(D[1]*T2[0][1]))/T2[0][0];
//
//minimum value of Q
double alp1, bet1, n1, alp05, bet05, n05;
alp1=iALP+D[0]; bet1=iBET+D[1]; n1=iN+D[2];
alp05=iALP+0.5*D[0]; bet05=iBET+0.5*D[1]; n05=iN+0.5*D[2];
double Q0=0,double Q05=0; double Q1=0;
for (i=0; i<no;i++) {
                Q0+=T1[i][3]*T1[i][3];
                Q1+=pow((y[i]-((alp1*pow(x[i],n1))/(bet1+pow(x[i],n1)))),2);
                Q05+=pow(y[i]-((alp05*pow(x[i],n05))/(bet05+pow(x[i],n05))),2);
                }


if ((Q1-2*Q05+Q0)!=0){
double Vmin=0.5+(0.25*(Q0-Q1)/(Q1-2*Q05+Q0));

NEWALP=iALP+Vmin*D[0];
NEWBET=iBET+Vmin*D[1];
NEWN=iN+Vmin*D[2];
}
else {COUNT=55;}
//cout<<endl<<NEWALP<<" (C) "<<NEWBET<<" A B N "<<NEWN;
ErroR=0;
if (NEWALP<0 || NEWBET<0 || NEWN<0)  {ErroR=1;COUNT=55;}


//find convergance percentage

double Ealp=((NEWALP-iALP)/NEWALP)*100;  Ealp=fabs(Ealp);
double Ebet=((NEWBET-iBET)/NEWBET)*100;  Ebet=fabs(Ebet);
double En=((NEWN-iN)/NEWN)*100;  En=fabs(En);
//cout<<endl<<Ealp<<" converge (c) "<<Ebet<<"  "<<En;

// converge check
if (Ealp<1e-5 && Ebet<1e-5 && En<1e-5) COUNT=55;

iALP=fabs(NEWALP); iBET=fabs(NEWBET); iN=fabs(NEWN);COUNT++;
RESULT.a=NEWALP;RESULT.b=NEWBET;RESULT.n=NEWN;
}while (COUNT<40);
//sum of squares of residuals
if (ErroR==1){SumSQ=1e10;}
else{
SumSQ=0;
for (int i=0; i<no; i++) {

                SumSQ+=pow(
                y[i]-(
                (RESULT.a*pow(x[i],RESULT.n))
                /(RESULT.b+pow(x[i],RESULT.n))),2); }
                }
Sum[move]=SumSQ;
keeper[move].a=RESULT.a;keeper[move].b=RESULT.b;keeper[move].n=RESULT.n;

move++;
}while(move<4);

double low;
low=Sum[4];
for (int z=0; z<3;z++){if (low>Sum[z]) {
                                        low=Sum[z];RESULT.a=keeper[z].a;RESULT.b=keeper[z].b;
                                        RESULT.n=keeper[z].n;       }}

return RESULT;
}
//////////////////////////////////////


//** MENU COMMANDS LINKED to Functions**
DEFINE_RESPONSE_TABLE1(TDbWindow, TFrameWindow)
                EV_WM_PAINT,
                EV_COMMAND(201, CmBH),                          //call BH steel points input
                EV_COMMAND(601, CmBHcurve),     //call BH steel equation input
                EV_COMMAND(701, CmMAG),                         //call BH magnet points input
                EV_COMMAND(801, CmMAGcurve),    //call BH magnet points input
                EV_COMMAND(52, CmBHsave),       //save steel bh data
                EV_COMMAND(51, CmBHload),       //load steel bh data
                EV_COMMAND(301, CmDim),   //calls dimensions input window
```

```
                EV_COMMAND(401, CmDen),    //call flux density window
                EV_COMMAND(501, CmPath).   //starts paths calculation
                EV_COMMAND(901, CmGraph),
                END_RESPONSE_TABLE;


//***** initialisation of main window *****//
TDbWindow::TDbWindow(TWindow* parent, const char far* title)
:TFrameWindow(parent, title),
                TWindow(parent, title)
{
                AssignMenu(1);                  //assign the menu to the screen
                MclassGEO.stringblank(); //clears the strings in class GEO
                BhCHK=0; //bh data is not entered
                DimCHK=1; //sets dimension check to 'dimensions not entered'
                FdCHK=0;  // flux density limit is entered
                graphyes=1;// set flag so no flux/mmf graph is displayed until results

                MclassGEO.maxFD.n=2.1;
                gcvt(MclassGEO.maxFD.n,7,MclassGEO.maxFD.s);

                MalFM.clearhead();
                MmidFM clearhead();
                MunFM.clearhead();
                Mclassbh.clearhead();
                Mclassmag clearhead();
                }

/////
//declare child window for graph
DEFINE_RESPONSE_TABLE1(TGraphFrame, TFrameWindow)
END_RESPONSE_TABLE;

TGraphFrame::TGraphFrame(TWindow* parent, const char far * title)
:TFrameWindow(parent, title)
{
Attr.X=100;
Attr.Y=100;
Attr W=350;
Attr.H=350;
Attr.Style=WS_VISIBLE | WS_CAPTION | WS_BORDER | WS_SYSMENU | WS_MINIMIZEBOX
                                                        |WS_CLIPSIBLINGS | WS_THICKFRAME,
}
//////////////////
//** DECLARE FUNCTIONS FOR TDbWindow (main window) class ****//
void
TDbWindow::EvPaint()
{

TPaintDC paintDC(HWindow); //paint main window
//**
if (Mclassbh.MbhFlag==1 && Mclassmag MbhFlag==1) {BhCHK=1;}
else if (Mclassbh.MbhFlag==2 && Mclassmag MbhFlag==2) {BhCHK=2;}
else {BhCHK=0;}
//***
char blank[]=" ";               //** charcters to tell user what curves have been selected**//
char cur[]=" (CURVE)";                  // with what method
char poi[]=" (POINTS)";
char methodbh[10];
char methodmag[10];
//**            steel BH curve**
if (Mclassbh.MbhFlag==0) {strcpy(methodbh,blank);}      //no points entered no method
else if (Mclassbh.MbhFlag==1) {strcpy(methodbh,poi);}   //points method
else {strcpy(methodbh,cur);}                                            //equation method
//**            magnet BH curve
if (Mclassmag MbhFlag==0) {strcpy(methodmag,blank);}   //no points entered no method
else if (Mclassmag.MbhFlag==1) {strcpy(methodmag,poi);} //points method
else {strcpy(methodmag,cur);}                                                   //equation method
//*********************

//** display names of steel and magnet
char title[]="BH curve in current use :";
paintDC.TextOut(0,10, title, strlen(title));
char holdbh[25];
strcpy(holdbh,Mclassbh.MbhName);
strcat(holdbh,methodbh);
paintDC.TextOut(175,10, holdbh, strlen(holdbh)),

char mtitle[]="Magnet in current use  :";
paintDC.TextOut(0,25, mtitle, strlen(title));
char holdmag[25];
strcpy(holdmag,Mclassmag.MbhName);
strcat(holdmag,methodmag);
paintDC.TextOut(175,25, holdmag, strlen(holdmag));
//*** - check to see if bh have been entered **
if (BhCHK==0){
char chkTMP[]="BH data missing. or of differng techniques!";
paintDC.TextOut(1,46, chkTMP, strlen(chkTMP));}
else{char chkTMP[]="BH data Entered";
paintDC.TextOut(1,46, chkTMP, strlen(chkTMP));}

//*** - check to see if dimensions have been entered **
if (DimCHK!=0){
char chkTMP[]="Dimensions not completed !";
paintDC.TextOut(1,70, chkTMP, strlen(chkTMP));}
else{char chkTMP[]="Dimensions Entered";
```

```
paintDC.TextOut(1,70, chkTMP, strlen(chkTMP)),}

// *** maximum flux density ***
char MXfd[]="Maximum Flux Density is set at : ";
strcat(MXfd,MclassGEO.maxFD.s);
paintDC.TextOut(1,85, MXfd, strlen(MXfd));

}
////////////////////////////////
.
//***** main window function *****//

// ** SAVE BH **
void
TDbWindow::CmBHsave()
{
//** SAVE DIALOG ** looks for .bh files
TOpenSaveDialog::TData data(
OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,
"BH Files (*.BH)|*.BH|",
0,
"",
"BH"
);

//** if file can be saved holds filename in data.filename
if ((new TFileSaveDialog(this, data))->Execute()==IDOK){

                ofpstream out(data.FileName);
                 if(!out) MessageBox("Cannot Open file", "Error", MB_OK);
                                        else{
                                                        out<<Mclassbh; //because class is streamed
                                        }

                }
}

//** load BH **
void
TDbWindow::CmBHload()
{
static TOpenSaveDialog::TData data ( //displays only .BH files
OFN_HIDEREADONLY,
"BH Files (*.BH)|*.BH|",
0,
"",
"BH"
);
if ((new TFileOpenDialog(this, data))->Execute()==IDOK){
                ifpstream in(data.FileName);
                 if(!in) MessageBox("Cannot Open file", "Error", MB_OK);
                                else{
                                                in>>Mclassbh; //load data back into class
                                 }
                }
}
/////// FUNCTIONS FOR BH FILES ENDED //////

//// BH STEEL MANUPLATION ///
void
TDbWindow::CmBH()
{
 TBhDialog* the_pad=new TBhDialog(this, TResId(1)); //calls the Dialog window (ID 1)

//*** loads data from main window ***///

                the_pad->steel_mag=0; // informs dialog window we are dealing with steel data

                for (int loop=0; loop<100; loop++) { //loads bh data to dialog window
the_pad->bh[loop].b=Mclassbh.Mbh[loop].b;the_pad->bh[loop].h=Mclassbh.Mbh[loop].h;}

                strcpy(the_pad->name,Mclassbh.MbhName);//copies the bh name to Dialog
                the_pad->point=Mclassbh.MbhPoint; //copies the number of points

//** when the user presses enter** - return the data back to the main window
if (the_pad->Execute()==IDOK){
                for(int j=0;j<100;j++){ Mclassbh.Mbh[j].b=the_pad->bh[j].b;
                                                Mclassbh.Mbh[j].h=the_pad->bh[j].h;}
                                                strcpy(Mclassbh.MbhName,the_pad->name);
                                                Mclassbh.MbhPoint=the_pad->point;
                                                Mclassbh.MbhFlag=1;
                        }
}

////////////////////////////////////////////
//** function to deal with steel equation data **
void
TDbWindow::CmBHcurve()
{

// creates a transfer buffer to put data directly into windows //
                struct TransferBuffer{
                char a[15],b[15],n[15],e[15],k[15],name[20];}tb;
```

```
                strcpy(tb a,Mclassbh.MbhCurve[0].s);
                strcpy(tb.b,Mclassbh.MbhCurve[1].s);
                strcpy(tb.n,Mclassbh.MbhCurve[2].s);
                strcpy(tb.e,Mclassbh.MbhCurve[3].s);
                strcpy(tb.k,Mclassbh.MbhCurve[4].s);
                strcpy(tb.name,Mclassbh.MbhName);


TBhCurDialog* the_pad=new TBhCurDialog(this, TResId(4)); //creates new dialog window
the_pad->steel_mag=0; //steel data is being dealt with
the_pad->SetTransferBuffer(&tb);
if (the_pad->Execute()==IDOK){
                                                        //return values from dialog to main window data
strcpy(Mclassbh.MbhCurve[0] s,tb a);
strcpy(Mclassbh.MbhCurve[1].s,tb.b);
strcpy(Mclassbh.MbhCurve[2].s,tb.n);
strcpy(Mclassbh.MbhCurve[3] s,tb.e);
strcpy(Mclassbh.MbhCurve[4].s,tb.k);
strcpy(Mclassbh.MbhName,tb.name);

Mclassbh.MbhCurve[0].n=atof(Mclassbh.MbhCurve[0].s); //atof function converts string data to
 Mclassbh.MbhCurve[1].n=atof(Mclassbh.MbhCurve[1].s); //numerical
 Mclassbh.MbhCurve[2].n=atof(Mclassbh.MbhCurve[2].s); //dialog data is string form
 Mclassbh.MbhCurve[3].n=atof(Mclassbh.MbhCurve[3].s);
 Mclassbh.MbhCurve[4].n=atof(Mclassbh.MbhCurve[4].s);

 Mclassbh.MbhFlag=2;        //flag set to equation
 }

}

/////
//**** magnet data **** same as CmBH and CmMAGcurve

////////////////////////////////////////////////
void
TDbWindow::CmMAG()
{
 TBhDialog* the_pad=new TBhDialog(this, TResId(1));
 for (int loop=0; loop<100; loop++) {
 the_pad->steel_mag=1;
 the_pad->bh[loop].b=Mclassmag.Mbh[loop].b,the_pad->bh[loop].h=Mclassmag.Mbh[loop].h,}
 strcpy(the_pad->name,Mclassmag.MbhName);
 the_pad->point=Mclassmag MbhPoint;
  if (the_pad->Execute()==IDOK){

                for(int j=0;j<100;j++){ Mclassmag.Mbh[j].b=the_pad->bh[j].b;
                                                        Mclassmag Mbh[j].h=the_pad->bh[j].h,}
                                                        strcpy(Mclassmag MbhName,the_pad->name);
                                                        Mclassmag.MbhPoint=the_pad->point,
                                                        Mclassmag.MbhFlag=1;
                                        }
}

////////////////////////////////////////////
void
TDbWindow::CmMAGcurve()
{

                struct TransferBuffer{
                char a[15],b[15],n[15],e[15],k[15],name[20];}tb;

                strcpy(tb.a,Mclassmag.MbhCurve[0].s);
                strcpy(tb.b,Mclassmag.MbhCurve[1].s);
                strcpy(tb.n,Mclassmag.MbhCurve[2].s);
                strcpy(tb.e,Mclassmag.MbhCurve[3].s);
                strcpy(tb.k,Mclassmag.MbhCurve[4] s);
                strcpy(tb.name,Mclassmag.MbhName);


 TBhCurDialog* the_pad=new TBhCurDialog(this, TResId(4));
 the_pad->steel_mag=1;
 the_pad->SetTransferBuffer(&tb);
 if (the_pad->Execute()==IDOK){
 strcpy(Mclassmag.MbhCurve[0].s,tb.a);
 strcpy(Mclassmag.MbhCurve[1].s,tb.b);
 strcpy(Mclassmag MbhCurve[2].s,tb.n);
 strcpy(Mclassmag.MbhCurve[3].s,tb e);
 strcpy(Mclassmag.MbhCurve[4].s,tb.k);
 strcpy(Mclassmag.MbhName,tb.name);

 Mclassmag.MbhCurve[0].n=atof(Mclassmag.MbhCurve[0].s);
 Mclassmag.MbhCurve[1].n=atof(Mclassmag.MbhCurve[1].s);
 Mclassmag.MbhCurve[2].n=atof(Mclassmag.MbhCurve[2].s);
 Mclassmag.MbhCurve[3].n=atof(Mclassmag.MbhCurve[3].s);
 Mclassmag.MbhCurve[4].n=atof(Mclassmag.MbhCurve[4].s);

 Mclassmag.MbhFlag=2;
 }
 ///get offset
 struct TransferBuffer{
                char a[15];}cb;

                strcpy(cb.a,Mclassmag offset.s);
```

```
                TBhOffDialog* the_padd=new TBhOffDialog(this, TResId(5));

 the_padd->SetTransferBuffer(&cb);
 if (the_padd->Execute()==IDOK){
 strcpy(Mclassmag offset.s,cb.a);
 }


}
//////////////////////////////////////////
///***function to deal with dimension input window***//
void
TDbWindow::CmDim()
{
 int loop;
 do { loop=0;
  struct TransferBuffer{
  char a[10],b[10],c[10],d[10],e[10],f[10],g[10],h[10],
                                i[10],j[10],k[10],l[10],m[10],n[10],o[10],p[10];
  }tb;

strcpy(tb.a,MclassGEO.nsDIM[0].s);strcpy(tb.b,MclassGEO.nsDIM[1].s);
strcpy(tb.c,MclassGEO.nsDIM[2].s);strcpy(tb.d,MclassGEO.nsDIM[3].s);
strcpy(tb.e,MclassGEO.nsDIM[4].s),strcpy(tb.f,MclassGEO.nsDIM[5].s);
strcpy(tb.g,MclassGEO.nsDIM[6].s),strcpy(tb.h,MclassGEO.nsDIM[7].s);
strcpy(tb.i,MclassGEO.nsDIM[8].s);strcpy(tb.j,MclassGEO.nsDIM[9].s);
strcpy(tb.k,MclassGEO.nsDIM[10].s),strcpy(tb.l,MclassGEO.nsDIM[11].s);
strcpy(tb.m,MclassGEO.nsDIM[12].s);strcpy(tb.n,MclassGEO.nsDIM[13] s);
strcpy(tb.o,MclassGEO.nsDIM[14].s);strcpy(tb.p,MclassGEO.nsDIM[15].s);


TDimDialog* dim_pad=new TDimDialog(this, TResId(2)); //ready to call dialog


                                                                                //window with ID 2
dim_pad->SetTransferBuffer(&tb);
if (dim_pad->Execute()==IDOK){          //if OK pressed

MclassGEO.transfer(tb a,tb.b,tb.c,tb.d,tb.e,tb.f,
                                tb g,tb.h,tb.i,tb.j,tb.k,tb.l,tb.m,tb.n,tb.o,tb p);}
                                 //transfer function of GEO class

//check to see if any dimensions missing or zero value
int checka=MclassGEO.zerochk();//call function to check
DimCHK=checka, //Main window check
if (checka!=0){                 //if dimensions missing
MessageBeep(-1);//call error box and makes user re-enter dialog

MessageBox("Dimension missing or of zero value!");
loop=1;
                }
 }while (loop!=0),

}

//////////////////////////////////////////
///***function to deal with flux density input window***//
void
TDbWindow::CmDen()
{
int loop;
 do { loop=0;
  struct TransferBuffer{  //set up transfer buffer
  char fd[10];
  }tb;
  //put function which copies
  strcpy(tb.fd,MclassGEO.maxFD s);

  TDenDialog* dim_pad=new TDenDialog(this, TResId(3));
  dim_pad->SetTransferBuffer(&tb);

  if (dim_pad->Execute()==IDOK){ //when ok pressed
  strcpy(MclassGEO.maxFD.s,tb.fd);}
  MclassGEO.maxFD.n=atof(MclassGEO.maxFD.s);
  //check to see if flux density missing or zero value
  int checka=MclassGEO.zerochkFD(); FdCHK=checka;
  if (checka!=0){
  MessageBeep(-1);
  MessageBox("Flux Density value missing or of zero value!");
  loop=1;
                }
  }while (loop!=0);

}
//////////////////////////////////////////
void TDbWindow::CmPath()
{
if (FdCHK!=0 || DimCHK!=0 || BhCHK==0){ //is data ready
MessageBox("Dimension, BH data, or maximum Flux Density value missing!");
}
else{  graphyes=1;
MessageBox(
"Path lengths and areas calculations started\n This may take a little time"
,"PATHS",
MB_ICONINFORMATION);
MessageBeep(-1);
```

```
minAREA=condim.indata(MclassGEO); //calls function to convert data to be read by

//paths
if (minAREA==0)
MessageBox("minarea","PATHS",MB_ICONINFORMATION);

int DESTROY=0; //run position calculation in loop at an attempt to save memory
do {//////////////*******************//////////////////
/////////// ***  calculate aligned position ***///////
pathLA alA,alB,alC,alD,alE; //create 5 paths
;
alA.Apath1(condim);alB.Apath2(condim);alC.Apath3(condim);
alD.Apath2(condim);alE.Apath1(condim);
//calculate area and lengths of steel paths

alA.Atooth1(condim);alB.Atooth2(condim);alC.Atooth3(condim);
alD.Atooth4(condim);alE.Atooth5(condim);
//calculate area and lengths of airgap paths
MessageBeep(-1);
alA.airgapREL();alB.airgapREL();alC.airgapREL();alD.airgapREL();
alE.airgapREL();

if (alA.AGrel==0 || alB.AGrel==0 || alC.AGrel==0
|| alD.AGrel==0 || alE.AGrel==0)
MessageBox("air gap","PATHS",MB_ICONINFORMATION);
//calculate reluctance of airgap paths

if (MclassGEO.maxFD.n==0)
MessageBox("flux density","PATHS",MB_ICONINFORMATION);

//fill flux table of class MASTER MalFM
MalFM.fluxset(5,minAREA,MclassGEO.maxFD);
if (MalFM.table[DIS-1].f==0)
MessageBox("flux table","PATHS",MB_ICONINFORMATION);

if (BhCHK==1) { //calculate mmf by tables method
 double ALarray[5];
                ALarray[0]=alA.calmmf(Mclassbh.Mbh, MalFM.table,Mclassbh.MbhPoint);
                ALarray[1]=alB.calmmf(Mclassbh.Mbh, MalFM.table,Mclassbh.MbhPoint);
                ALarray[2]=alC.calmmf(Mclassbh.Mbh, MalFM.table,Mclassbh.MbhPoint);
                ALarray[3]=alD.calmmf(Mclassbh.Mbh, MalFM.table,Mclassbh.MbhPoint);
                ALarray[4]=alE.calmmf(Mclassbh.Mbh, MalFM table,Mclassbh.MbhPoint);
//find average
                double AVE=(ALarray[0]+ALarray[1]+ALarray[2]+ALarray[3]+ALarray[4])/5;



                if (AVE==0)
MessageBox("AVE","PATHS",MB_ICONINFORMATION);
//send average to compute mmf table
                MalFM.mmfset(AVE);
if (MalFM.table[DIS-1].m==0)
MessageBox("mmf table","PATHS",MB_ICONINFORMATION);
//////////(error in mmf table)

                alA.newflux(MalFM table,MalFM.tmpflux);//calculates the flux
                alB.newflux(MalFM.table,MalFM.tmpflux);//in each path at the
                alC.newflux(MalFM.table,MalFM.tmpflux);//final mmf, and sums
                alD.newflux(MalFM table,MalFM.tmpflux);//the values in tmpflux
                alE.newflux(MalFM.table,MalFM.tmpflux);

//final table, replace table.f by tmpflux readings.
MalFM.finalupdate();
if (MalFM.table[1].f==0)
MessageBox("error","PATHS",MB_ICONINFORMATION);

MessageBeep(-1);
}
else{//calculate mmf by curve fitting
double store[DIS]; //store for passing arrays to curve fitting
for (int s=0; s<DIS; s++) store[s]=MalFM.table[s].f;
double ALarray[5];
ALarray[0]=alA.eqummf(Mclassbh.MbhCurve, MalFM table);
alA.Pathmmf=hart(alA.MMF,store,0);
ALarray[1]=alB.eqummf(Mclassbh.MbhCurve, MalFM.table);
alB.Pathmmf=hart(alB.MMF,store,0);
ALarray[2]=alC.eqummf(Mclassbh.MbhCurve, MalFM table);
alC.Pathmmf=hart(alC.MMF,store,0);
ALarray[3]=alD.eqummf(Mclassbh.MbhCurve, MalFM.table);
alD.Pathmmf=hart(alD.MMF,store,0);
ALarray[4]=alE.eqummf(Mclassbh.MbhCurve, MalFM.table);
alE.Pathmmf=hart(alE.MMF,store,0);

//find average
                //by putting maximum flux value in each of the paths equations and then
//averaging the results
double AVE=(ALarray[0]+ALarray[1]+ALarray[2]+ALarray[3]+ALarray[4])/5;

//send average to compute mmf table
                MalFM.mmfset(AVE);

                alA.eqnewflux(MalFM.table,MalFM.tmpflux);//calculates the flux
                alB eqnewflux(MalFM.table,MalFM.tmpflux);//in each path at the
                alC.eqnewflux(MalFM.table,MalFM tmpflux);//final mmf, and sums
                alD.eqnewflux(MalFM.table,MalFM.tmpflux);//the values in tmpflux
```

```
                alE.eqnewflux(MalFM.table,MalFM tmpflux);

//final table, replace table.f by tmp flux readings.
MalFM.finalupdate();
//need an equation to model relationship
static double storem[DIS];
for (int kk=0; kk<DIS; kk++){
                                        storem[kk]=MalFM.table[kk].m;
                                        store[kk]=MalFM.table[kk] f;}
MalFM.equ=hart(storem,store,0);
//int poo=store[0];

}

}while (DESTROY!=0);
do {//////////////*********************//////////////////
//////////// *** calculate mid position ***///////
pathLA mA,mB,mC,mD,mE; //create 5 paths

mA.Mpath1(condim);mB.Mpath2(condim);mC.Mpath3(condim);
mD.Mpath2(condim);mE.Mpath1(condim);
//calculate area and lengths of steel paths

mA.Mtooth1(condim);mB.Mtooth2(condim);mC.Mtooth3(condim);
mD.Mtooth5(condim),mE.Atooth4(condim);
//calculate area and lengths of airgap paths
MessageBeep(-1);
mA.airgapREL();mB.airgapREL();mC.airgapREL();mD.airgapREL();
mE.airgapREL();
//calculate reluctance of airgap paths

//fill flux table of class MASTER MalFM
MmidFM.fluxset(6,mmAREA,MclassGEO.maxFD);

if (BhCHK==1) { //calculate mmf by tables method
 double ALarray[5];
                ALarray[0]=mA.calmmf(Mclassbh.Mbh, MmidFM.table,Mclassbh.MbhPoint);
                ALarray[1]=mB.calmmf(Mclassbh.Mbh, MmidFM.table,Mclassbh.MbhPoint);
                ALarray[2]=mC.calmmf(Mclassbh.Mbh, MmidFM.table,Mclassbh.MbhPoint);
                ALarray[3]=mD.calmmf(Mclassbh.Mbh, MmidFM.table,Mclassbh.MbhPoint);
                ALarray[4]=mE.calmmf(Mclassbh.Mbh, MmidFM.table,Mclassbh.MbhPoint);


                //find average
double AVE=(ALarray[0]+ALarray[1]+ALarray[2]+ALarray[3]+ALarray[4])/5;

//send average to compute mmf table
                MmidFM.mmfset(AVE);

                mA.newflux(MmidFM.table,MmidFM.tmpflux);//calculates the flux
                mB.newflux(MmidFM.table,MmidFM.tmpflux);//in each path at the
                mC.newflux(MmidFM.table,MmidFM.tmpflux);//final mmf, and sums
                mD.newflux(MmidFM table,MmidFM.tmpflux);//the values in tmpflux
                mE.newflux(MmidFM.table,MmidFM.tmpflux);

//final table, replace table.f by tmpflux readings.
MmidFM.finalupdate();
//MessageBeep(-1);
}
else{//calculate mmf by curve fitting
double store[DIS]; //store for passing arrays to curve fitting
for (int s=0; s<DIS; s++) store[s]=MmidFM.table[s].f;
double ALarray[5];
ALarray[0]=mA.eqummf(Mclassbh MbhCurve, MmidFM table);
mA.Pathmmf=hart(mA.MMF,store,1);
ALarray[1]=mB.eqummf(Mclassbh.MbhCurve, MmidFM.table);
mB Pathmmf=hart(mB.MMF,store,1);
ALarray[2]=mC.eqummf(Mclassbh.MbhCurve, MmidFM.table);
mC.Pathmmf=hart(mC.MMF,store,1);
ALarray[3]=mD.eqummf(Mclassbh.MbhCurve, MmidFM.table);
mD.Pathmmf=hart(mD.MMF,store,1);
ALarray[4]=mE.eqummf(Mclassbh.MbhCurve, MmidFM.table);
mE.Pathmmf=hart(mE.MMF,store,1);

//find average
                //by putting maximum flux value in each of the paths equations and then
//averaging the results
double AVE=(ALarray[0]+ALarray[1]+ALarray[2]+ALarray[3]+ALarray[4])/5;

//send average to compute mmf table
                MmidFM.mmfset(AVE);

                mA.eqnewflux(MmidFM.table,MmidFM.tmpflux),//calculates the flux
                mB.eqnewflux(MmidFM table,MmidFM.tmpflux);//in each path at the
                mC.eqnewflux(MmidFM.table,MmidFM.tmpflux),//final mmf, and sums
                mD.eqnewflux(MmidFM.table,MmidFM.tmpflux);//the values in tmpflux
                mE.eqnewflux(MmidFM table,MmidFM tmpflux);

//final table, replace table.f by tmpflux readings.
MmidFM.finalupdate();
//need an equation to model relationship
static double storem[DIS];
for (int kk=0; kk<DIS; kk++){
                                        storem[kk]=MmidFM.table[kk] m;
                                        store[kk]=MmidFM table[kk].f;}
MmidFM.equ=hart(storem,store,1);
```

```
//int poo=store[0];

}
}while (DESTROY!=0);
MessageBeep(-1);
do {//////////////**********////////////////////
////////// *** calculate un-aligned position ***///////
pathLA unA,unB,unC,unD,unE,unF,unG,unH,unI,unJ; //create 10 paths

unA.Upath1(condim);unB Upath2(condim);unC.Upath3(condim);unD.Upath4(condim);
unE.Upath5(condim);

unF.Upath6(condim);unG Upath7(condim);unH.Upath8(condim);
unI.Upath9(condim);unJ.Upath10(condim);
//calculate area and lengths of steel paths

unA.Utooth3(condim);unB.Utooth4(condim);unC.Utooth1(condim);unD.Utooth2(condim);
unE.Utooth1(condim);

unF.Utooth2(condim);unG.Utooth1(condim);unH.Utooth2(condim);
unI.Utooth6(condim);unJ.Utooth5(condim);
//calculate area and lengths of airgap paths
 MessageBeep(-1);
unA.airgapREL();unB.airgapREL();unC.airgapREL();unD.airgapREL();
unE.airgapREL();

unF.airgapREL(),unG.airgapREL();unH.airgapREL();
unI.airgapREL(),unJ.airgapREL();
//calculate reluctance of airgap paths

//fill flux table of class MASTER MalFM
MunFM.fluxset(10,minAREA,MclassGEO.maxFD);

if (BhCHK==1) { //calculate mmf by tables method
  double UNarray[10];

UNarray[0]=unA.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[1]=unB.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[2]=unC.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[3]=unD.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[4]=unE.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);

UNarray[5]=unF.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[6]=unG.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[7]=unH.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[8]=unI.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);
UNarray[9]=unJ.calmmf(Mclassbh.Mbh,MunFM.table,Mclassbh.MbhPoint);

//find average
                double AVE=(UNarray[0]+UNarray[1]+UNarray[2]+UNarray[3]+UNarray[4]
                +UNarray[5]+UNarray[6]+UNarray[7]+UNarray[8]+UNarray[9])/10;



//send average to compute mmf table
                MunFM.mmfset(AVE);

                unA.newflux(MunFM.table,MunFM.tmpflux);//calculates the flux
                unB.newflux(MunFM.table,MunFM.tmpflux);//in each path at the
                unC.newflux(MunFM.table,MunFM.tmpflux);//final mmf, and sums
                unD.newflux(MunFM.table,MunFM.tmpflux);//the values in tmpflux

                unE.newflux(MunFM.table,MunFM.tmpflux);
                unF.newflux(MunFM.table,MunFM.tmpflux);
                unG.newflux(MunFM table,MunFM.tmpflux);
                unH.newflux(MunFM.table,MunFM.tmpflux);
                unI.newflux(MunFM.table,MunFM.tmpflux);
                unJ.newflux(MunFM.table,MunFM.tmpflux);

//final table, replace table.f by tmpflux readings.
MunFM.finalupdate();
MessageBeep(-1);
}
else{//calculate mmf by curve fitting
double store[DIS]; //store for passing arrays to curve fitting
for (int s=0; s<DIS; s++) store[s]=MunFM.table[s].f;
double unarray[10];
unarray[0]=unA.eqummf(Mclassbh.MbhCurve, MunFM.table);
unA.Pathmmf=hart(unA.MMF,store,2);
unarray[1]=unB.eqummf(Mclassbh.MbhCurve, MunFM.table);
unB.Pathmmf=hart(unB.MMF,store,2);
unarray[2]=unC.eqummf(Mclassbh.MbhCurve, MunFM.table);
unC.Pathmmf=hart(unC.MMF,store,2);
unarray[3]=unD.eqummf(Mclassbh.MbhCurve, MunFM.table);
unD.Pathmmf=hart(unD.MMF,store,2);
unarray[4]=unE eqummf(Mclassbh.MbhCurve, MunFM.table);
unE.Pathmmf=hart(unE.MMF,store,2);
unarray[5]=unF.eqummf(Mclassbh.MbhCurve, MunFM.table);
unF Pathmmf=hart(unF.MMF,store,2);
unarray[6]=unG.eqummf(Mclassbh.MbhCurve, MunFM.table);
unG.Pathmmf=hart(unG.MMF,store,2);
unarray[7]=unH.eqummf(Mclassbh.MbhCurve, MunFM.table);
unH.Pathmmf=hart(unH.MMF,store,2);
unarray[8]=unI.eqummf(Mclassbh.MbhCurve, MunFM.table);
unI Pathmmf=hart(unI.MMF,store,2);
unarray[9]=unJ.eqummf(Mclassbh.MbhCurve, MunFM.table);
```

```
unJ.Pathmmf=hart(unJ.MMF,store,2);

//find average
            //by putting maximum flux value in each of the paths equations and then
//averaging the results
double AVE=(unarray[0]+unarray[1]+unarray[2]+unarray[3]+unarray[4]
                                                +unarray[5]+unarray[6]+unarray[7]+unarray[8]+unarray[9])/5,

//send average to compute mmf table
            MunFM.mmfset(AVE);

            unA.eqnewflux(MunFM.table,MunFM.tmpflux);//calculates the flux
            unB.eqnewflux(MunFM.table,MunFM.tmpflux),//in each path at the
            unC.eqnewflux(MunFM.table,MunFM.tmpflux);//final mmf, and sums
            unD.eqnewflux(MunFM.table,MunFM.tmpflux);//the values in tmpflux
            unE.eqnewflux(MunFM.table,MunFM.tmpflux);
            unF.eqnewflux(MunFM.table,MunFM.tmpflux);//calculates the flux
            unG eqnewflux(MunFM.table,MunFM.tmpflux);//in each path at the
            unH eqnewflux(MunFM.table,MunFM tmpflux);//final mmf, and sums
            unI.eqnewflux(MunFM.table,MunFM.tmpflux);//the values in tmpflux
            unJ.eqnewflux(MunFM.table,MunFM.tmpflux);
//final table, replace table.f by tmpflux readings.
MunFM.finalupdate();
//need an equation to model relationship
static double storem[DIS];
for (int kk=0; kk<DIS; kk++){
                                    storem[kk]=MunFM.table[kk].m;
                                    store[kk]=MunFM.table[kk].f;}
MunFM.equ=hart(storem,store,2);
//int poo=store[0];

}
}while (DESTROY!=0);



MessageBox(
"Airgap Reluctance is computed \n          Arrays Built"
,"PATHS",
MB_ICONINFORMATION);
graphyes=0;
 }
}
////////////////////////////////////////////////
//display results of graphs
void TDbWindow::CmGraph(){
if (graphyes==1){ //is data ready
MessageBox("FLUX / MMF DATA IS NOT COMPUTED!");
}
else{
            if (BhCHK==1){
            double maxX,maxY;

            if (MmidFM.table[DIS-1].f>MalFM.table[DIS-1].f)maxY=MmidFM.table[DIS-1].f;
            if (MunFM table[DIS-1].f>maxY) maxY=MunFM.table[DIS-1].f;

            if (MmidFM.table[DIS-1].m>MalFM table[DIS-1].m)maxX=MmidFM table[DIS-1].m,
            if (MunFM table[DIS-1].m>maxX) maxX=MunFM.table[DIS-1].m;

            double kflux=250/maxY;//auto-scaling flux
            double kmmf=250/maxX; //auto scaling mmf
            //***************
            //sets up origin of graph
            struct xy {double x;double y;}; //new structure for x and y
            xy origin; xy point; //origin and current point
            origin.x=50; origin y=300;

                        TGraphFrame * ThisChild,
                        ThisChild = new TGraphFrame(this, "GRAPH");
                        ThisChild->Create();
                        TClientDC dc(*ThisChild);//get this window
            //////////////

            //plot aligned graph//
                        dc.MoveTo(origin.x,origin.y);//move to start
                                    for (int i=0; i<DIS, i++)
                                    {
                        point.x=MalFM.table[i].m,
                        point.y=MalFM.table[i].f;
                        dc.LineTo(origin.x+(point.x*kmmf),
                                    origin.y-(point.y*kflux));
                        }
            //////////

            //plot mid graph//
            dc.MoveTo(origin.x,origin.y);//move to start
                        for (i=0; i<DIS; i++)
                        {
                        point.x=MmidFM.table[i] m;
                        point.y=MmidFM.table[i].f;
                        dc.LineTo(origin.x+(point x*kmmf),
                                    origin.y-(point y*kflux));
                        }
            //////////
             //plot un-aligned
```

```
                    dc MoveTo(ongin.x,ongin.y);//move to start
                    for (i=0; i<DIS; i++)
                    {
                    point.x=MunFM.table[i].m;
                    point.y=MunFM.table[i].f;
                    dc.LineTo(ongin.x+(point.x*kmmf),
                              ongin.y-(point.y*kflux));
                    }
          ////////////
          //axis
          dc.MoveTo(origin.x,ongin.y);//move to start
          dc.LineTo(origin.x+255,ongin.y);
          dc.MoveTo(ongin.x,origin y),//move to start
          dc.LineTo(origin.x,origin.y-255);
/*        char xaxis[15];
          gcvt(maxX,4,xaxis);
          dc.TextOut(ongin.x+255,origin y+5,xaxis,strlen(xaxis));
          char yaxis[15];
          gcvt(maxY,4,yaxis);
*///      dc.TextOut(origin.x-5,origin.y-255,yaxis,strlen(yaxis));
}
          else{MessageBox("GRAPH DRAWN FROM EQUATIONS");
                    ///get offset
          char maxmmf[]="0";
          struct TransferBuffer{
          char a[8],}cb,

          strcpy(cb.a,maxmmf);

          TBhOffDialog* the_padd=new TBhOffDialog(this, TResId(6));

the_padd->SetTransferBuffer(&cb);
if (the_padd->Execute()==IDOK){
strcpy(maxmmf,cb.a);}

double maxX=atof(maxmmf);
double maxY=(MalFM equ.a*pow(maxX,MalFM.equ.n))/
                                        (MalFM.equ.b+pow(maxX,MalFM.equ.n));


double kflux=250/maxY;//auto-scaling flux
double kmmf=250/maxX; //auto scaling mmf
          //*****************
           //sets up origin of graph
           struct xy {double x;double y;}; //new structure for x and y
           xy origin; xy point; //origin and current point
           origin.x=50; origin.y=300;

                    TGraphFrame * ThisChild;
          ThisChild = new TGraphFrame(this, "GRAPH");
          ThisChild->Create();
          TClientDC dc(*ThisChild);//get this window
//////////////
           //plot aligned graph//
          dc.MoveTo(origin.x,origin.y);//move to start

                    for (int i=0; i<DIS, i++)
                    {
                    point.x=(maxX/(DIS-1))*i;
                    point.y=(MalFM.equ.a*pow(point.x,MalFM.equ.n))/
                                        (MalFM.equ.b+pow(point.x,MalFM equ.n));
                    dc.LineTo(origin.x+(point.x*kmmf),
                              origin.y-(point.y*kflux));
                    }
          ////////////
           //plot mid graph//
          dc.MoveTo(origin.x,origin.y);//move to start

                    for (int k=0; k<DIS; k++)
                    {
                    point.x=(maxX/(DIS-1))*k;
                    point.y=(MmidFM.equ.a*pow(point.x,MmidFM.equ.n))/
                                        (MmidFM.equ.b+pow(point.x,MmidFM.equ.n));
                    dc.LineTo(origin.x+(point.x*kmmf),
                              origin.y-(point.y*kflux));
                    }
          ////////////
//////////
           //plot ungraph//
          dc.MoveTo(ongin.x,ongin.y);//move to start

                    for (int kk=0, kk<DIS; kk++)
                    {
                    point.x=(maxX/(DIS-1))*kk;
                    point.y=(MunFM.equ.a*pow(point.x,MunFM.equ.n))/
                                        (MunFM.equ.b+pow(point.x,MunFM equ.n));
                    dc.LineTo(origin.x+(point.x*kmmf),
                              origin y-(point.y*kflux));
                    }
          ////////////
 }

          }
}
///////**** end of main windows dialog function ***//
```

```
ThisChild->Create();
TClientDC dc(*ThisChild);//get this window
//////////////
 TColor pool=RGB(255,255,0);
 //dc.SetBkColor(pool);

dc.MoveTo(pic p1.x,pic.p1 y);//move to start
                dc.LineTo(pic p1.x,pic.p1.y);
                dc.LineTo(pic.pA1.x,pic.pA1.y);
                dc.LineTo(pic.pA.x,pic.pA.y);
                dc.LineTo(pic.pB.x,pic.pB.y);
                dc.LineTo(pic pB1.x,pic pB1.y);

                CoOr cen,c1;
                cen.x=pic.pB1.x+0.5*(pic.pC1.x-pic.pB1.x);
                cen.y=pic pC1.y+0.5*(pic.pB1.y-pic pC1.y);

 //             float LENGTH=(sqrt(
                //              ((pic.pC1.x-pic pB1.x)*(pic pC1.x-pic.pB1.x))+
                 //             ((pic.pB1 y-pic.pC1.y)*(pic pB1.y-pic.pC1.y))));
                float x1,y1,x2,y2,x3,y3;
                x1=cen.x-pic.k*pic.SR;
                y1=cen.y-pic.k*pic.SR;
                x2=cen.x+pic.k*pic.SR;
                y2=cen.y+pic.k*pic.SR;


dc.Arc(
                                                x2,y2,
                                                x1,y1,
                                                pic pC1.x,pic pC1.y,
                                                pic.pB1.x,1.01*pic pB1.y);


//dc Ellipse(  x1,y1,
                //                              x2,y2);
                dc.MoveTo(pic.pC1.x,pic.pC1.y);
                dc.LineTo(pic.pC1.x,pic.pC1.y),
                dc.LineTo(pic.pC.x,pic.pC.y);
                dc.LineTo(pic.pD.x,pic.pD.y);
                dc.LineTo(pic.pD1.x,pic.pD1.y);


                c1.x=cen.x=pic pD1.x+0.5*(pic.pE1.x-pic.pD1.x);
                c1.y=cen.y=pic.pE1.y+0.5*(pic.pD1.y-pic.pE1.y);

                x1=cen.x-pic.k*pic SR,
                y1=cen.y-pic k*pic SR,
                x2=cen.x+pic.k*pic.SR;
                y2=cen.y+pic k*pic.SR;


dc.Arc(  x1,y1,
                                                x2,y2,
                                                pic.pE1.x,pic pE1.y,
                                                pic.pD1.x,1.01*pic.pD1 y);

                //dc Ellipse(  x1,y1,
                //                              x2,y2);

                dc.MoveTo(pic pE1.x,pic pE1.y);
                dc.LineTo(pic.pE1.x,pic pE1.y);
                dc.LineTo(pic.pE.x,pic.pE.y);

                dc.LineTo(pic pF.x,pic.pF.y);
                dc.LineTo(pic.pF1.x,pic.pF1 y);

                cen.x=pic.pF1.x+0.5*(pic.pG1.x-pic.pF1.x);
                cen.y=pic.pF1.y+0.5*(pic.pG1.y-pic.pF1 y);

                x1=cen.x-pic k*pic.SR;
                y1=cen y-pic.k*pic.SR;
                x2=cen.x+pic.k*pic.SR;
                y2=cen y+pic.k*pic.SR;


dc.Arc(  x1,y1,
                                                x2,y2,
                                                pic.pG1.x,pic.pG1.y,
                                                pic.pF1.x,1.01*pic.pF1.y);

                // dc Ellipse(  x1,y1,
                //                              x2,y2);
                dc.MoveTo(pic.pG1.x,pic.pG1.y);
                dc.LineTo(pic pG1.x,pic.pG1.y);
                dc.LineTo(pic.pG.x,pic.pG.y);
                dc.LineTo(pic.pH.x,pic.pH.y);
                dc.LineTo(pic.pH1.x,pic.pH1.y);

                cen.x=pic pH1.x+0.5*(pic pI1.x-pic pH1.x);
                cen.y=pic.pH1.y+0.5*(pic.pI1.y-pic.pH1.y),

                x1=cen.x-pic.k*pic SR,
                y1=cen y-pic k*pic.SR;
```

```
                x2=cen.x+pic.k*pic.SR;
                y2=cen.y+pic.k*pic.SR;


dc.Arc( x1,y1,
                                                        x2,y2,
                                                        pic pI1.x,pic.pI1.y,
                                                        pic pH1.x,1.01*pic.pH1.y);

                dc.MoveTo(pic.pI1.x,pic pI1.y);
                dc.LineTo(pic pI1.x,pic.pI1.y);

                dc.LineTo(pic pI.x,pic.pI.y);
                dc.LineTo(pic.pJ.x,pic.pJ.y);
                dc.LineTo(pic.pJ1.x,pic.pJ1.y);
                dc.LineTo(pic.p2.x,pic.p2.y);
                dc.LineTo(pic p4.x,pic p4.y);
                dc.LineTo(pic.p3.x,pic.p3.y);
                dc.LineTo(pic.p1.x,pic.p1 y);

                //rotor
                dc.MoveTo(pic p5.x,pic.p5.y);//move to start
                dc.LineTo(pic.p5.x,pic.p5.y);

                dc.LineTo(pic rA1.x,pic.rA1.y);
                dc.LineTo(pic.rA.x,pic.rA.y);
                dc.LineTo(pic.rB.x,pic.rB.y);
                dc.LineTo(pic.rB1.x,pic.rB1.y);


                cen.x=pic.rB1.x+0.5*(pic.rC1.x-pic.rB1.x);
                cen.y=pic.rC1.y+0.5*(pic.rB1.y-pic.rC1.y);

                x1=cen.x-pic.k*pic.RR;
                y1=cen.y-pic.k*pic.RR;
                x2=cen.x+pic k*pic.RR;
                y2=cen.y+pic.k*pic.RR,


dc.Arc( x1,y1,
                                                        x2,y2,
                                                        0.99*pic.rB1.x,pic.rB1.y,
                                                        1 01*pic.rC1.x,0.99*pic.rC1.y); //


dc.MoveTo(pic.rC1.x,pic.rC1.y);
                dc.LineTo(pic.rC1.x,pic.rC1.y);
                dc.LineTo(pic.rC.x,pic.rC.y);
                dc LineTo(pic.rD.x,pic.rD y);
                dc.LineTo(pic.rD1.x,pic.rD1.y);

                cen.x=pic.rD1.x+0.5*(pic.rE1.x-pic.rD1.x);
                cen.y=pic.rE1.y+0.5*(pic.rD1.y-pic.rE1.y);

                x1=cen.x-pic.k*pic.RR;
                y1=cen.y-pic.k*pic RR;
                x2=cen.x+pic.k*pic.RR;
                y2=cen.y+pic.k*pic.RR;


dc.Arc( x1,y1,
                                                        x2,y2,
                                                        pic.rD1.x,0 99*pic.rD1.y,
                                                        1.01*pic.rE1.x,pic.rE1.y); //

                dc.MoveTo(pic.rE1.x,pic.rE1.y);
                dc.LineTo(pic.rE1.x,pic rE1 y);
                dc.LineTo(pic.rE.x,pic.rE.y);
                dc.LineTo(pic.rF.x,pic.rF.y);
                dc.LineTo(pic.rF1.x,pic.rF1 y);

                cen.x=pic.rF1.x+0.5*(pic.rG1.x-pic.rF1.x);
                cen.y=pic.rG1 y+0.5*(pic.rF1.y-pic.rG1.y);

                x1=cen.x-pic.k*pic.RR;
                y1=cen y-pic.k*pic RR;
                x2=cen.x+pic k*pic.RR;
                y2=cen.y+pic.k*pic RR;


dc.Arc( x1,y1,
                                                        x2,y2,
                                                        pic.rF1.x,pic.rF1.y,
                                                        1.01*pic.rG1.x,pic.rG1.y); //

                dc.MoveTo(pic.rG1.x,pic.rG1 y);
                dc.LineTo(pic.rG1.x,pic.rG1.y);
                dc.LineTo(pic.rG.x,pic.rG.y);

                dc.LineTo(pic.rH.x,pic.rH.y);
                dc.LineTo(pic.rH1.x,pic rH1.y);
                cen.x=pic.rH1.x+0.5*(pic.rI1.x-pic.rH1.x);
                cen y=pic.rH1 y+0.5*(pic.rI1.y-pic.rH1.y);

                x1=cen.x-pic.k*pic.RR,
```

```
                y1=cen.y-pic.k*pic RR;
                x2=cen.x+pic.k*pic RR;
                y2=cen.y+pic.k*pic RR;


dc.Arc( x1,y1,
                                                x2.y2.
                                                pic.rH1.x,pic.rH1.y,
                                                pic.rI1.x*1.01,pic.rI1.y), //

                dc MoveTo(pic.rI1.x,pic.rI1.y);

                dc LineTo(pic rI1.x,pic.rI1.y);
                dc LineTo(pic rI.x,pic.rI.y);
                dc LineTo(pic.rJ.x,pic.rJ y);
                dc LineTo(pic.rJ1.x,pic.rJ1.y);

                cen.x=pic.rJ1.x+0 5*(pic.rK1.x-pic.rJ1.x);
                cen.y=pic rJ1.y+0 5*(pic.rK1.y-pic.rJ1.y);

                x1=cen.x-pic.k*pic.RR,
                y1=cen.y-pic.k*pic.RR,
                x2=cen.x+pic.k*pic RR,
                y2=cen.y+pic.k*pic RR;


dc.Arc( x1,y1,
                                                x2,y2,
                                                pic.rJ1.x,pic.rJ1 y,
                                                pic.rK1.x*1.01,pic.rK1.y); //

                dc.MoveTo(pic.rK1.x,pic.rK1.y);
                dc.LineTo(pic.rK1.x,pic.rK1 y);
                dc.LineTo(pic.rK.x,pic.rK.y);
                dc LineTo(pic rL.x,pic.rL y);
                dc.LineTo(pic.rL1.x,pic rL1.y);

                cen.x=pic.rL1.x+0.5*(pic.rM1.x-pic.rL1.x);
                cen.y=pic.rL1.y+0.5*(pic.rM1 y-pic rL1.y);

                x1=cen.x-pic.k*pic.RR;
                y1=cen.y-pic.k*pic.RR,
                x2=cen.x+pic.k*pic.RR;
                y2=cen.y+pic.k*pic.RR;


dc.Arc( x1,y1,
                                                x2,y2,
                                                pic rL1.x,pic.rL1.y,
                                                pic.rM1.x*1.01,pic.rM1.y); //

                dc MoveTo(pic.rM1.x,pic.rM1 y);

                dc LineTo(pic.rM1.x,pic.rM1 y);
                dc.LineTo(pic.rM.x,pic.rM.y);
                dc LineTo(pic.rN.x,pic.rN.y);
                dc LineTo(pic rN1.x,pic.rN1.y);

                dc LineTo(pic.p6.x,pic.p6.y);
                dc.LineTo(pic p5.x,pic.p5.y);

//fill with colour

        TColor color=RGB(0,0,0);
                TColor poo=RGB(0,0,255);

                TPoint p(50,50);
                TPoint c(200,250);
                TPoint a(200,150);

        dc.SelectStockObject(GRAY_BRUSH);


        //dc.LineTo(p);
//      TBrush* pPen=new TBrush;
//      pPen->CreateBrush(PS_SOLID,1,RGB(0,0,255));
//      dc.SelectObject(Ppen);
         dc SetROP2(R2_NOTCOPYPEN);
                dc.ExtFloodFill(p.color,FLOODFILLBORDER);
//              dc.ExtFloodFill(c,color,FLOODFILLBORDER);
                dc.SetROP2(R2_COPYPEN);
                TBrush brushy(poo);
         dc SelectObject(brushy);
                dc.ExtFloodFill(a.color,FLOODFILLBORDER);
                TBrush brushi(poo1);
         dc.SelectObject(brushi);
                dc ExtFloodFill(c,color,FLOODFILLBORDER);
          //            dc SetPixel(c,poo);
        }
//////////

//display results of graphs
void TDbWindow::CmGraph(){
if (graphyes==1){ //is data ready
MessageBox("FLUX / MMF DATA IS NOT COMPUTED!");
```

```
}
else{
        if (BhCHK==1){
        double maxX,maxY;

        if (MmidFM.table[DIS-1].f>MalFM.table[DIS-1].f)maxY=MmidFM.table[DIS-1].f;
        if (MunFM.table[DIS-1].f>maxY) maxY=MunFM.table[DIS-1].f;

        if (MmidFM.table[DIS-1].m>MalFM.table[DIS-1].m)maxX=MmidFM.table[DIS-1].m;
        if (MunFM.table[DIS-1].m>maxX) maxX=MunFM.table[DIS-1].m;

        double kflux=250/maxY;//auto-scaling flux
        double kmmf=250/maxX; //auto scaling mmf
        //****************
        //sets up origin of graph
        struct xy {double x;double y;}; //new structure for x and y
        xy origin; xy point; //origin and current point
        origin.x=50; origin.y=300;

                TGraphFrame * ThisChild,
                ThisChild = new TGraphFrame(this, "GRAPH");
                ThisChild->Create();
                TClientDC dc(*ThisChild);//get this window
////////////////

//plot aligned graph//
                dc.MoveTo(origin.x,origin.y);//move to start
                        for (int i=0; i<DIS; i++)
                        {
                        point.x=MalFM.table[i].m;
                        point.y=MalFM.table[i].f;
                        dc.LineTo(origin.x+(point.x*kmmf),
                                origin.y-(point.y*kflux));
                        }
                ////////////

                //plot mid graph//
                dc.MoveTo(origin.x,origin.y);//move to start
                        for (i=0; i<DIS; i++)
                        {
                        point.x=MmidFM.table[i].m;
                        point.y=MmidFM.table[i].f;
                        dc.LineTo(origin.x+(point.x*kmmf),
                                origin.y-(point.y*kflux));
                        }
                ////////////
                 //plot un-aligned
                        dc.MoveTo(origin.x,origin.y);//move to start
                        for (i=0; i<DIS; i++)
                                {
                                point.x=MunFM.table[i].m;
                                point.y=MunFM.table[i].f;
                                dc.LineTo(origin.x+(point.x*kmmf),
                                        origin.y-(point.y*kflux));
                                }
                        ////////////
                        //axis
                        dc.MoveTo(origin.x,origin.y);//move to start
                        dc.LineTo(origin.x+255,origin.y);
                        dc.MoveTo(origin.x,origin.y);//move to start
                        dc.LineTo(origin.x,origin.y-255);
         }
         else{
                        ///get offset
                char maxmmf[]="0";
                struct TransferBuffer{
                char a[8];}cb;

                strcpy(cb.a,maxmmf);

                TGraphMmfDialog* the_padd=new TGraphMmfDialog(this, TResId(6));

the_padd->SetTransferBuffer(&cb);
if (the_padd->Execute()==IDOK){
strcpy(maxmmf,cb.a);}

double maxX=atof(maxmmf);
double maxY=(MalFM.equ.a*pow(maxX,MalFM.equ.n))/
                (MalFM.equ.b+pow(maxX,MalFM.equ.n));

double kflux=250/maxY;//auto-scaling flux
double kmmf=250/maxX; //auto scaling mmf
        //****************
         //sets up origin of graph
         struct xy {double x,double y;}; //new structure for x and y
         xy origin; xy point; //origin and current point
         origin.x=50; origin.y=300;

                TGraphFrame * ThisChild,
                ThisChild = new TGraphFrame(this, "GRAPH");
                ThisChild->Create();
                TClientDC dc(*ThisChild);//get this window
         ////////////////
                 //plot aligned graph//
                dc.MoveTo(origin.x,origin.y);//move to start
```

```
                for (int i=0; i<DIS; i++)
                {
                point.x=(maxX/(DIS-1))*i;
                point.y=(MalFM equ.a*pow(point.x,MalFM.equ.n))/
                                        (MalFM.equ.b+pow(point.x,MalFM equ.n));
                dc.LineTo(origin.x+(point.x*kmmf),
                                origin.y-(point.y*kflux));
                }
//////////
 //plot mid graph//
dc.MoveTo(origin.x,origin.y);//move to start

                for (int k=0; k<DIS; k++)
                {
                point.x=(maxX/(DIS-1))*k;
                point y=(MmidFM equ.a*pow(point.x,MmidFM equ.n))/
                                        (MmidFM.equ.b+pow(point.x,MmidFM.equ.n));
                dc LineTo(origin.x+(point.x*kmmf),
                                origin.y-(point.y*kflux));
                }
                //////////
                                //////////
                 //plot ungraph//
                dc.MoveTo(origin.x,origin y);//move to start

                                for (int kk=0; kk<DIS; kk++)
                                {
                                point.x=(maxX/(DIS-1))*kk;

                                if (MunFM.equ.k==1){point.y=point.x*MunFM.equ.k;}
                                else{
                                point.y=(MunFM.equ.a*pow(point.x,MunFM.equ.n))/
                                                        (MunFM.equ.b+pow(point.x,MunFM equ.n));}

                                dc.LineTo(origin.x+(point.x*kmmf),
                                                origin.y-(point.y*kflux));
                                }

                                //////////
                //axis
                dc.MoveTo(origin.x,origin.y);//move to start
                dc.LineTo(origin.x+255,origin y);
                dc.MoveTo(origin.x,origin.y);//move to start
                dc.LineTo(origin.x,origin.y-255);
                //////////
}

        }
}
////////////////////////////////////////////
//static torque
void TDbWindow::CmStat(){
int phase;
char MXcurr[10]="7";
char INC[10]="0 5";
char NuTP[10]="96";


struct TransferBuffer{
char dCURR[10],dTURN[10],dINC[10];}tb;

strcpy(tb.dCURR,MXcurr);strcpy(tb.dINC,INC);strcpy(tb.dTURN,NuTP);
TSDialog* the_padd=new TSDialog(this, TResId(7));

 the_padd->SetTransferBuffer(&tb);
 if (the_padd->Execute()==IDOK){
 phase=the_padd->RadioState;
 strcpy(MXcurr,tb.dCURR);strcpy(INC,tb.dINC);strcpy(NuTP,tb.dTURN);
                }
//calculate magnet/back iron relationship
FluxMmf magbi[DIS];
double flux[DIS];
double mmf[DIS];
MbimagMMF=BImag(Mclassbh.MbhCurve,Mclassmag MbhCurve,magbi,MclassGEO,Mclassmag);
for (int r=0,j<DIS; i++){flux[i]=magbi[i] f,mmf[1]=magbi[1].m;}
McurveBIMAG=hart(mmf,flux,0);
Curve magnet;
magnet.a=Mclassmag MbhCurve[0].n;
magnet.b=Mclassmag.MbhCurve[1].n;
magnet.n=Mclassmag MbhCurve[2].n;
/////
double MXCURR=atof(MXcurr);
int NUTP=atof(NuTP);
double cINC=atof(INC);
int points=MXCURR/cINC;
float curr=0;

///do next curve depend on phase single
if (phase==1){
//build aligned curve

//double OFFSET,
FluxLink ALIGNED;//make make this global
lnklist al:    //or this
LINKlist ALTORm;
```

```
//float last=cINC*-1;
for (int k=points; k>-1; k--)
{
curr=k*cINC;
ALIGNED=get(McurveBIMAG, MalFM.equ,MmidFM.equ,MunFM equ,MbimagMMF,magnet,
MclassGEO.nsDIM[15].n,MclassGEO.nsDIM[16].n,Mclassmag offset.n,curr,NUTP,0);
al.additem(ALIGNED),

double ALf,UNf,
if (k==points){ALf=ALIGNED.al,UNf=ALIGNED.un,}
else {
double torqueSa=coENG(cINC,NUTP,ALf,UNf,ALIGNED.al,ALIGNED.un,0,0);
ALTORm.toradd(torqueSa);
ALf=ALIGNED.al;UNf=ALIGNED.un;}
//if (curr==0) OFFSET=ALIGNED.al-ALIGNED.un;
}
//curr=0;


FluxLink MID;//make make this global
linklist mid;    //or this
LINKlist MIDTORa;
//find mid curve
for ( k=points; k>-1; k--)
                {
curr=k*(cINC);
MID=get(McurveBIMAG, MalFM.equ,MmidFM.equ,MunFM.equ,MbimagMMF,magnet,
MclassGEO.nsDIM[15].n,MclassGEO.nsDIM[16].n,Mclassmag.offset.n,curr,NUTP,1);
mid.additem(MID),


double Maf,Mbf,
if (k==points){Maf=MID.al,Mbf=MID.un;}
else {
double torqueSa=coENG(cINC,NUTP,Maf,Mbf,MID al,MID.un,0,0);
MIDTORa.toradd(torqueSa);
Maf=MID.al,Mbf=MID.un;}
}

 LINKlist TORQUEsp;
 TORQUEsp.finaltorque(ALTORm,MIDTORa);

 ///////
                //            TGraphFrame * ThisChild;
                //            ThisChild = new TGraphFrame(this, "GRAPH");
                //            ThisChild->Create();
                //            TClientDC dc(*ThisChild);//get this window
                //////////////
                             //plot torque graph//
                             //dc.MoveTo(10,250);//move to start

                             for (int position=0, position<points; position++){
                             double plot=TORQUEsp.PLOT(position);
                             //dc.LineTo(10+position*0.5*50,300-plot*100);
                             double value=plot;
                             }

}
//two phase
else{

//build aligned curve

double OFFSET;
FluxLink ALIGNED2;//make make this global
linklist al2;    //or this
LINKlist ALTORu2;

//float last=cINC*-1;
for (int k=points; k>-1; k--)
{
curr=k*cINC;
ALIGNED2=get(McurveBIMAG, MalFM.equ,MmidFM.equ,MunFM.equ,MbimagMMF,magnet,
MclassGEO.nsDIM[15].n,MclassGEO.nsDIM[16].n,Mclassmag.offset.n,curr,NUTP,2);
al2.additem(ALIGNED2);

double ALf,UNf,
if (k==points){ALf=ALIGNED2.al,UNf=ALIGNED2.un;}
else {
double torqueSa=coENG(cINC,NUTP,ALf,UNf,ALIGNED2 al,ALIGNED2.un,0.0);
ALTORu2 toradd(torqueSa);
ALf=ALIGNED2.al,UNf=ALIGNED2.un;}
if (curr==0) OFFSET=ALIGNED2.al-ALIGNED2 un;
}


FluxLink UNALIGNED;//make make this global
linklist un;    //or this
LINKlist UNTORa;
```

```
//float last=cINC*-1;
for ( k=points; k>-1; k--)
{
curr=-k*cINC;
UNALIGNED=get(McurveBIMAG, MalFM.equ,MmidFM.equ,MunFM.equ,MbimagMMF,magnet,
MclassGEO.nsDIM[15].n,MclassGEO.nsDIM[16].n,Mclassmag.offset.n,curr,NUTP,2);
un.additem(UNALIGNED);

double ALf,UNf;
if (k==points){ALf=UNALIGNED.al,UNf=UNALIGNED.un;}
else {
double torqueSa=coENG(cINC,NUTP,ALf,UNf,UNALIGNED.al,UNALIGNED.un,1,OFFSET);
UNTORa.toradd(torqueSa);
ALf=UNALIGNED.al;UNf=UNALIGNED.un;}
}
 //caclulate co-energy use trapezium
 LINKlist TORQUE2p;
 TORQUE2p.finaltorque(ALTORu2,UNTORa);

///////
                //              TGraphFrame * ThisChild;
                //              ThisChild = new TGraphFrame(this, "GRAPH");
                //              ThisChild->Create();
                //              TClientDC dc(*ThisChild);//get this window
                ////////////////
                              //plot torque graph//
                              //dc.MoveTo(10,250);//move to start

                              for (int position=0; position<points; position++){
                              double plot=TORQUE2p.PLOT(position);
                              //dc.LineTo(10+position*0.5*50,300-plot*100);
                              double value=plot;
                              }
 }
/////*detent*
FluxLink DETENT;
DETENT=getDENT(McurveBIMAG, MalFM.equ,MmidFM.equ,MunFM.equ,MbimagMMF);
double Dflux=(DETENT.al-DETENT.un)-(DETENT.mid1-DETENT.mid2);
double Dtor=0.5*Dflux*NUTP*200/M_PI;
int poo=1;
}
///////**** end of main windows dialog function ***//


//*** MAIN WINDOW ***
void
TDbApp::InitMainWindow()
                {
                              MainWindow = new TDbWindow(0,"HYSTEP - Permeance");
                }

// ** RUN APPLICATION ** //
int
OwlMain(int /*argc*/, char* /*argv*/ [])
{
 TDbApp app;
 return app.Run();
}
```

```
#include "com.h"
//function
Curve hart(double xx[DIS],double yy[DIS],int path)
{
const int no=10;
double x[no],x[0]=xx[5],x[1]=xx[10],x[2]=xx[15],x[3]=xx[20],x[4]=xx[23];
x[5]=xx[25];x[6]=xx[29],x[7]=xx[33];x[8]=xx[36],x[9]=xx[39];
double y[no];y[0]=yy[5];y[1]=yy[10];y[2]=yy[15];y[3]=yy[20];y[4]=yy[23];
y[5]=yy[25],y[6]=yy[29];y[7]=yy[33];y[8]=yy[36];y[9]=yy[39];
int COUNT;
 //rows columns

// intial guess may do work on intial guess for frolich function!!
Curve RESULT;
double iALP;//yy[DIS-1]*1.1;//y[no-1]*1e4;  cout<<endl<<"Alpha"<<iALP;
double iBET;//xx[DIS-1]/6, //cout<<endl<<"beta"<<iBET;
double iN=1;
double SumSQ=0;
int ErroR=0;

double NEWALP=0;
double NEWBET=0;
double NEWN=0;

static double Sum[4];
static double T1[no][4];
static double T2[3][4];

static double COEFF[3][2][4];//alpha beta
COEFF[0][0][0]=1 05*y[9],COEFF[0][1][0]=1.5*x[9];
COEFF[0][0][1]=2*y[9],COEFF[0][1][1]=3 8*x[9];
COEFF[0][0][2]=1.5*y[9];COEFF[0][1][2]=3.3*x[9];
COEFF[0][0][3]=1.25*y[9];COEFF[0][1][3]=2.5*x[9];

COEFF[1][0][0]=1.05*y[9];COEFF[1][1][0]=1.5*x[9];
COEFF[1][0][1]=2.3*y[9],COEFF[1][1][1]=4*x[9];
COEFF[1][0][2]=1 6*y[9],COEFF[1][1][2]=3.5*x[9];
COEFF[1][0][3]=1.3*y[9];COEFF[1][1][3]=2.5*x[9];

COEFF[2][0][0]=1.05*y[9],COEFF[2][1][0]=1.5*x[9];
COEFF[2][0][1]=1.25*y[9];COEFF[2][1][1]=3.7*x[9];
COEFF[2][0][2]=1.05*y[9],COEFF[2][1][2]=3.1*x[9];
COEFF[2][0][3]=1*y[9],COEFF[2][1][3]=2.5*x[9];

COEFF[3][0][0]=1.05*y[9],COEFF[2][1][0]=0.5*x[9];
COEFF[3][0][1]=1.25*y[9];COEFF[2][1][1]=0 5*x[9],
COEFF[3][0][2]=1.05*y[9];COEFF[2][1][2]=0.25*x[9];
COEFF[3][0][3]=1*y[9],COEFF[2][1][3]=0.75*x[9],

Curve keeper[4];
int move=0;
do {
iALP=COEFF[path][0][move]; iBET=COEFF[path][1][move];iN=1;COUNT=0;
/////
do {
T2[0][0]=T2[0][1]=T2[0][2]=T2[0][3]=T2[1][0]=
T2[1][1]=T2[1][2]=T2[1][3]=T2[2][0]=T2[2][1]=
T2[2][2]=T2[2][3]=0;

for (int i=0; i<no; i++){

                                                    T1[i][0]=pow(x[i],iN)/(iBET+pow(x[i],iN));
                                                    T1[i][1]=(-iALP*pow(x[i],iN))/pow((iBET+pow(x[i],iN)),2);
                                                    T1[i][2]=
                                                    (
                                                    (iALP*pow(x[i],iN)*log(x[i]))/
                                                    (iBET+pow(x[i],iN)))
                                                    -
                                                    (
                                                    (iALP*pow((pow(x[i],iN)),2)*log(x[i]))/
                                                    (pow((iBET+pow(x[i],iN)),2))
                                                    );
                                                    T1[i][3]=y[i]-(
                                                    (iALP*pow(x[i],iN))
                                                    /(iBET+pow(x[i],iN)));
                                                    T2[0][0]+=pow(T1[i][0],2);
                                                    T2[0][1]+=T1[i][0]*T1[i][1];
                                                    T2[0][2]+=T1[i][0]*T1[i][2];
                                                    T2[0][3]+=T1[i][0]*T1[i][3];

                                                    T2[1][0]+=T1[i][0]*T1[i][1];
                                                    T2[1][1]+=pow(T1[i][1],2);
                                                    T2[1][2]+=T1[i][1]*T1[i][2];
                                                    T2[1][3]+=T1[i][1]*T1[i][3];

                                                    T2[2][0]+=T1[i][0]*T1[i][2];
                                                    T2[2][1]+=T1[i][2]*T1[i][1];
                                                    T2[2][2]+=T1[i][2]*T1[i][2];
                                                    T2[2][3]+=T1[i][2]*T1[i][3];
                                                    }

// solve systems of equations for D.
double EQ1[4], EQ2[4];
for ( i=0; i<4; i++) {
                                                    EQ1[i]=(T2[1][0]/T2[0][0])*T2[0][i];
                                                    EQ2[i]=T2[1][i]-EQ1[i];)
```

```
double EQ3[4];
for (i=0; i<4; i++) {
                                        EQ1[i]=(T2[2][0]/T2[0][0])*T2[0][i];
                                        EQ3[i]=T2[2][i]-EQ1[i];}

double EQ4[4];
for ( i=0; i<4, i++) {
                                        EQ1[i]=(EQ3[1]/EQ2[1])*EQ2[i];
                                        EQ4[i]=EQ3[i]-EQ1[i];}
double D[3];
D[2]=EQ4[3]/EQ4[2];
D[1]=(EQ2[3]-(D[2]*EQ2[2]))/EQ2[1],
D[0]=(T2[0][3]-(D[2]*T2[0][2])-(D[1]*T2[0][1]))/T2[0][0];
//
//minimum value of Q
double alp1, bet1, n1, alp05, bet05, n05;
alp1=iALP+D[0]; bet1=iBET+D[1]; n1=iN+D[2];
alp05=iALP+0.5*D[0]; bet05=iBET+0.5*D[1]; n05=iN+0.5*D[2];
double Q0=0;double Q05=0; double Q1=0;
for (i=0; i<no;i++) {
                Q0+=T1[i][3]*T1[i][3];
                Q1+=pow((y[i]-((alp1*pow(x[i],n1))/(bet1+pow(x[i],n1)))),2);
                Q05+=pow(y[i]-((alp05*pow(x[i],n05))/(bet05+pow(x[i],n05))),2);
                }


if ((Q1-2*Q05+Q0)!=0){
double Vmin=0.5+(0.25*(Q0-Q1)/(Q1-2*Q05+Q0));

NEWALP=iALP+Vmin*D[0];
NEWBET=iBET+Vmin*D[1];
NEWN=iN+Vmin*D[2];
}
else {COUNT=55,}
//cout<<endl<<NEWALP<<" (C) "<<NEWBET<<" A B N "<<NEWN;
ErroR=0;
if (NEWALP<0 || NEWBET<0 || NEWN<0)  {ErroR=1,COUNT=55,}


//find convergance percentage

double Ealp=((NEWALP-iALP)/NEWALP)*100;  Ealp=fabs(Ealp);
double Ebet=((NEWBET-iBET)/NEWBET)*100;  Ebet=fabs(Ebet);
double En=((NEWN-iN)/NEWN)*100;  En=fabs(En);
//cout<<endl<<Ealp<<" converge (c) "<<Ebet<<"  "<<En;

// converge check
if (Ealp<1e-5 && Ebet<1e-5 && En<1e-5) COUNT=55;
iALP=fabs(NEWALP); iBET=fabs(NEWBET); iN=fabs(NEWN),COUNT++;
RESULT.a=NEWALP;RESULT.b=NEWBET,RESULT.n=NEWN,
}while (COUNT<40);
//sum of squares of residuals
if (ErroR==1){SumSQ=1e10;}
else{
SumSQ=0;
for (int i=0; i<no, i++) {

                SumSQ+=pow(
                y[i]-(
                (RESULT.a*pow(x[i],RESULT.n))
                /(RESULT.b+pow(x[i],RESULT.n))),2); }
                }
Sum[move]=SumSQ;
keeper[move] a=RESULT.a.keeper[move].b=RESULT.b.keeper[move].n=RESULT.n;

move++;
}while(move<4);

double low;
low=Sum[4];
for (int z=0; z<3,z++){if (low>Sum[z]) {
                                        low=Sum[z],RESULT.a=keeper[z].a,RESULT.b=keeper[z].b;
                                        RESULT.n=keeper[z].n;       }}

/// section to check linear unaligned;
RESULT.k=0;
if (path==2){
double LinSumSQ=0;
for (int i=0; i<no; i++)          {
 LinSumSQ +=pow(
                y[i]-(x[i]*(y[9]/x[9]))
                ,2);
 if (LinSumSQ<low) {RESULT.e=(y[5]/x[5]); RESULT.k=1;}
                                                                }
return RESULT;
}
////////////////////////////////////////
```

```
#include "static.h"



//////////////////////
//** TStaticDialog ** //

DEFINE_RESPONSE_TABLE1(TStaticDialog, TDialog)
                // EV_COMMAND(954, Radio1),
                // EV_COMMAND(955, Radio2),
END_RESPONSE_TABLE,

//****** intial set up for static torque Dialog ******
TStaticDialog::TStaticDialog(TWindow* parent, const char* name)
:TDialog(parent, name),
                TWindow(parent)
{


//Rbutton=0; //radio button selected for no phase;
}
//*functions contained within static torque dialog
/*
void
TStaticDialog::Radio1()
{
Rbutton=1;
}
void
TStaticDialog :Radio2()
{
Rbutton=2;
}
 */
//////////////////////
```

```
////// this header stores the path calculation file ///
#include "com.h"
#include "classla.h"

#include "genfun.h"
///////
const float MU0=1.2566e-6;   //permability of free space
//functions for pathla class
///////

void pathLA::Apath1(pathdim geo)
{
//for (int inc=0; inc<9; inc++) AL[inc].in=0,

AL[0].A=geo.STpwid*(geo.DEP*1e-6/5);
AL[0].L=geo.STphgt*1e-3;

double lenA, lenB;
lenA=pythag(geo.sbSTp,geo.tSTet);
CoOr COtmp;
COtmp.x=geo.sbSTp.x-(geo.STpwid/5); COtmp y=geo.sbSTp y;
lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(lenA+lenB)*0.5e-3;
lenA=pythag(geo.tSTet,geo.tSTea);
AL[1].A=(lenA+(geo.STpwid/5))*0.5*geo.DEP*1e-6;

AL[2].L=((0.785398+geo.BETA)*(geo.STrad+2.5*geo.STwid/41)+
          (geo.STendT-geo.SThgt-((13*lenA/74)*sin(geo BETA))))*0.5e-3;
AL[2].A=((53.5*lenA/73.5)+(35.5*geo.STwid/41))*0.5*geo.DEP*1e-6;

double LENGTH, AREA;

LENGTH=geo.SThgt*1.027e-3;
AREA=geo.STwid*geo.DEP*0.853e-6;

//axis pathA;
AL[3].L=LENGTH;
AL[3].A=AREA;

AL[4].L=1.027*geo.RThgt*1e-3;
AL[4].A=0.965*geo.RTwid*geo.DEP*1e-6;

AL[5].L=geo.RTrad*1e-3;
AL[5].A=(0.93*geo.RTwid+0.434782*geo RTrad)*geo.DEP*1e-6;

AL[6].L=geo.RTweb*1e-3,
AL[6].A=AL[5].A;
}

//////////////////////

void pathLA::Apath2(pathdim geo)
{

AL[0].A=geo STpwid*(geo.DEP*1e-6/5);
AL[0] L=geo.STphgt*1e-3,

double lenA, lenB;

CoOr COtmpA, COtmpB,
COtmpA.x=geo.sbSTp.x-(geo.STpwid/5); COtmpA y=geo.sbSTp y;
COtmpB.x=geo.mbSTp.x+(geo.STpwid/10); COtmpB.y=geo.mbSTp.y;
lenA=pythag(COtmpA,geo.tSTea),
lenB=pythag(COtmpB,geo.tSTfa);

AL[1].L=(lenA+lenB)*0.5e-3;
lenA=pythag(geo.tSTfa,geo.tSTea);
AL[1].A=(lenA+(geo.STpwid/5))*0.5*geo.DEP*1e-6;

AL[2].L=((0 785398+geo.BETA)*(geo STrad+2*geo.STwid/42)+
          (geo.STrad))*0.5e-3;
AL[2].A=((80*geo.STwid/42)+(9*geo.STrad/21))*0.5*geo.DEP*1e-6;

double LENGTH, AREA;

LENGTH=geo.SThgt*1.027e-3;
AREA=geo STwid*geo.DEP*0.853e-6;

//axis pathA;
AL[3].L=LENGTH;
AL[3].A=AREA;

AL[4] L=1.027*geo.RThgt*1e-3;
AL[4].A=0.965*geo.RTwid*geo.DEP*1e-6;

AL[5].L=geo.RTrad*1e-3;
AL[5].A=(0.93*geo.RTwid+0.434782*geo.RTrad)*geo.DEP*1e-6;

AL[6].L=geo.RTweb*1e-3;
AL[6].A=AL[5].A;
}

//////////////////////
```

```
void pathLA::Apath3(pathdim geo)
{
AL[0].A=geo.STpwid*(geo.DEP*1e-6/5);
AL[0].L=geo.STphgt*1e-3,

double lenA, lenB;

CoOr COtmpA, COtmpB;
COtmpA.x=geo.sbSTp x-(geo.STpwid/5); COtmpA.y=geo.sbSTp.y;
COtmpB.x=geo.mbSTp.x+(geo STpwid/10); COtmpB y=geo.mbSTp.y;
lenA=pythag(COtmpA,geo.tSTea);
lenB=pythag(COtmpB,geo.tSTfa);

AL[1].L=(lenA+lenB)*0.5e-3;
lenA=pythag(geo.tSTfa,geo.tSTea);
AL[1].A=(lenA+(geo STpwid/5))*0.5*geo.DEP*1e-6;

AL[2].L=(M_PI*((geo STrad/3)+(geo.STwid/41.5))+geo.STrad)*0.333333e-3;

AL[2].A=((2*geo.STwid)+(6.5*geo.STrad/21))*0.5*geo.DEP*1e-6;
double LENGTH, AREA;

LENGTH=geo.SThgt*1.027e-3,
AREA=geo.STwid*geo.DEP*0.853e-6;

//axis pathA,
AL[3].L=LENGTH;
AL[3].A=AREA;

AL[4].L=1.027*geo RThgt*1e-3;
AL[4].A=0.965*geo RTwid*geo.DEP*1e-6;

AL[5].L=geo.RTrad*1e-3;
AL[5].A=(0.93*geo.RTwid+0.434782*geo.RTrad)*geo.DEP*1e-6;

AL[6].L=geo RTweb*1e-3;
AL[6].A=AL[5].A;
}


//////////////////////

void pathLA::Atooth1(pathdim geo)
{
//set up ellipse function
axis pathA;

//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth A

        pathA.a=geo.AGlen+geo.RThgt*0 15;
        pathA.b=3.5*(geo.STwid-geo.RTwid)/22;
        float ellen=ellipse(pathA);
        Tal[0].L=((geo.AGlen)+(ellen))*0.5e-3;
        Tal[0].A=(pathA.b+0.1587*geo.RTwid+0.2*geo RThgt)*geo.alDEP*0.5e-6;

        Tal[1].L=geo.AGlen*1e-3;
        Tal[1].A=geo.RTwid*0.86*geo.alDEP*1e-6;

                pathA.a=geo.AGlen+0.2195*geo.RThgt;
        pathA.b=0.393*(geo.STwid-geo.RTwid)/2;
                ellen=ellipse(pathA);
        Tal[2].L=(geo.AGlen+ellen)*0.5e-3;
        Tal[2].A=(pathA.b+0.118*geo.RTwid+0.2195*geo.RThgt)*geo.alDEP*0.5e-6;

        Tal[3].L=0;

        Tal[3].A=0;

                Tal[4].L=0;
        Tal[4].A=0;

        Tal[5] L=0;
        Tal[5].A=0;

                empty=3;

}

void pathLA::Atooth2(pathdim geo)
{
//set up ellipse function
axis pathA,pathB;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth B and D

        pathA.a=geo AGlen+geo.RThgt*0.238,
        pathA.b=0.2143*(geo.STwid-geo.RTwid);
        float ellenA=ellipse(pathA);
```

```
Tal[0].L=((ellenA)+(geo.AGlen))*0.5e-3,
Tal[0].A=(pathA.b+0.131*geo.RTwid+0.244*geo.RThgt)*geo.alDEP*0.5e-6;


Tal[1] L=geo.AGlen*1e-3;
Tal[1].A=geo.RTwid*0.8455*geo.alDEP*1e-6;

Tal[2].L=geo.AGlen*1.041e-3;
Tal[2].A=(geo.RTwid*0.213)*geo.alDEP*0.5e-6;

pathA.a=geo.AGlen+geo SThgt*0.357;
pathA.b=0.29*geo.RTrad;
ellenA=ellipse(pathA);
pathB.a=geo RTrad*0.5;
pathB.b=0.5*(geo.STwid-geo RTwid)+0.255*geo.RTrad;
float ellenB=ellipse(pathB);

Tal[3].L=(ellenA+ellenB+1.08*geo.AGlen+geo.RThgt
                        -(0.2222*geo.RTrad*M_PI*sin(0.589)))*0.5e-3;
Tal[3].A=(0.435*geo.SThgt+0.444*(geo.STwid-geo.RTwid)
                        +(geo.RThgt+0.2222*M_PI*geo.RTrad))*0.5e-6,

        //Tal[3].L=0;
        //Tal[3].A=0;


        Tal[4] L=0;
        Tal[4].A=0;


        Tal[5] L=0;
        Tal[5].A=0;

                empty=2;

}

void pathLA::Atooth3(pathdim geo)
{
//set up ellipse function
axis pathA,pathB;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth  C

        pathA.a=geo.AGlen+geo.RThgt*0.262;
        pathA.b=0.158*(geo.STwid-geo.RTwid);
        float ellenA=ellipse(pathA);


Tal[0].L=((ellenA)+(geo.AGlen))*0.5e-3;
Tal[0].A=(pathA.b+0.115*geo.RTwid+0.2857*geo.RThgt)*geo.alDEP*0.5e-6;


Tal[1].L=geo.AGlen*1e-3;
Tal[1].A=geo.RTwid*0 8443*geo alDEP*1e-6;

Tal[2].L=geo.AGlen*1.041e-3,
Tal[2].A=(geo.RTwid*0.10656+geo.STwid*0.08475)*geo.alDEP*0.5e-6;

pathA a=geo.AGlen+geo SThgt*0.42;
pathA.b=0.2766*geo.RTrad;
ellenA=ellipse(pathA);
pathB a=geo.RTrad*0.5;
pathB.b=0.5*(geo.STwid-geo.RTwid)+0.29*geo.RTrad;
float ellenB=ellipse(pathB);

Tal[3].L=(ellenA+ellenB+1.08*geo.AGlen+geo.RThgt
                        -(0.2222*geo.RTrad*M_PI*sin(0.589)))*0.5e-3;
Tal[3].A=(0.435*geo.SThgt+0.444*(geo.STwid-geo.RTwid)
                        +(geo.RThgt+0.2222*M_PI*geo.RTrad))*0.5e-6;


Tal[4].L=0;
Tal[4].A=0;

        Tal[5].L=0;
        Tal[5].A=0;

                empty=2;

}

void pathLA::Atooth4(pathdim geo)
{
//set up ellipse function
axis pathA;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth D

        pathA a=geo.AGlen+(geo.RThgt+geo.RTrad)*0.096,
        pathA.b=0.316*(geo.STwid-geo RTwid);
```

```
float ellenA=ellipse(pathA);


Tal[0].L=((ellenA)+(geo.AGlen))*0.5e-3;
Tal[0].A=(pathA.b+0.1168*geo.RTwid+0 096*(geo.RThgt+geo.RTrad))*geo.alDEP*0.5e-6;


Tal[1].L=geo.AGlen*1e-3;
Tal[1].A=geo.RTwid*0 898551*geo.alDEP*1e-6;


pathA.a=geo.AGlen+(geo.RThgt+geo.RTrad)*0.1097;
pathA.b=0.33*(geo.STwid-geo.RTwid);
ellenA=ellipse(pathA);


Tal[2].L=(ellenA+geo.AGlen)*0.5e-3;
Tal[2].A=(pathA.b+0.1037*geo.RTwid+0.1097*(geo.RThgt+geo.RTrad))*geo.alDEP*0.5e-6;

Tal[3].L=0;
Tal[3].A=0;

        Tal[4].L=0;
        Tal[4].A=0;


        Tal[5].L=0;
        Tal[5].A=0;

                empty=3;

}

void pathLA::Atooth5(pathdim geo)
{
//set up ellipse function
axis pathA;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth E

        pathA.a=geo.AGlen+(geo.RThgt+geo.RTrad)*0.089;
        pathA.b=0.723*(geo.STwid-geo.RTwid)/2;
        float ellen=ellipse(pathA);
        Tal[0].L=((geo.AGlen)+(ellen))*0.5e-3;
        Tal[0].A=(pathA.b+0.11852*geo RTwid+(geo.RThgt+geo.RTrad)*0.089)*geo.alDEP*0.5e-6;

        Tal[1].L=geo.AGlen*1e-3;
        Tal[1].A=geo.RTwid*0.925373*geo.alDEP*1e-6;

        pathA.a=geo.AGlen+(geo.RThgt+geo.RTrad)*0.24;
        pathA.b=(geo.STwid-geo.RTwid)/2;
                ellen=ellipse(pathA);
        Tal[2].L=(geo.AGlen+ellen)*0.5e-3;
        Tal[2].A=(1.067*pathA.b+0.075*geo.RTwid+(geo.RThgt+geo RTrad)*0.24)
        *geo alDEP*0.5e-6;

        Tal[3] L=0;

        Tal[3].A=0;

                Tal[4] L=0;
        Tal[4].A=0;

        Tal[5] L=0;
        Tal[5].A=0;

                empty=3;

}

////////////////////////////////////////////////////
void pathLA::Mpath1(pathdim geo)
{
//for (int inc=0; inc<9; inc++) AL[inc].in=0;
AL[0].A=geo.STpwid*(geo DEP*1e-6/5);
AL[0].L=geo.STphgt*1e-3;

double lenA, lenB;
lenA=pythag(geo.sbSTp,geo.tSTet);
COr COtmp;
COtmp.x=geo sbSTp.x-(geo.STpwid/5); COtmp.y=geo.sbSTp y;
lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(lenA+lenB)*0.5e-3;
lenA=pythag(geo.tSTet,geo.tSTea);
AL[1].A=(lenA+(geo.STpwid/5))*0.5*geo.DEP*1e-6;

AL[2].L=((0 785398+geo BETA)*(geo.STrad+5.5*geo STwid/38)+
            (geo.STendT-geo.SThgt-((11*lenA/66)*sin(geo.BETA))))*0.5e-3;
AL[2].A=((53*lenA/67)+(30*geo.STwid/38))*0.5*geo.DEP*1e-6;

//AL[3].in=2,
```

```
//AL[4].in=1;

double LENGTH, AREA;

LENGTH=geo SThgt/(cos(geo.EPSILON))*1e-3;
AREA=(0 6579*geo STwid+0.23077*geo SThgt)*0.5*geo.DEP*1e-6;

axis pathA;
pathA.a=geo SThgt; pathA.b=0 21052*geo.STwid;
float ellen=ellipse(pathA);
AL[3].L=((geo SThgt/(cos(geo.EPSILON))+ellen)*0 5e-3+LENGTH)*0.5;
AL[3].A=0.3947*geo.STwid*geo.DEP*1e-6+AREA;

LENGTH=1.166667*geo.RThgt*1e-3;
AREA=(0.7547*geo RTwid+0.25*geo.RThgt)*0.5*geo.DEP*1e-6;
pathA.a=geo.RTwid*0 1842, pathA.b=geo.RThgt;
ellen=ellipse(pathA);
AL[4].L=((1.166667*geo.RThgt+ellen)*0.5e-3+LENGTH)*0.5;
AL[4].A=0.396222*geo RTwid*geo DEP*1e-6+AREA;

AL[5].L=geo RTrad*1e-3;
AL[5].A=(geo.RTwid+0 434782*geo.RTrad)*geo.DEP*1e-6;

AL[6].L=geo.RTweb*1e-3;
AL[6].A=AL[5].A;
}

//////////////////////////

void pathLA::Mpath2(pathdim geo)
{
//for (int inc=0; inc<9; inc++) AL[inc].in=0;

AL[0].A=geo STpwid*(geo DEP*1e-6/5);
AL[0].L=geo STphgt*1e-3;

double lenA, lenB,

CoOr COtmpA, COtmpB;
COtmpA.x=geo.sbSTp.x-(geo.STpwid/5); COtmpA.y=geo.sbSTp y;
COtmpB.x=geo mbSTp.x+(geo.STpwid/10); COtmpB y=geo.mbSTp y;
lenA=pythag(COtmpA,geo.tSTea);
lenB=pythag(COtmpB,geo.tSTfa);

AL[1].L=(lenA+lenB)*0.5e-3;
lenA=pythag(geo tSTfa,geo tSTea);
AL[1].A=(lenA+(geo.STpwid/5))*0.5*geo.DEP*1e-6;

AL[2].L=((0.785398+geo.BETA)*(geo STrad+2 5*geo.STwid/38)+
        (geo.STrad*M_PI*0.25)+(geo.STrad*(7/18)))*0.5e-3;
AL[2].A=((74*geo.STwid/38)+(6*geo.STrad/20))*0.5*geo.DEP*1e-6;

//AL[3].in=2;
//AL[4].in=1;

float LENGTH=geo.SThgt/(cos(geo.EPSILON))*1e-3;
float AREA=(24*geo.STwid/39)*geo.DEP*1e-6;

axis pathA;
pathA.a=geo SThgt; pathA.b=0.25641*geo.STwid;
float ellen=ellipse(pathA);
AL[3] L=(((geo.SThgt/(cos(geo EPSILON))+ellen)*0.5e-3)+LENGTH)/2,
AL[3].A=0.34210*geo.STwid*geo.DEP*1e-6+AREA;

AL[4].L=1.25*geo.RThgt*1e-3;
AL[4].A=0.90909*geo.RTwid*geo.DEP*1e-6;

AL[5].L=1.1*geo.RTrad*1e-3,
AL[5].A=0.954545*geo RTrad*geo.DEP*1e-6;

AL[6].L=geo.RTweb*1e-3;
AL[6].A=AL[5].A;

}

//////////////////////////

void pathLA::Mpath3(pathdim geo)
{
//for (int inc=0; inc<9; inc++) AL[inc].in=0;

AL[0].A=geo STpwid*(geo.DEP*1e-6/5);
AL[0].L=geo.STphgt*1e-3;

double lenA;

CoOr COtmpA;
COtmpA.x=geo mbSTp.x; COtmpA.y=geo.mbSTp y-geo.STweb;
lenA=pythag(COtmpA,geo.tSTfa);


AL[1].L=geo STweb*1e-3;
AL[1].A=(lenA+(geo.STpwid/10))*geo.DEP*1e-6;
```

////////////////////here/////

```
AL[2].L=((0.785398+geo.BETA)*(geo.STrad+3*geo.STwid/25)+
                (geo STrad*M_PI*0.25)+(geo.STrad*(1/3)))*0.5e-3;
AL[2].A=((47.5*geo STwid/25)+(7.5*geo.STrad/12))*0.5*geo.DEP*1e-6;

//AL[3].in=2;
//AL[4].in=1;

float LENGTH=geo.SThgt/(cos(geo.EPSILON))*1e-3,
float AREA=(24*geo.STwid/39)*geo.DEP*1e-6;

axis pathA;
pathA.a=geo.SThgt; pathA.b=0.25641*geo.STwid;
float ellen=ellipse(pathA);
AL[3].L=((geo.SThgt/(cos(geo.EPSILON))+ellen)*0.5e-3+LENGTH)/2;
AL[3].A=0.28*geo.STwid*geo.DEP*1e-6+AREA;

AL[4].L=1.25*geo.RThgt*1e-3;
AL[4].A=0.90909*geo.RTwid*geo.DEP*1e-6;

AL[5].L=1.1*geo RTrad*1e-3;
AL[5].A=0.954545*geo.RTrad*geo.DEP*1e-6;

AL[6].L=geo.RTweb*1e-3;
AL[6].A=AL[5].A;

}

//////////////////////////

void pathLA..Mpath4(pathdim geo)
{
//for (int inc=0; inc<9; inc++) AL[inc] in=0;

AL[0].A=geo.STpwid*(geo.DEP*1e-6/5);
AL[0].L=geo STphgt*1e-3;

double lenA, lenB;

CoOr COtmpA, COtmpB;
COtmpA.x=geo.sbSTp.x-(geo.STpwid/5); COtmpA y=geo.sbSTp.y;
COtmpB.x=geo.mbSTp.x+(geo.STpwid/10); COtmpB y=geo.mbSTp.y;
lenA=pythag(COtmpA,geo.tSTea);
lenB=pythag(COtmpB,geo.tSTfa);

AL[1].L=(lenA+lenB)*0.5e-3;
lenA=pythag(geo.tSTfa,geo tSTea);
AL[1].A=(lenA+(geo.STpwid/5))*0.5*geo.DEP*1e-6;

AL[2].L=(0.366666*M_PI*geo.STrad+geo.STrad)*0.5e-3;
AL[2].A=((78*geo.STwid/40)+(8*geo.STrad/20))*0.5*geo.DEP*1e-6;

axis pathA;
pathA.a=12*geo.STwid/41;
pathA.b=geo.SThgt;
float ellen=ellipse(pathA);
AL[3].L=(geo.SThgt+ellen)*0.5e-3;
AL[3].A=(69*geo.STwid/80)*geo.DEP*1e-6;

AL[4].L=1.25*geo.RThgt*1e-3;
AL[4].A=0 90909*geo.RTwid*geo.DEP*1e-6;

AL[5].L=1.1*geo.RTrad*1e-3;
AL[5].A=0 954545*geo.RTrad*geo.DEP*1e-6;

AL[6].L=geo.RTweb*1e-3;
AL[6].A=AL[5].A;

}

//////////////////////////
void pathLA::Mpath5(pathdim geo)
{
//for (int inc=0; inc<9; inc++) AL[inc] in=0;

AL[0].A=geo.STpwid*(geo DEP*1e-6/5);
AL[0].L=geo.STphgt*1e-3;

double lenA, lenB;
lenA=pythag(geo.sbSTp,geo.tSTet);
CoOr COtmp;
COtmp.x=geo.sbSTp.x-(geo.STpwid/5); COtmp y=geo.sbSTp y;
lenB=pythag(COtmp,geo tSTea);

AL[1].L=(lenA+lenB)*0.5e-3;
lenA=pythag(geo.tSTet,geo tSTea);
AL[1].A=(lenA+(geo.STpwid/5))*0.5*geo.DEP*1e-6;

axis pathA;
pathA.b=1.1707*geo.STrad;
pathA.a=1.0243*geo.STrad;
float ellen;
ellen=ellipse(pathA);

AL[2] L=(ellen+geo.STendT-geo SThgt)*0.5e-3;
AL[2].A=((0.975*geo.STwid)+
((geo.STrad+0 92*geo STwid)/cos(geo.BETA))
```

```
)*0.5*geo.DEP*1e-6,


pathA.a=12*geo.STwid/41;
pathA.b=geo.SThgt;
ellen=ellipse(pathA);

AL[3].L=(geo.SThgt+ellen)*0.5e-3;
AL[3].A=(69*geo.STwid/80)*geo.DEP*1e-6;

//////////AIRGAP//////

pathA.a=geo.RTwid*0.1842; pathA.b=geo.RThgt;
ellen=ellipse(pathA);
float LENGTH=(1.166667*geo RThgt+ellen)*0.5e-3;
float AREA=0.396222*geo.RTwid*geo.DEP*1e-6;

AL[4].L=(1.166667*geo.RThgt*1e-3+LENGTH)/2;
AL[4].A=((0.7547*geo.RTwid+0.25*geo.RThgt)*0.5*geo.DEP*1e-6)+AREA;

AL[5].L=geo.RTrad*1e-3;
AL[5].A=(geo.RTwid+0 434782*geo.RTrad)*geo.DEP*1e-6;

AL[6].L=geo.RTweb*1e-3;
AL[6].A=AL[5].A,

}

//////////////////////

void pathLA::Mtooth1(pathdim geo)
{
//set up ellipse function
axis pathA,pathB;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth A

                pathA.a=geo.mAGlen+geo.SThgt/4;
                pathA.b=0.17*geo.RTwid;
                float ellen=ellipse(pathA);
                Tal[0].L=((geo.mAGlen*1.3)+(ellen))*0.5e-3;
                Tal[0].A=((geo.SThgt/4)+0.19*geo.RTwid)*geo.ADEP*0.5e-6;

                Tal[1].L=geo.mAGlen*1.238e-3;
                Tal[1].A=0.5*geo.RTwid*0.1275*geo.ADEP*1e-6;

                Tal[2].L=geo.mAGlen*1.094e-3,
                Tal[2].A=geo.RTwid*0.32*geo.ADEP*1e-6;

                pathA.a=geo.mAGlen+0.152*geo.RThgt;
                pathA.b=0.07*geo.RTwid;
                ellen=ellipse(pathA);

                Tal[3].L=(geo.mAGlen*1.0588+ellen)*0.5e-3;
                Tal[3].A=(geo.RTwid*1.55+0.1364*geo.RThgt)*geo.ADEP*0.5e-6;

                pathB.a=geo.RTrad*0.062*M_PI+geo.mAGlen;
                pathB.b=geo.STwid*0.56756,
                float ellenB=ellipse(pathB);
                Tal[4].L=(ellenB+ellen)*0 5e-3;

                Tal[4].A=(geo.STwid*0.216+0.045*geo.RTwid+0.8088*geo.RThgt+
                M_PI*geo.RTrad*0.06)*geo.ADEP*0.5e-6;


                Tal[5].L=0;
                                Tal[5].A=0;

                                empty=1;

}

void pathLA::Mtooth2(pathdim geo)
{
//set up ellipse function
axis pathA;
axis pathB;

//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth B

                pathA.a=geo.mAGlen+geo.SThgt;
                pathA.b=0.2778*geo.RTwid;
                float ellenA=ellipse(pathA);

                pathB.a=geo.mAGlen+0.027*geo.SThgt;
                pathB.b=0.0488*geo.RTwid;
                float ellenB=ellipse(pathB);

                Tal[0] L=((ellenA)+(ellenB))*0.5e-3;
                Tal[0].A=(0.28*geo.RTwid+0.92*geo SThgt)*geo.ADEP*1e-6;


                Tal[1].L=(1.2*geo mAGlen+(ellenB))*0 5e-3;
```

```
Tal[1].A=(geo.SThgt*0.0135+geo.RTwid*0.0864)*geo.ADEP*1e-6;

Tal[2].L=geo.mAGlen*1.1167e-3;
Tal[2].A=geo.RTwid*0.3963*geo.ADEP*1e-6;

Tal[3].L=(geo.mAGlen*2.35)*0.5e-3;
Tal[3].A=(geo RTwid*0.08)*geo.ADEP*1e-6;

pathA.a=geo RThgt*1.2+geo.mAGlen;
pathA.b=geo.STwid*0.1833,
ellenA=ellipse(pathA);
Tal[4].L=(geo.mAGlen*1.4+(ellenA))*0.5e-3,
Tal[4].A=(geo.STwid*0.154+1.2*geo.RThgt)*geo.ADEP*0.5e-6;


Tal[5].L=0;
                Tal[5].A=0;

                empty=1;

}

void pathLA::Mtooth3(pathdim geo)
{
//set up ellipse function
axis pathA;

//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth C

                pathA.a=geo.mAGlen+0.10714*geo.SThgt;
                pathA.b=0.097*geo.RTwid;
                float ellenA=ellipse(pathA);

                Tal[0].L=(ellenA+geo.mAGlen*1.125)*0.5e-3;
                Tal[0].A=(geo.SThgt*0.10714+0.1791*geo.RTwid)*geo.ADEP*0.5e-6;

                Tal[1].L=geo.mAGlen*1.05e-3;
                Tal[1].A=geo.RTwid*0.3731*geo.ADEP*1e-6,

                Tal[2].L=geo.mAGlen*1.125e-3;
                Tal[2].A=geo.RTwid*0.07836*geo.ADEP*1e-6;

                pathA.a=geo.RThgt*0.6786+geo.mAGlen;
                pathA.b=geo.RTwid*0.186567;
                ellenA=ellipse(pathA);
                Tal[3] L=(geo.mAGlen*1.125+(ellenA))*0.5e-3;
                Tal[3].A=(geo RTwid*0.16418+0.67857*geo.RThgt)*geo.ADEP*0.5e-6;
                Tal[4].L=0;
                Tal[4].A=0;
                Tal[5].L=0;
                Tal[5].A=0;

                empty=2;

}

void pathLA::Mtooth4(pathdim geo)
{
//set up ellipse function
axis pathA,

//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth D

                pathA.a=geo.mAGlen+0.21875*geo.SThgt,
                pathA.b=0.135135*geo RTwid;
                float ellenA=ellipse(pathA);

                Tal[0].L=(ellenA+geo.mAGlen*1.05)*0.5e-3;
                Tal[0].A=(geo.SThgt*0.21875+0.175675*geo.RTwid)*0.5*geo.ADEP*1e-6;

                Tal[1].L=geo.mAGlen*1.0175e-3;
                Tal[1].A=geo RTwid*0.37162*geo.ADEP*1e-6;

                Tal[2].L=geo.mAGlen*1.05e-3,
                Tal[2].A=geo.RTwid*0 0777*geo.ADEP*1e-6;

                pathA.a=geo.RThgt*0.591+geo.mAGlen;
                pathA.b=geo.RTwid*0.135135;
                ellenA=ellipse(pathA);
                Tal[3].L=(geo.mAGlen*1.0175+(ellenA))*0.5e-3;
                Tal[3].A=(geo.RTwid*0 162162+0.5959*geo.RThgt)*0.5*geo.ADEP*1e-6;

                pathA.a=geo.RThgt*0.5959+1.03*geo mAGlen;
                pathA b=geo RTwid*0.135135;
                ellenA=ellipse(pathA);
                pathA a=geo.RThgt+geo mAGlen+geo.RTrad*0.5;
                pathA.b=geo.RTwid*0.38679;
                float ellenB=ellipse(pathA);
                float arc=0.011898*geo RTrad*M_PI*M_PI,
                Tal[4].L=(ellenA+ellenB-1.4062*arc)*0.5e-3;
                Tal[4].A=((M_PI*2.8125*geo.RTrad/12)+
                                                0.5625*geo.RThgt+0.31132*geo.STwid)*0.5e-6*geo.ADEP;
                Tal[5] L=0;
```

```
                Tal[5].A=0;

                empty=1;

}
void pathLA::Mtooth5(pathdim geo)
{
//set up ellipse function
axis pathA;

//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth E

                pathA.a=geo.mAGlen+0.3846*geo.SThgt;
                pathA.b=0.191011*geo.RTwid;
                float ellenA=ellipse(pathA);

                Tal[0].L=(ellenA+1.076*geo.mAGlen)*0.5e-3;
                Tal[0].A=(geo.SThgt*0.21875+0.175675*geo.RTwid)*0.5*geo.ADEP*1e-6;

                Tal[1].L=1.038*geo.mAGlen*0.5e-3;
                Tal[1].A=geo.RTwid*0.08888*geo.ADEP*1e-6;

                Tal[2].L=geo.mAGlen*1.03e-3;
                Tal[2].A=geo.RTwid*0.32608*geo.ADEP*1e-6;

                Tal[3].L=(geo.mAGlen*1.03)*1e-3,
                Tal[3].A=geo.RTwid*0.08152*geo.ADEP*1e-6;

                Tal[4].L=1.5*geo.mAGlen*1e-3;
                Tal[4].A=(0.16304*geo.RTwid+0.28947*geo.RThgt)*0.5e-6*geo.ADEP;

                pathA.a=geo.mAGlen+geo RThgt+(M_PI*geo.RTrad/12)*cos(12.86)+
                                                        (M_PI*geo.RTrad/20)*cos(12.86);
                pathA.b=0.328*geo.STwid;
                ellenA=ellipse(pathA);
                if (2.4166667*geo.mAGlen+ellenA<0.0071388*M_PI*M_PI*geo.RTrad)
                Tal[5] L=(2.4166667*geo.mAGlen+ellenA)*1e-3;
                else
                Tal[5].L=(2.4166667*geo.mAGlen+ellenA-(0.0071388*M_PI*M_PI*geo.RTrad))*1e-3;

                Tal[5].A=(0.7*geo.RThgt+0.1333*geo.RTrad+0.264*geo.STwid)*0.5e-6*geo.ADEP;

                empty=0,

}
////////////////////////////////////////////////////
void pathLA::Upath1(pathdim geo)
{

AL[0].A=geo.STpwid*(geo.DEP*1e-6/10);
AL[0].L=geo.STphgt*1e-3;

double lenA, lenB;
lenA=pythag(geo.sbSTp,geo.tSTet);
CoOr COtmp,COtmp1;
COtmp.x=geo.sbSTp.x-(geo.STpwid/5); COtmp y=geo sbSTp y;
lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(0.5*(lenA+lenB)+lenA)*0.5e-3;
lenA=pythag(geo.tSTet,geo.tSTea);
AL[1].A=(0.5*lenA+(geo.STpwid/10))*0.5*geo.DEP*1e-6;

COtmp1.x=0;COtmp1.y=0; COtmp.x=(0.48148*lenA*sin(geo.BETA));
COtmp.y=0.11428*(geo.STrad+geo.STwid);
lenB=pythag(COtmp1,COtmp);
AL[2].L=(((0.90123*lenA*sin(geo.BETA))+lenB)*0.5+geo STrad)*0.5e-3;
AL[2].A=((32*lenA/90)+(21*geo.STwid/45))*0.5*geo.DEP*1e-6;


double LENGTHA,LENGTHB;

LENGTHA=(geo.RTrad+geo.RThgt);
LENGTHB=(geo.RTrad+geo.RTwid);


axis pathA,pathB;
pathA.a=LENGTHA; pathA.b=0.2*LENGTHB;
pathB.a=LENGTHA; pathB.b=0.2167*LENGTHB;
float ellenA=ellipse(pathA);
float ellenB=ellipse(pathB);


AL[3].L=(ellenA+ellenB+1 81666*LENGTHA+0.5217*geo.RTrad)*0.2e-3;
AL[3].A=(1.28783*geo RTrad+0.9458*geo RTwid+geo RThgt)*geo.DEP*0.5e-6;

//AL[5].in=2;
//AL[6].in=1;

AL[4].L=geo.RTweb*1e-3;
AL[4].A=0.6333*LENGTHB*geo.DEP*1e-6;


AL[5].L=0;
AL[5].A=0;
```

```
AL[6].L=0;
AL[6].A=0;
}

//////////////////////

void pathLA::Upath2(pathdim geo)
{

AL[0].A=geo.STpwid*(geo.DEP*1e-6/10);
AL[0].L=geo.STphgt*1e-3;

double lenA, lenB;
lenA=pythag(geo.sbSTp,geo.tSTet);
CoOr COtmp,//Cotmp1;
COtmp.x=geo.sbSTp.x-(geo.STpwid/5); COtmp.y=geo.sbSTp y;
lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(0.5*(lenA+lenB)+lenB)*0.5e-3;
lenA=pythag(geo.tSTet,geo.tSTea);
AL[1].A=(0.5*lenA+(geo STpwid/10))*0.5*geo.DEP*1e-6;

axis pathA,pathB;
pathA.a=0.089*geo.STwid; pathA.b=geo.STrad;
pathB.a=0.5*geo.STrad; pathB b=geo.STrad;
float ellenA=ellipse(pathA);
float ellenB=ellipse(pathB);
pathA.a=0.57*(geo.STwid+geo.STrad); pathA.b=0.5*geo STendT;
pathB.a=0.6*geo.STrad; pathB.b=0.0667*geo.STendT;
float ellenC=ellipse(pathA);
float ellenD=ellipse(pathB);


AL[2].L=(0.5*ellenA+0.5*ellenB+(geo.BETA/M_PI)*(ellenC+ellenD))*1e-3,
AL[2].A=(0.4375*lenA+0.48889*geo.STwid)*0.5*geo.DEP*1e-6;


double LENGTHA;

LENGTHA=(geo STrad+geo.SThgt);

pathA.a=0.3793*LENGTHA; pathA.b=0.22857*geo.STwid;

 ellenA=ellipse(pathA);


AL[3].L=1e-3*(2*ellenA)/3;
AL[3].A=(0.6*geo.STwid+0 46667*(geo.SThgt+geo.STrad))*geo.DEP*0.5e-6;

pathA.a=0.1818*geo RTwid; pathA.b=geo.RThgt;

ellenA=ellipse(pathA);

AL[4] L=(ellenA+geo.RTrad+0.523598*geo.RTrad+0.5778*geo.RTrad+
                                0.654498*geo.RTrad)*(1/3)*1e-3;
AL[4].A=(0.371*geo.RTrad+0.2615*geo.RTwid+geo.RThgt+0.4*
(geo.RTwid+geo.RTrad))*geo.DEP*0.5e-6;



AL[5].L=geo.RTweb*1e-3,
AL[5].A=0 48837*(2*geo RTrad+geo.RTwid)*geo.DEP*1e-6.;

AL[6].L=0;
AL[6].A=0;
}


//////////////////////

void pathLA.:Upath3(pathdim geo)
{

AL[0].A=geo.STpwid*(geo.DEP*1e-6/10);
AL[0].L=geo.STphgt*1e-3,
double lenA, lenB;

CoOr COtmp,COtmp1;
COtmp.x=geo.sbSTp.x-(geo.STpwid/5); COtmp y=geo.sbSTp y;
COtmp1.x=geo.mbSTp.x+(1.5*geo.STpwid/5);COtmp1.y=geo.mbSTp.y;
lenA=pythag(COtmp1,geo.tSTfa);
lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(0.5*(lenA+lenB)+lenB)*0.5e-3;
lenA=pythag(geo.tSTfa,geo.tSTea);
AL[1].A=(0.5*lenA+(geo.STpwid/10))*0.5*geo DEP*1e-6,

axis pathA;
pathA.a=0.089*geo.STwid, pathA.b=geo.SThgt,
float ellenA=ellipse(pathA);

COtmp x=0 234*geo STwid; COtmp y=geo.SThgt;
COtmp1.x=0; COtmp1 y=0;
lenA=pythag(COtmp1,COtmp);
```

```
AL[2].L=(((ellenA+lenA+0 5*geo.SThgt+(M_PI/12)*geo.RTrad)+3*geo.STrad)/4)*1e-3;
AL[2].A=(geo.SThgt+0.7446*geo.STwid+0.375*M_PI*geo.STrad)*0.5*geo DEP*1e-6;


double LENGTHA;

LENGTHA=(2*geo.RTrad+geo.RTwid);

pathA.a=0.136136*geo.RTwid; pathA.b=geo.RThgt;

ellenA=ellipse(pathA);
COtmp.x=0.212*geo.RTwid; COtmp y=geo RThgt;
lenA=pythag(COtmp1,COtmp);

AL[3].L=1e-3*(ellenA+3.43478*geo.RTrad+lenA+0.5*geo.RThgt)/4;
AL[3].A=(0.5*geo.RTwid+geo.RThgt+0.23*M_PI*geo.RTrad+
                                0.3488*LENGTHA)*geo.DEP*0.5e-6;

AL[4].A=0.3837e-3*LENGTHA;
AL[4].L=geo.RTweb*1e-3;

AL[5].L=0.
AL[5].A=0;

AL[6].L=0;
AL[6].A=0;
}

void pathLA::Upath4(pathdim geo)
{

AL[0].A=geo.STpwid*(geo.DEP*1e-6/10);
AL[0].L=geo.STphgt*1e-3,

double lenA, lenB,

CoOr COtmp,COtmp1;
COtmp.x=geo.sbSTp.x-(geo.STpwid/5); COtmp.y=geo.sbSTp.y;
COtmp1.x=geo.mbSTp.x+0.1*geo.STpwid, COtmp1.y=geo.mbSTp y;
lenA=pythag(COtmp1,geo.tSTfa);
lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(0.5*(lenA+lenB)+lenA)*0 5e-3;
lenA=pythag(geo.tSTfa,geo.tSTea);
AL[1].A=(0.5*lenA+(geo.STpwid/10))*0 5*geo.DEP*1e-6;

ecis pathA,pathB;
pathA.a=0.089*geo.STwid; pathA.b=geo.STrad,
pathB.a=0.5*geo.STrad; pathB.b=geo.STrad;
float ellenA=ellipse(pathA);
float ellenB=ellipse(pathB);

AL[2].L=(0.5*ellenA+0.5*ellenB)*1e-3;
AL[2].A=(0.6*(geo.STrad+0.5*geo.STwid)+0.48889*geo.STwid)*0.5*geo DEP*1e-6;


double LENGTHA;

LENGTHA=(geo STrad+geo SThgt);

pathA.a=0.3793*LENGTHA; pathA.b=0.22857*geo.STwid;

 ellenA=ellipse(pathA);


AL[3] L=1e-3*(2*ellenA)/3;
AL[3].A=(0.6*geo.STwid+0.46667*(geo.SThgt+geo.STrad))*geo.DEP*0.5e-6,

pathA.a=0 1818*geo.RTwid; pathA.b=geo.RThgt;

ellenA=ellipse(pathA);

AL[4] L=(ellenA+geo.RTrad+0.523598*geo.RTrad+0.5778*geo RTrad+
                                0 654498*geo RTrad)*(1/3)*1e-3;
AL[4].A=(0.371*geo RTrad+0.2615*geo.RTwid+geo.RThgt+0.4*
        (geo.RTwid+geo.RTrad))*geo.DEP*0.5e-6;


AL[5].L=geo.RTweb*1e-3,
AL[5].A=0.48837*(2*geo.RTrad+geo.RTwid)*geo.DEP*1e-6;

AL[6] L=0;
AL[6].A=0;
}


void pathLA::Upath5(pathdim geo)
{

AL[0].A=geo.STpwid*(geo.DEP*1e-6/10);
AL[0].L=geo.STphgt*1e-3;

double lenA;//lenB;
```

```
CoOr COtmp,COtmp1;
COtmp.x=0; COtmp.y=geo.mbSTp y-geo.STweb;
COtmp1.x=geo.mbSTp.x+(geo STpwid/10),COtmp1.y=geo.mbSTp y;
lenA=pythag(COtmp1,geo.tSTfa);
//lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(lenA)*0.5e-3;
lenA=pythag(geo tSTfa,COtmp);
AL[1].A=(lenA+(geo.STpwid/10))*0.5*geo.DEP*1e-6;

axis pathA;
pathA.a=0.089*geo.STwid; pathA.b=geo.SThgt;
float ellenA=ellipse(pathA);

COtmp.x=0.234*geo.STwid; COtmp y=geo.SThgt;
COtmp1.x=0; COtmp1.y=0;
lenA=pythag(COtmp1,COtmp);

AL[2].L=(((ellenA+lenA+0.5*geo.SThgt+(M_PI/12)*geo.RTrad)+3*geo.STrad)/4)*1e-3;
AL[2].A=(geo.SThgt+0.7446*geo.STwid+0.375*M_PI*geo.STrad)*0.5*geo.DEP*1e-6;


double LENGTHA;

LENGTHA=(2*geo.RTrad+geo RTwid);

pathA.a=0.136136*geo.RTwid; pathA.b=geo.RThgt;

ellenA=ellipse(pathA);
COtmp.x=0.212*geo.RTwid; COtmp y=geo.RThgt;
lenA=pythag(COtmp1,COtmp);

AL[3].L=1e-3*(ellenA+3.43478*geo.RTrad+lenA+0.5*geo.RThgt)/4;
AL[3].A=(0.5*geo.RTwid+geo.RThgt+0.23*M_PI*geo.RTrad+
                                0.3488*LENGTHA)*geo.DEP*0.5e-6;


AL[4].A=0.3837e-6*LENGTHA*geo.DEP;
AL[4].L=geo.RTweb*1e-3;

AL[5].L=0;
AL[5].A=0;

AL[6].L=0;
AL[6].A=0;
}

void pathLA::Upath6(pathdm geo)
{

AL[0].A=geo.STpwid*(geo.DEP*1e-6/10);
AL[0].L=geo.STphgt*1e-3;

double lenA;// lenB;

CoOr COtmp,COtmp1;
COtmp.x=0; COtmp y=geo.mbSTp y-geo.STweb;
COtmp1.x=geo.mbSTp.x+(geo.STpwid/10);COtmp1.y=geo.mbSTp y;
lenA=pythag(COtmp1,geo.tSTfa);
//lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(lenA)*0.5e-3;
lenA=pythag(geo.tSTfa,COtmp);
AL[1].A=(lenA+(geo.STpwid/10))*0.5*geo.DEP*1e-6;

axis pathA,pathB;
pathA.a=0.089*geo.STwid; pathA.b=geo.STrad.
pathB.a=0.5*geo.STrad; pathB.b=geo STrad;
float ellenA=ellipse(pathA);
float ellenB=ellipse(pathB);

AL[2].L=(0.5*ellenA+0.5*ellenB)*1e-3;
AL[2].A=(0.6*(geo.STrad+0.5*geo.STwid)+0.48889*geo.STwid)*0.5*geo.DEP*1e-6;

double LENGTHA;

LENGTHA=(geo STrad+geo.SThgt);

pathA.a=0.3793*LENGTHA; pathA.b=0.22857*geo.STwid;

ellenA=ellipse(pathA);


AL[3].L=1e-3*(2*ellenA)/3;
AL[3].A=(0.6*geo.STwid+0.46667*(geo.SThgt+geo.STrad))*geo.DEP*0.5e-6,


LENGTHA=(2*geo.RTrad+geo.RTwid);

pathA.a=0.136136*geo.RTwid; pathA.b=geo.RThgt;

ellenA=ellipse(pathA);
```

```
COtmp.x=0.212*geo.RTwid; COtmp y=geo.RThgt;
lenA=pythag(COtmp1,COtmp);

AL[4].L=1e-3*(ellenA+3.43478*geo.RTrad+lenA+0.5*geo.RThgt)/4;
AL[4].A=(0.5*geo.RTwid+geo.RThgt+0.23*M_PI*geo.RTrad+
                                0.3488*LENGTHA)*geo.DEP*0.5e-6;


AL[5].A=0.3837e-6*LENGTHA*geo.DEP;
AL[5] L=geo RTweb*1e-3;


AL[6].L=0;
AL[6].A=0;
}


void pathLA::Upath7(pathdim geo)
{

AL[0].A=geo.STpwid*(geo.DEP*1e-6/10);
AL[0].L=geo.STphgt*1e-3;

double lenA, lenB;

CoOr COtmp,COtmp1;
COtmp.x=geo.sbSTp.x-(geo.STpwid/5); COtmp.y=geo.sbSTp y;
COtmp1.x=geo.mbSTp.x+0.1*geo.STpwid; COtmp1.y=geo.mbSTp.y;
lenA=pythag(COtmp1,geo tSTfa);
lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(0.5*(lenA+lenB)+lenA)*0.5e-3;
lenA=pythag(geo tSTfa,geo.tSTea);
AL[1].A=(0.5*lenA+(geo.STpwid/10))*0.5*geo.DEP*1e-6,

axis pathA;
pathA.a=0 089*geo.STwid; pathA.b=geo.SThgt;
float ellenA=ellipse(pathA);

COtmp.x=0.234*geo.STwid, COtmp.y=geo.SThgt;
COtmp1.x=0; COtmp1.y=0;
lenA=pythag(COtmp1,COtmp);

AL[2].L=(((ellenA+lenA+0.5*geo SThgt+(M_PI/12)*geo.RTrad)+3*geo.STrad)/4)*1e-3,
AL[2].A=(geo.SThgt+0.7446*geo.STwid+0.375*M_PI*geo.STrad)*0.5*geo.DEP*1e-6;


pathA.a=0.1818*geo.RTwid; pathA.b=geo RThgt;
ellenA=ellipse(pathA);

AL[3].L=(ellenA+geo.RTrad+0.523598*geo.RTrad+0.5778*geo RTrad+
                                0.654498*geo.RTrad)*(1/3)*1e-3,
AL[3].A=(0.371*geo.RTrad+0.2615*geo.RTwid+geo.RThgt+0.4*
            (geo.RTwid+geo.RTrad))*geo.DEP*0.5e-6;



AL[4].L=geo.RTweb*1e-3;
AL[4].A=0.48837*(2*geo.RTrad+geo.RTwid)*1e-6*geo DEP;

AL[5].L=0;
AL[5].A=0,

AL[6].L=0;
AL[6].A=0;
}

void pathLA::Upath8(pathdim geo)
{

AL[0].A=geo.STpwid*(geo.DEP*1e-6/10);
AL[0].L=geo STphgt*1e-3;

double lenA, lenB;

CoOr COtmp,COtmp1;
COtmp.x=geo.sbSTp.x-(geo.STpwid/5); COtmp y=geo.sbSTp y;
COtmp1.x=geo.mbSTp.x+(1.5*geo.STpwid/5);COtmp1.y=geo.mbSTp y;
lenA=pythag(COtmp1,geo.tSTfa);
lenB=pythag(COtmp,geo.tSTea);

AL[1] L=(0.5*(lenA+lenB)+lenB)*0.5e-3;
lenA=pythag(geo.tSTfa,geo.tSTea);
AL[1].A=(0.5*lenA+(geo STpwid/10))*0.5*geo.DEP*1e-6,
axis pathA,pathB;
pathA.a=0.089*geo.STwid; pathA.b=geo.STrad;
pathB a=0 5*geo STrad; pathB b=geo STrad;
float ellenA=ellipse(pathA);
float ellenB=ellipse(pathB);

AL[2].L=(0.5*ellenA+0.5*ellenB)*1e-3;
AL[2].A=(0.6*(geo.STrad+0.5*geo.STwid)+0.48889*geo.STwid)*0.5*geo.DEP*1e-6,


double LENGTHA;
```

```
LENGTHA=(geo.STrad+geo.SThgt);

pathA.a=0.3793*LENGTHA; pathA.b=0.22857*geo.STwid;

 ellenA=ellipse(pathA);


AL[3].L=1e-3*(2*ellenA)/3;
AL[3].A=(0.6*geo.STwid+0.46667*(geo.SThgt+geo.STrad))*geo.DEP*0.5e-6;



LENGTHA=(2*geo.RTrad+geo.RTwid);

pathA.a=0 136136*geo.RTwid, pathA.b=geo RThgt;

ellenA=ellipse(pathA);
COtmp.x=0.212*geo.RTwid; COtmp y=geo.RThgt;
lenA=pythag(COtmp1,COtmp);

AL[4].L=1e-3*(ellenA+3.43478*geo.RTrad+lenA+0.5*geo.RThgt)/4;
AL[4].A=(0.5*geo.RTwid+geo.RThgt+0.23*M_PI*geo.RTrad+
                                0.3488*LENGTHA)*geo.DEP*0.5e-6;


AL[5].A=0.3837e-6*LENGTHA*geo.DEP;
AL[5].L=geo RTweb*1e-3;

AL[6].L=0;
AL[6].A=0;


}


//////////////////////

void pathLA::Upath9(pathdim geo)
{

AL[0].A=geo.STpwid*(geo.DEP*1e-6/10);
AL[0].L=geo.STphgt*1e-3;

double lenA, lenB;
lenA=pythag(geo.sbSTp,geo.tSTet);
CoOr COtmp,//,Cotmp1;
COtmp.x=geo.sbSTp.x-(geo.STpwid/5); COtmp y=geo.sbSTp y;
lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(0.5*(lenA+lenB)+lenB)*0.5e-3;
lenA=pythag(geo.tSTet,geo.tSTea);
AL[1].A=(0.5*lenA+(geo STpwid/10))*0.5*geo.DEP*1e-6;

axis pathA,pathB;
pathA.a=0.094*geo.STwid; pathA.b=geo STrad;
pathB.a=0.516*geo.STrad, pathB.b=geo STrad;
float ellenA=ellipse(pathA);
float ellenB=ellipse(pathB);
pathA.a=0.58*(geo.STwid+geo.STrad); pathA.b=0.43*geo.STendT;
pathB.a=0 48*geo.STrad, pathB.b=0.0333*geo.STendT;
float ellenC=ellipse(pathA);
float ellenD=ellipse(pathB);


AL[2].L=(0.5*ellenA+0.5*ellenB+(geo.BETA/M_PI)*(ellenC+ellenD))*1e-3;
AL[2].A=(0.4*lenA+0 42857*geo.STwid)*0.5*geo.DEP*1e-6,


double LENGTHA;

LENGTHA=(geo.STrad+geo.SThgt);

pathA.a=0 3793*LENGTHA; pathA.b=0.1587*geo.STwid,

 ellenA=ellipse(pathA);


AL[3].L=1e-3*(2*ellenA)/3;
AL[3].A=(0.5*geo.STwid+0.6*(geo.SThgt+geo.STrad))*geo.DEP*0.5e-6;

pathA.a=0.1818*geo.RTwid; pathA.b=geo.RThgt;

ellenA=ellipse(pathA);

AL[4].L=(ellenA+geo.RTrad+0.523598*geo.RTrad+0.5778*geo RTrad+
                                0.654498*geo RTrad)*(1/3)*1e-3;
AL[4].A=(0.371*geo.RTrad+0.2615*geo.RTwid+geo.RThgt+0.4*
(geo.RTwid+geo.RTrad))*geo.DEP*0.5e-6;



AL[5].L=geo.RTweb*1e-3;
AL[5].A=0.48837e-6*(2*geo.RTrad+geo.RTwid)*geo.DEP;
```

```
AL[6].L=0;
AL[6].A=0;
}

void pathLA::Upath10(pathdim geo)
{

AL[0].A=geo STpwid*(geo DEP*1e-6/10);
AL[0].L=geo.STphgt*1e-3;

double lenA, lenB;
lenA=pythag(geo.sbSTp,geo.tSTet);
CoOr COtmp,COtmp1;
COtmp.x=geo.sbSTp.x-(geo.STpwid/5); COtmp y=geo.sbSTp y;
lenB=pythag(COtmp,geo.tSTea);

AL[1].L=(0.5*(lenA+lenB)+lenA)*0.5e-3;
lenA=pythag(geo.tSTet,geo.tSTea);
AL[1].A=(0.5*lenA+(geo.STpwid/10))*0.5*geo.DEP*1e-6;

COtmp1.x=0;COtmp1.y=0; COtmp.x=(0.48148*lenA*sin(geo.BETA));
COtmp.y=0.11428*(geo.STrad+geo.STwid);
lenB=pythag(COtmp1,COtmp);
AL[2].L=(((0.90123*lenA*sin(geo.BETA))+lenB)*0 5+geo.STrad)*0.5e-3;
AL[2].A=((32*lenA/90)+(21*geo.STwid/45))*0.5*geo.DEP*1e-6;


double LENGTHA,LENGTHB;

LENGTHA=(geo.RTrad+geo.RThgt);
LENGTHB=(geo.RTrad+geo RTwid);


axis pathA,pathB;
pathA.a=LENGTHA; pathA.b=0.2*LENGTHB;
pathB a=LENGTHA, pathB.b=0.2167*LENGTHB;
float ellenA=ellipse(pathA);
float ellenB=ellipse(pathB);


AL[3].L=(ellenA+ellenB+1.81666*LENGTHA+0.5217*geo.RTrad)*0.2e-3;
AL[3].A=(1.28783*geo.RTrad+0.9458*geo.RTwid+geo.RThgt)*geo.DEP*0.5e-6,

//AL[5].in=2;
//AL[6].in=1;

AL[4].L=geo.RTweb*1e-3;
AL[4].A=0 6333*LENGTHB*geo.DEP*1e-6;


AL[5].L=0;
AL[5].A=0;

AL[6].L=0;
AL[6].A=0;
}

//////////////////////////

void pathLA::Utooth1(pathdim geo)
{
//set up ellipse function
axis pathA,pathB,pathC;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth  mid left
                float Gap=geo.RTrad-0.5*geo.STwid;

                pathA.a=geo uAGlen+geo.SThgt+geo.STrad+(geo.STrad*0.214);
                pathA.b=Gap+0.5*geo RTwid;
                float ellenA=ellipse(pathA);
                pathB.a=geo.uAGlen+geo.SThgt*0.6111;
                pathB.b=Gap+0.15853*geo.RTwid;
                float ellenB=ellipse(pathB);
                pathC.a=0.49*geo.STrad;
                pathC.b=0.4035*geo.STrad;
float arc=ellipse(pathC);

Tal[0].L=(ellenA+ellenB-arc)*0.5e-3;

Tal[0].A=(0.2625*geo.RTwid+0.9163*geo.STrad+0.38889*geo.SThgt)
                *geo.ADEP*0.5e-6;

//////////////////////////////
float Len=sqrt(pathB.b*pathB b+(0.22727*geo.SThgt+geo.uAGlen)*
                                (0.22727*geo.SThgt+geo.uAGlen));

Tal[1].L=(ellenB+Len)*0.5e-3;
Tal[1].A=(geo.RTwid*0.134+0.4722*geo.SThgt)*geo.ADEP*0.5e-6;

////

pathA.a=geo.STwid*0.0254+Gap;
pathA.b=0.2778*geo.RTrad+geo.uAGlen;
```

```
ellenA=ellipse(pathA);
Tal[2].L=(Len+ellenA)*0.5e-3;
Tal[2].A=(0.02439*geo.RTwid+0.2777*geo.RThgt+
                        0.0254*geo STwid+0.1667*geo.SThgt)
                        *geo.ADEP*0.5e-6,


pathB.a=geo.STwid*0.12711+Gap;
pathB.b=1.333333*geo.RThgt+geo uAGlen;
ellenB=ellipse(pathB);
Tal[3].L=(ellenA+ellenB)*0.5e-3;
Tal[3].A=(geo.RThgt+
                        0.101695*geo.STwid)
                        *geo.ADEP*0.5e-6;

pathA.a=geo.STwid*0.322+Gap,
pathA.b=0.84848*geo.RTrad+geo.uAGlen+geo.RThgt;
ellenA=ellipse(pathA);
            arc=geo RTrad*0.285;
Tal[4].L=(ellenA+ellenB-arc)*0.5e-3,
            Tal[4].A=(0.18644*geo.STrad+
                        0.64141*geo.RTrad)
                        *geo.ADEP*0.5e-6;

            Tal[5].L=0;
            Tal[5].A=0;

                    empty=1;

}

void pathLA::Utooth2(pathdim geo)
{
//set up ellipse function
axis pathA,pathB;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap tooth mid right
            float Gap=geo.RTrad-0.5*geo.STwid;

            pathA.a=geo.uAGlen+geo.RThgt*0.70588;
            pathA.b=Gap+0.05084*geo STwid;
            float ellenA=ellipse(pathA);
            //pathB a=geo.uAGlen+geo.SThgt*0 6111;
            //pathB b=Gap+0.15853*geo.RTwid;
            //float ellenB=ellipse(pathB);
            float arc=0.05825*geo STrad,

Tal[0].L=((geo.uAGlen+geo.RThgt)+(geo.RTrad-(0.015893*M_PI*
            geo.RTrad))+(geo.uAGlen+1.5708*(geo.RThgt+0.257*geo.RTrad))-arc)
                        *0.5e-3,

Tal[0].A=(0.2203*geo.STwid+0.981747*geo.RTrad)
            *geo.ADEP*0.5e-6;

////////////////////////////
//float Len=sqrt(pathB.b*pathB.b+(0.22727*geo.SThgt+geo uAGlen)*
//                        (0 22727*geo.SThgt+geo.uAGlen));

Tal[1].L=((geo.uAGlen+1.0472*geo RThgt)+ellenA)*0.5e-3;
Tal[1].A=(geo.RTrad*0.2618+0.3529*geo.RThgt+0.127118*geo.STwid)
                        *geo.ADEP*0.5e-6,

////
float Len=sqrt((Gap*Gap)+(geo.uAGlen+0.041667*geo.RThgt)*
(geo uAGlen+0.041667*geo RThgt));

Tal[2].L=(Len+ellenA)*0.5e-3;
Tal[2].A=(0.55555*geo RThgt+
                        0.050847*geo.STwid+0.0416667*geo.SThgt)
                        *geo.ADEP*0.5e-6,


pathB a=geo.SThgt/3+geo.uAGlen;
pathB b=0 075*geo.RTwid+Gap;
float ellenB=ellipse(pathB);
Tal[3].L=(Len+ellenB)*0.5e-3,
Tal[3].A=(0.0625*geo.RThgt+0.075*geo RTwid+
                        geo.SThgt/3)
                        *geo.ADEP*0.5e-6;

pathA.a=geo.RTwid*0.2875+Gap;
pathA.b=0.2617994*geo.STrad+geo.uAGlen+geo.RThgt;
ellenA=ellipse(pathA);

Tal[4].L=(ellenA+ellenB)*0.5e-3;
Tal[4].A=(0.2125*geo.RTwid+
                        0.2617994*geo STrad+0 6527778*geo.SThgt)
                        *geo.ADEP*0.5e-6;

Tal[5] L=0;
Tal[5].A=0,

            empty=1;
```

```
}


void pathLA::Utooth3(pathdim geo)
{
//set up ellipse function
axis pathA,pathB;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap left end tooth left
                float Gap=geo.RTrad-0.5*geo.STwid;

                pathA.a=geo.uAGlen+geo.STendT*0.20833;
                pathA.b=Gap+0.3142857*geo.RTwid,
                float ellenA=ellipse(pathA);
                pathB.a=geo.uAGlen+geo.SThgt*0.3;
                pathB.b=Gap+0.0642857*geo RTwid;
                float ellenB=ellipse(pathB);

Tal[0].L=(ellenA+ellenB)*0.5e-3;

Tal[0].A=(0.144*geo.STendT+0.25714*geo.RTwid)
                *geo.ADEP*0.5e-6,

////////////////////////////////
float Len=sqrt((Gap*Gap)+
(geo.STendT/150+geo.uAGlen+0.12*geo.RThgt)*
                                (geo.SThgt/150+geo.uAGlen+0.12*geo.RThgt));

Tal[1].L=(ellenB+Len)*0.5e-3;
Tal[1].A=(geo.RTwid*0.0714+0.2*geo.RThgt+0 05666*geo.STendT)
                                *geo.ADEP*0.5e-6,

////

pathA.a=geo RThgt*1.0625+geo.uAGlen;
pathA.b=0.05714*geo.STwid+Gap;
ellenA=ellipse(pathA);
Tal[2].L=(Len+ellenA)*0.5e-3;
Tal[2].A=(0.833*geo.RThgt+
                                0.08163*geo.STwid+geo.SThgt/150)
                                *geo.ADEP*0.5e-6,


pathB.a=geo.STwid*0.2577+Gap,
pathB.b=0.6333333*geo.RTrad+geo.uAGlen+geo.RThgt;
ellenB=ellipse(pathB);
    float arc=0 187*geo.RTrad;

                Tal[3] L=(ellenA+ellenB-arc)*0.5e-3;
                Tal[3].A=(0.644*geo RTrad+
                                                0.1875*geo.STwid)
                                                *geo.ADEP*0.5e-6;

                Tal[4].L=0;
                Tal[4].A=0,


                Tal[5].L=0;
                Tal[5].A=0;

                                empty=2;

}

void pathLA::Utooth4(pathdim geo)
{
//set up ellipse function
axis pathA,pathB;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap left end tooth  right
                float Gap=geo.RTrad-0.5*geo.STwid;
pathA.b=geo.uAGlen+geo.RThgt*0.74;
pathA.a=Gap+0.0618*geo.STwid;
float ellenA=ellipse(pathA);
//pathB a=geo.uAGlen+geo.SThgt*0.6111;
//pathB.b=Gap+0.15853*geo.RTwid,
//float ellenB=ellipse(pathB);
float arc=0 07767*geo STrad;

Tal[0].L=((geo uAGlen+geo.RThgt)+(geo RTrad-(0 012913*M_PI*
                geo RTrad))+(geo.uAGlen+1.5708*(geo.RThgt+0.3625*geo.RTrad))-arc)
                *0.5e-3;

Tal[0].A=(0.2474*geo.STwid+0.9948*geo.RTrad)
                *geo.ADEP*0.5e-6;

////////////////////////////////
float Len=(geo uAGlen+1.5708*(geo.RThgt+0.3625*geo.RTrad))-arc;

Tal[1].L=(Len+ellenA)*0.5e-3,
```

```
Tal[1].A=(geo.RTrad*0.3553+0.875*geo.RThgt+0.15464*geo.STwid)
                                        *geo.ADEP*0.5e-6;

////
 Len=sqrt((Gap*Gap)+(geo.uAGlen+0 13*geo RThgt)*
                                (geo.uAGlen+0.13*geo.RThgt));

Tal[2] L=(Len+ellenA)*0.5e-3;
Tal[2].A=(0.57*geo.RThgt+
                                0.059829*geo.STwid+0.033333*geo.SThgt)
                                *geo.ADEP*0.5e-6,


pathB a=(0.266*geo.SThgt+geo.uAGlen);
pathB b=(0.07246*geo.RTwid+Gap);
float ellenB=ellipse(pathB);
Tal[3].L=(Len+ellenB)*0.5e-3;
Tal[3].A=(0.2*geo.RThgt+0.072464*geo RTwid+
                                geo.SThgt/3.75)
                                *geo.ADEP*0.5e-6;
                pathA.a=((geo.RTwid*0.26)+Gap)/10;
                pathA.b=((0.0952*geo.STrad)+geo.uAGlen+geo.SThgt)/10;

                ellenA=(ellipse(pathA))*10;

                Tal[4].L=(ellenA+ellenB)*0.5e-3;
                Tal[4].A=(0.203*geo.RTwid+
                                        0.0952*geo.STrad+0.7333*geo.SThgt)
                                        *geo.ADEP*0.5e-6;

                Tal[5] L=0;
                Tal[5].A=0;

                        empty=1;

}

void pathLA::Utooth5(pathdim geo)
{
//set up ellipse function
axis pathA,pathB;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap right end tooth right
                float Gap=geo.RTrad-0.5*geo.STwid;
pathA.a=geo.uAGlen+geo.STendT*0.3,
pathA.b=Gap+0.48*geo RTwid;
float ellenA=ellipse(pathA);
pathB.a=geo uAGlen+geo.STendT*0.11;
pathB.b=Gap+0.146*geo.RTwid;
float ellenB=ellipse(pathB);


Tal[0].L=(ellenA+ellenB)*0.5e-3;

Tal[0].A=(0 1966*geo.STendT+geo.RTwid*0.35227)
                *geo.ADEP*0 5e-6;

////////////////////////////
float Len=sqrt((Gap*Gap)+
(0.017*geo.STendT+geo.uAGlen)*
                                (0.017*geo STendT+geo.uAGlen));

Tal[1].L=(ellenB+Len)*0.5e-3;
Tal[1].A=(geo.RTwid*0.142+0.0776*geo.STendT)
                                *geo.ADEP*0 5e-6;


Tal[2].L=Len*1e-3;
Tal[2].A=(0 45445*geo.RThgt+
                                0 02*geo.STwid+0.0299*geo STendT)
                                *geo.ADEP*0.5e-6;

pathB.a=geo.STwid*0 13+Gap;
pathB.b=0.95*geo RTrad+geo.uAGlen+geo.RThgt;
ellenB=ellipse(pathB);
                float arc=0 024*geo.RTrad;

Tal[3].L=(Len+ellenB-arc)*0.5e-3;
Tal[3].A=(0.10*geo.RTrad+0 636*geo.RThgt+
                                0 112*geo.STwid)
                                *geo.ADEP*0.5e-6;

pathA a=geo.STwid*0.336+Gap;
pathA.b=0.857*geo RTrad+geo.uAGlen+geo.RThgt;
ellenA=ellipse(pathA);

Tal[4].L=(ellenA+ellenB-(0.267*geo.RTrad))*0.5e-3,
Tal[4].A=(0.69*geo RTrad+0.208*geo.STwid)*geo.ADEP*0.5e-6;


Tal[5].L=0;
```

```
                Tal[5].A=0;

                        empty=1;

}

void pathLA::Utooth6(pathdim geo)
{
//set up ellipse function
axis pathA,pathB;


//THIS COMPUTES THE PATH LENGTH AND DISTANCES
//TOOTH air-gap right end tooth  left
                float Gap=geo RTrad-0.5*geo.STwid;
                        float arca=0.148*geo.RTrad;
                pathA.b=geo.uAGlen+geo.RThgt+geo.RTrad*1.19;
                pathA.a=Gap+0.0618*geo.STwid,
                float ellenA=ellipse(pathA);
                pathB.a=geo.uAGlen+geo.RThgt+0.19*geo.RTrad,
pathB.b=Gap+0.16*geo.RTwid;
float ellenB=ellipse(pathB);
float arcb=0.0423*geo.STrad;

Tal[0] L=(ellenA+ellenB-arca-arcb)
                *0.5e-3;

Tal[0].A=(0.2258*geo STwid+1.047*geo.RTrad)
                *geo.ADEP*0.5e-6;

////////////////////////////////
pathA.b=geo.uAGlen+0.5*geo.RThgt;
pathA.a=Gap+0.04*geo.STwid;
 ellenA=ellipse(pathA);
Tal[1].L=(ellenA+ellenB)*0.5e-3;
Tal[1].A=(geo.RTrad*0.18+0.5*geo.RThgt+0.12698*geo.STwid)
                                        *geo.ADEP*0.5e-6;

//// ///
 float Len=sqrt((Gap*Gap)+(geo.uAGlen+0.068*geo.RThgt)*
                                (geo.uAGlen+0.068*geo.RThgt));

Tal[2].L=(Len+ellenA)*0.5e-3;
Tal[2].A=(0.6*geo RThgt+
                                0.0436*geo.STwid+0.06818*geo.SThgt)
                                *geo.ADEP*0.5e-6,

                pathB.a=(0 4318*geo.SThgt+geo uAGlen);
                pathB.b=(0 088*geo.RTwid+Gap);
                ellenB=ellipse(pathB);
                Tal[3].L=(Len+ellenB)*0 5e-3,
                Tal[3].A=(0.0889*geo RTwid+
                                        geo.SThgt/2.75)
                                        *geo.ADEP*0.5e-6;

                pathA.a=((geo.RTwid/3)+Gap)/10;
                pathA.b=((0.42857*geo STrad)+geo.uAGlen+geo SThgt)/10;

                ellenA=(ellipse(pathA))*10;

                        float arc=0.115*geo.STrad;
                Tal[4].L=(ellenA+ellenB-arc)*0.5e-3;
                Tal[4].A=(0.2333*geo.RTwid+
                                        0.49*geo.STrad+0.5454*geo.SThgt)
                                        *geo.ADEP*0.5e-6;

                Tal[5].L=0;
                Tal[5].A=0;

                        empty=1;

}

////////////////////////////////
////////////////////////////////
//function to calculate reluctance
void pathLA::airgapREL() //calculate air gap reluctance
{

double INVtmpR=0;

for (int te=0; te<(Npte-empty); te++)
{
INVtmpR +=(Tal[te].A*MU0)/Tal[te].L; //sum the permeances
                }


AGrel=1/INVtmpR;

}



////////////////////////////////
```

```
////////////////////////////////////////////////////////
/* EVENTUALLY put a check so that STwid>RTwid
ie. if (geo.STwid>geo.RTwid) {
                                                    Mwid=geo.STwid;
                                                    Mhgt=geo.SThgt;
                                                    mwid=geo RTwid
                                                    mhgt=geo.RThgt}
        else {      Mwid=geo.RTwid;
                    Mhgt=geo RThgt;
                    mwid=geo.STwid;
                    mwid=geo SThgt;}

                    ///
                    put geo.STwid and geo.SThgt to be MWid in program and
                    mwid to be RTwid etc.

                     This could be in the end of the dimension def,
                     therefore these commands would have to be geo.Mwid..etc*/
```

```
#include "com.h"
#include "mast.h"
Curve master :hart(double x[DIS],double y[DIS])
{

int COUNT=0;
 //rows columns

// intial guess may do work on intual guess for frolich function!!
Curve RESULT;
int mid=DIS/2; //mid point
double iALP=y[DIS-1]*1.1;//y[no-1]*1e4;  cout<<endl<<"Alpha"<<iALP;
double iBET=x[mid]/2; cout<<endl<<"beta"<<iBET;
double iN=1;
double SumSQ=0:
for (int i=0; i<DIS, i++) {

                SumSQ+=pow(
                y[i]-(
                (iALP*pow(x[i],iN))
                /(iBET+pow(x[i],iN))),2); }


//cout<<endl<<"RSQ= "<<SumSQ<<endl;


double T1[DIS][4];
double T2[3][4];


/////
do {
for (int k=0; k<4,k++){
for (int kk=0; kk<3,kk++)T2[kk][k]=0;}
for (i=0; i<DIS, i++){

                T1[i][0]=pow(x[i],iN)/(iBET+pow(x[i],iN));
                T1[i][1]=(-iALP*pow(x[i],iN))/pow((iBET+pow(x[i],iN)),2);
                T1[i][2]=
                (
                (iALP*pow(x[i],iN)*log(x[i]))/
                (iBET+pow(x[i],iN)))
                -
                (
                (iALP*pow((pow(x[i],iN)),2)*log(x[i]))/
                (pow((iBET+pow(x[i],iN)),2))
                );
                T1[i][3]=y[i]-(
                (iALP*pow(x[i],iN))
                /(iBET+pow(x[i],iN)));
                T2[0][0]+=pow(T1[i][0],2);
                T2[0][1]+=T1[i][0]*T1[i][1];
                T2[0][2]+=T1[i][0]*T1[i][2];
                T2[0][3]+=T1[i][0]*T1[i][3];

                T2[1][0]+=T1[i][0]*T1[i][1];
                T2[1][1]+=pow(T1[i][1],2);
                T2[1][2]+=T1[i][1]*T1[i][2];
                T2[1][3]+=T1[i][1]*T1[i][3];

                T2[2][0]+=T1[i][0]*T1[i][2];
                T2[2][1]+=T1[i][2]*T1[i][1];
                T2[2][2]+=T1[i][2]*T1[i][2];
                T2[2][3]+=T1[i][2]*T1[i][3];
                }

// solve systems of equations for D.
double EQ1[4], EQ2[4];
for ( i=0; i<4; i++) {
                EQ1[i]=(T2[1][0]/T2[0][0])*T2[0][i];
                EQ2[i]=T2[1][i]-EQ1[i];}

double EQ3[4];
for (i=0; i<4; i++) {
                EQ1[i]=(T2[2][0]/T2[0][0])*T2[0][i];
                EQ3[i]=T2[2][i]-EQ1[i];}

double EQ4[4],
for ( i=0; i<4; i++) {
                EQ1[i]=(EQ3[1]/EQ2[1])*EQ2[i];
                EQ4[i]=EQ3[i]-EQ1[i];}
double D[3];
D[2]=EQ4[3]/EQ4[2];
D[1]=(EQ2[3]-(D[2]*EQ2[2]))/EQ2[1];
D[0]=(T2[0][3]-(D[2]*T2[0][2])-(D[1]*T2[0][1]))/T2[0][0];
//
//minimum value of Q
double alp1, bet1, n1, alp05, bet05, n05;
alp1=iALP+D[0]; bet1=iBET+D[1]; n1=iN+D[2];
alp05=iALP+0 5*D[0]; bet05=iBET+0.5*D[1]; n05=iN+0.5*D[2];
double Q0=0,double Q05=0; double Q1=0,
for (i=0; i<DIS,i++) {
                Q0+=T1[i][3]*T1[i][3];
                Q1+=pow((y[i]-((alp1*pow(x[i],n1))/(bet1+pow(x[i],n1)))),2);
                Q05+=pow(y[i]-((alp05*pow(x[i],n05))/(bet05+pow(x[i],n05))),2);
                }

double Vmin=0.5+(0.25*(Q0-Q1)/(Q1-2*Q05+Q0));
```

```
double NEWALP=iALP+Vmin*D[0];
double NEWBET=iBET+Vmin*D[1]; double NEWN=iN+Vmin*D[2];

//cout<<endl<<NEWALP<<" (C) "<<NEWBET<<" A B N "<<NEWN;

//sum of squares of residuals

SumSQ=0,
for (i=0; i<DIS; i++) {

                SumSQ+=pow(
                y[i]-(
                (NEWALP*pow(x[i],NEWN))
                /(NEWBET+pow(x[i],NEWN))),2); }


//cout<<endl<<"RSQ= "<<SumSQ;

//find convergance percentage

double Ealp=((NEWALP-iALP)/NEWALP)*100;  Ealp=fabs(Ealp);
double Ebet=((NEWBET-iBET)/NEWBET)*100;  Ebet=fabs(Ebet);
double En=((NEWN-iN)/NEWN)*100,  En=fabs(En);
//cout<<endl<<Ealp<<" converge (c) "<<Ebet<<"  "<<En;

// converge check
if (Ealp<1e-5 && Ebet<1e-5 && En<1e-5) COUNT=55;

iALP=NEWALP; iBET=NEWBET, iN=NEWN;COUNT++;
RESULT.a=NEWALP,RESULT.b=NEWBET,RESULT.n=NEWN;
}while (COUNT<50);
return RESULT,
}
```

```
#include"com.h"
//linklist

double LINKlist::PLOT(int position)
{
LINK* current=first;
double value;
for (int i=0; i<position, i++)
{
current=current->next;
}
value=current->data;
return value;
}


void linklist::additem(FluxLink d)
{
link* newlink= new link;
newlink->data=d,
newlink->next=first;
first=newlink;
}

void LINKlist::toradd(double t)
{
LINK* newlink= new LINK;
newlink->data=t,
newlink->next=first;
first=newlink;
}



double linklist::detent()
{
link* current=first;
double mmf=current->data.mag;
return mmf;
}

double linklist::coenergy(int NUTP,double cINC,double offset, double Mx)
{

double total=0;
double b=0;
double a,fa,fb;
int pos=0;
link* current=first;
while(current !=NULL && b!=Mx)
                {
                a=pos*(cINC/2);
                fa=((current->data.al)-(current->data un))*NUTP+offset;
                current=current->next;
                pos++;
                b=pos*(cINC/2);
                fb=((current->data.al)-(current->data.un))*NUTP+offset;
                total+=(b-a)*(fa+fb)/2.
                }


return total;
}
////////////////////////////////////////
void LINKlist::finaltorque(LINKlist A,LINKlist B)
{
double total,
double keep=0;
LINK* currentA=A first;
LINK* currentB=B.first;
while(currentA !=NULL || currentB !=NULL )
                {
                total=100*(currentA->data-currentB->data)/M_PI;
                keep+=total;


                LINK* newlink=new LINK;
                newlink->data=keep;
                newlink->next=first;
                first=newlink;

                currentA=currentA->next;
                currentB=currentB->next;
                }
}
////////////////////////////////
```

```
double flux(Curve, double); //function to calculate flux
double flux(Curve equ,double MMF)
{
double fluxi=(equ.a*pow(MMF,equ.n))/
                                        (equ.b+pow(MMF,equ.n));
return fluxi;
}

double dflux(Curve, double),
double dflux(Curve equ,double MMF)
{
double dfluxi=((equ.n*equ.a*pow(MMF,equ.n))/
                                        (MMF*(equ.b+pow(MMF,equ.n))))
                                        -
                                        ((equ.n*equ.a*pow(pow(MMF,equ.n),2))/
                                        (MMF*pow(equ.b+pow(MMF,equ.n),2)));
return dfluxi;
}
```

```
double coENG
(float width,int turns. double AL1.double UN1,double AL2,double UN2,
int neg,double offset)
{
int a=1;int b=0;
if (neg==1){ a=-1; b=1;}
double AREA=(0.5*width*turns*a*((AL1-UN1+offset)+(AL2-UN2+offset)))-
                                                width*turns*b*offset;

return AREA;
}

/////function start///
FluxLink get(Curve McurveBIMAG,Curve MalFMe,Curve MmidFMe, Curve MunFMe,
double MbimagMMF,Curve magnetequ,double lenMAG, double areaMAG,double Ho,
float current, int turns,int pos)
{

double coil=turns*current/4;
double coilb;
double coilc1;
double coilc2;
double coild,

int flagb=0;
int flagc1=0;
int flagc2=0;
int flagd=0;


int count=0;
double mmf=200;
double mmfn;
double con,
do{
//BRANCH A (magnet and back-iron relationship)
double MMFcorr=MbimagMMF+(-1*mmf);
double fluxMAG=flux(McurveBIMAG,MMFcorr);

double dfluxMAG=dflux(McurveBIMAG,MMFcorr),

//BRANCH B (so called aligned branch)  //MalFM.equ

if (pos==0 || pos==2) coilb=coil; else coilb=0;
if ((mmf+coilb)<0) flagb=1; else flagb=0;
double mmfb=fabs(mmf+coilb);
double fluxb=flux(MalFMe,mmfb);
if (flagb==1) fluxb=fluxb*-1;
double dfluxb=dflux(MalFMe,mmfb);


//BRANCH C1 (so called mid branch 1)  //MmidFM.equ

if (pos==1 || pos==2 ) coilc1=coil; else coilc1=0;
if ((mmf+coilc1)<0) flagc1=1; else flagc1=0,
double mmfc1=fabs(mmf+coilc1);
double fluxc1=flux(MmidFMe,mmfc1);
if (flagc1==1) fluxc1=fluxc1*-1;
double dfluxc1=dflux(MmidFMe,mmfc1);

//BRANCH C2 (so called mid branch_2)  //MmidFM.equ

if (pos==1 || pos==2) coilc2=-1*coil; else coilc2=0;
double mmfc2=fabs(mmf+coilc2);
double fluxc2=flux(MmidFMe,mmfc2);
if (flagc2==1) fluxc2=fluxc2*-1;
double dfluxc2=dflux(MmidFMe,mmfc2);

//BRANCH D (so called un branch)  //MunFM.equ

if (pos==0 || pos==2) coild=coil, else coild=0;
if ((mmf-coild)<0) flagd=1; else flagd=0;
double mmfd=fabs(mmf-coild);
double fluxd=flux(MunFMe,mmfd);
```

```
void pathLA::eqnewflux(FluxMmf Fn[DIS], double addflux[DIS])
{
int inc=0;
double bh[5];
bh[0]=alp
bh[1]=beta
bh[2]=n
bh[3]=e
bh[4]=k

do {
                Hi=Fn[inc].m;

                addflux[inc] +=(((bh[0]*pow(Hi,bh[2])+bh[4]*pow(Hi,bh[2]+1))/
                    (bh[1]+pow(Hi,bh[2])+bh[3]*pow(Hi,bh[2]+1)));
                inc++;
}while (inc<DIS),
}
```

```
int count=0; int errflag=0;
double con=1;
do{
//BRANCH A (magnet and back-iron relationship)
double MMFcorr=MbumagMMF+(-1*mmfl);
double fluxMAGl=flux(McurveBIMAG,MMFcorr);
 MMFcorr=MbumagMMF+(-1*mmfu);
double fluxMAGu=flux(McurveBIMAG,MMFcorr);

double fluxbl=flux(MalFMe,mmfl);double fluxbu=flux(MalFMe,mmfu);
double fluxcl=flux(MmidFMe,mmfl),double fluxcu=flux(MmidFMe,mmfu);
double fluxdl=flux(MunFMe,mmfl),double fluxdu=flux(MunFMe,mmfu);

double A0l=(fluxbl+2*fluxcl+fluxdl)/4.
double A1l=0.5*(fluxbl-fluxdl);
double A2l=0.25*(fluxbl-fluxcl+fluxdl-fluxcl);

float ang1=0 9;
float ang2=2.7;
double Fl09=A0l+A1l*cos(2*M_PI*ang1/7.2)+A2l*cos(4*M_PI*ang1/7.2);
double Fl27=A0l+A1l*cos(2*M_PI*ang2/7.2)+A2l*cos(4*M_PI*ang2/7.2);
double A0u=(fluxbu+2*fluxcu+fluxdu)/4.
double A1u=0.5*(fluxbu-fluxdu);
double A2u=0.25*(fluxbu-fluxcu+fluxdu-fluxcu);

double Fu09=A0u+A1u*cos(2*M_PI*ang1/7.2)+A2u*cos(4*M_PI*ang1/7.2);
double Fu27=A0u+A1u*cos(2*M_PI*ang2/7.2)+A2u*cos(4*M_PI*ang2/7.2);

//chk to see with in range;

double Fl=4*Fl09+4*Fl27-fluxMAGl.
double Fu=4*Fu09+4*Fu27-fluxMAGu.

if (errflag==1){
        if(Fl*Fu<0) errflag=0; else mmfu+=50.}

if (errflag!=1){
mmfR=mmfu-(Fu*(mmfl-mmfu))/(Fl-Fu).
MMFcorr=MbumagMMF+(-1*mmfR);
double fluxMAGR=flux(McurveBIMAG,MMFcorr);


fluxbR=flux(MalFMe,mmfR);
double fluxcR=flux(MmidFMe,mmfR);
 fluxdR=flux(MunFMe,mmfR);

double A0R=(fluxbR+2*fluxcR+fluxdR)/4.
double A1R=0 5*(fluxbR-fluxdR);
double A2R=0.25*(fluxbR-fluxcR+fluxdR-fluxcR);
fluxR09=A0R+A1R*cos(2*M_PI*ang1/7.2)+A2R*cos(4*M_PI*ang1/7.2);
fluxR27=A0R+A1R*cos(2*M_PI*ang2/7.2)+A2R*cos(4*M_PI*ang2/7.2);

double FR=4*fluxR09+4*fluxR27-fluxMAGR.

double conM.
if (Fl*FR<0) {conM=mmfu; mmfu=mmfR.} else {conM=mmfl; mmfl=mmfR.}

con=fabs((mmfR-conM)/mmfR)*100.

}
count++;
}
while (count<100 && con>1e-5);

FluxLink detent;
detent.al=fluxbR;
detent.un=fluxdR.
detent mid1=fluxR09;
detent mid2=fluxR27;
detent mag=mmfR;

return detent;
}
```

```
if (flagd==1) fluxd=fluxd*-1;
double dfluxd=dflux(MunFMe,mmfd);


///////
double y=2*fluxb+2*fluxc1+2*fluxc2+2*fluxd-fluxMAG;
double dy=2*dfluxb+2*dfluxc1+2*dfluxc2+2*dfluxd-dfluxMAG;
mmfn=mmf-(y/dy);
con=100*fabs((mmfn-mmf)/mmfn);
mmf=mmfn,
count++;
}while (count<100 && con>1e-5);


//find magnet mmf //not important
//flux flowing through magnet
//double MMFcorr=MbimagMMF+(-1*mmf);
//double fluxMAG=flux(McurveBIMAG,MMFcorr);
//count=0;
//double Bmag=fluxMAG/(areaMAG*1e-6); //flux through mag

                //converted to flux density
//double iguessH=(2*mmf/(1e-3*lenMAG))+Ho; //H guess using mmf calculated

                                        //with backiron

//double magHn;
//do {

//double B=flux(magnetequ,iguessH);
//double dB=dflux(magnetequ,iguessH);

//B=B-Bmag;
//magHn=(iguessH-(B/dB));
//con=100*fabs((magHn-iguessH)/magHn);
//iguessH=magHn;
//}while(count<100 && con>1e-5);

FluxLink details;
//details.mag=(magHn-Ho)*lenMAG*1e-3;
details.mag=mmf;
////////// get flux values/////
//BRANCH B (so called aligned branch)  //MalFM equ

if (pos==0 || pos==2) coilb=coil, else coilb=0;
if ((mmf+coilb)<0) flagb=1; else flagb=0;
double mmfb=fabs(mmf+coilb);
double fluxb=flux(MalFMe,mmfb);
if (flagb==1) fluxb=fluxb*-1;

if (pos==0 || pos==2) details.al=fluxb,else details.mid1=fluxb;
//////////////////////////
//BRANCH C1 (so called mid branch 1)  //MmidFM.equ

if (pos==1 || pos==2) coilc1=coil, else coilc1=0;
if ((mmf+coilc1)<0) flagc1=1; else flagc1=0;

double mmfc1=fabs(mmf+coilc1);
double fluxc1=flux(MmidFMe,mmfc1);
if (flagc1==1) fluxc1=fluxc1*-1;
if (pos==1) details.al=fluxc1;else details.mid1=fluxc1;


//BRANCH C2 (so called mid branch 2)  //MmidFM.equ

if (pos==1 || pos==2) coilc2=-1*coil; else coilc2=0;
double mmfc2=fabs(mmf+coilc2);
double fluxc2=flux(MmidFMe,mmfc2);
if (flagc2==1) fluxc2=fluxc2*-1;
if (pos==1) details.un=fluxc2,else details.mid2=fluxc2;
//////////////////////////
//BRANCH D (so called un branch)  //MunFM equ

if (pos==0 || pos==2) coild=coil; else coild=0;
if ((mmf-coild)<0) flagd=1; else flagd=0;
double mmfd=fabs(mmf-coild);
double fluxd=flux(MunFMe,mmfd);
if (flagd==1) fluxd=fluxd*-1;
if (pos==0 || pos==2) details.un=fluxd;else details.mid2=fluxd;

return details;
}


//////////////////
FluxLink getDENT(Curve McurveBIMAG,Curve MalFMe,Curve MmidFMe, Curve MunFMe,
double MbarnagMMF)
{
double mmfR;
double mmfl=10;
double mmfu=350;


double fluxbR,fluxdR,fluxR09,fluxR27;
```

APPENDIX D

Data sheets for soft magnetic composite materials used in chapter 7.

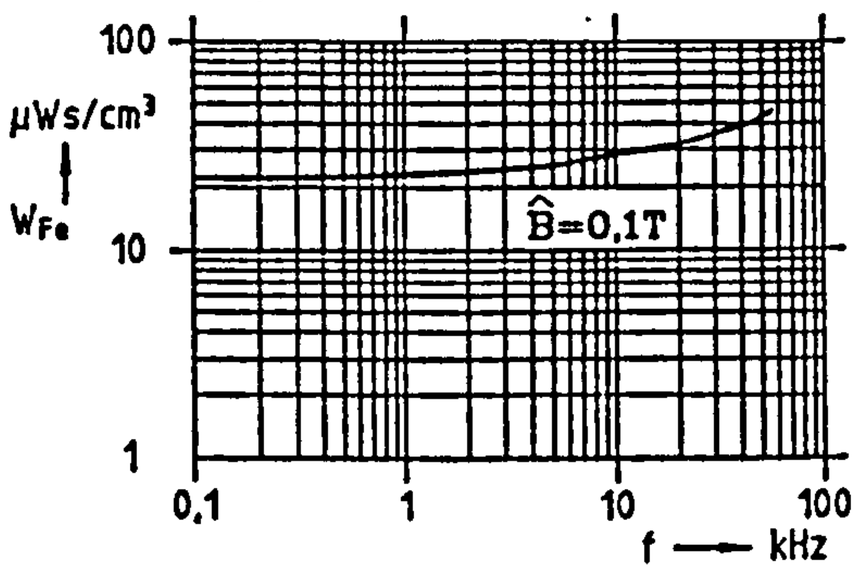
SINTERMETALLE PROMETHEUS
Ottostraße 4 · D-75676 Graben-Neudorf
Telefon (07255) 5077 · Telefax (07255) 3077 · Teletex 725510

DATA SHEET

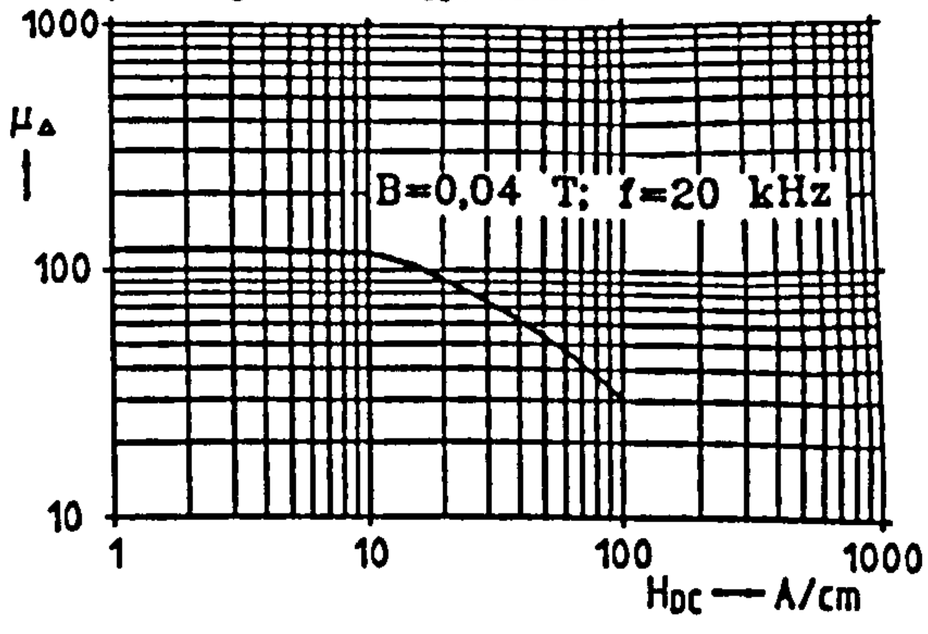
011.72.189.0

Powder Composite Material-1172

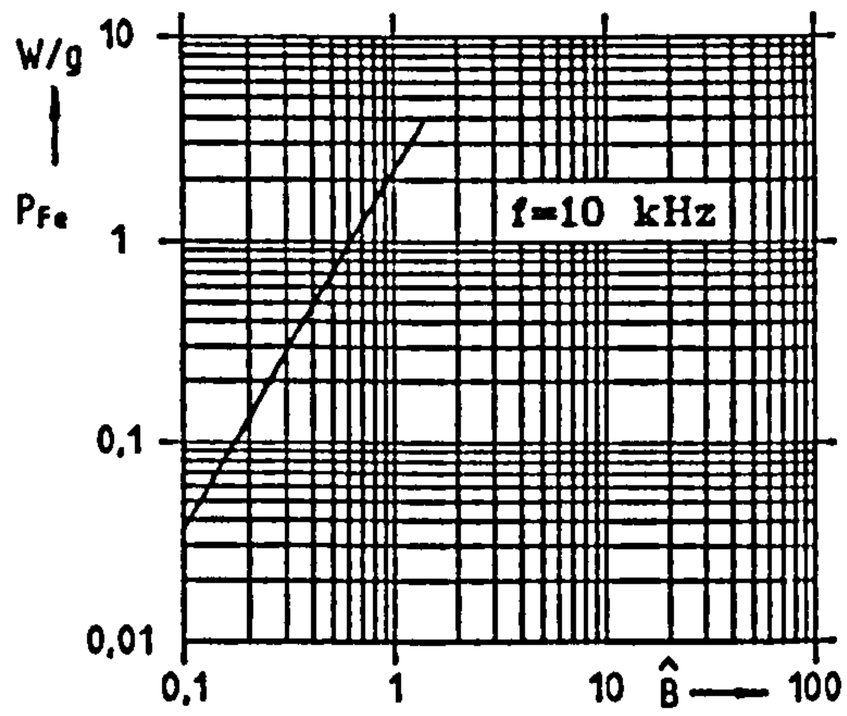
Specific magnetization losses W_{Fe} per cycle corresponding to the frequency



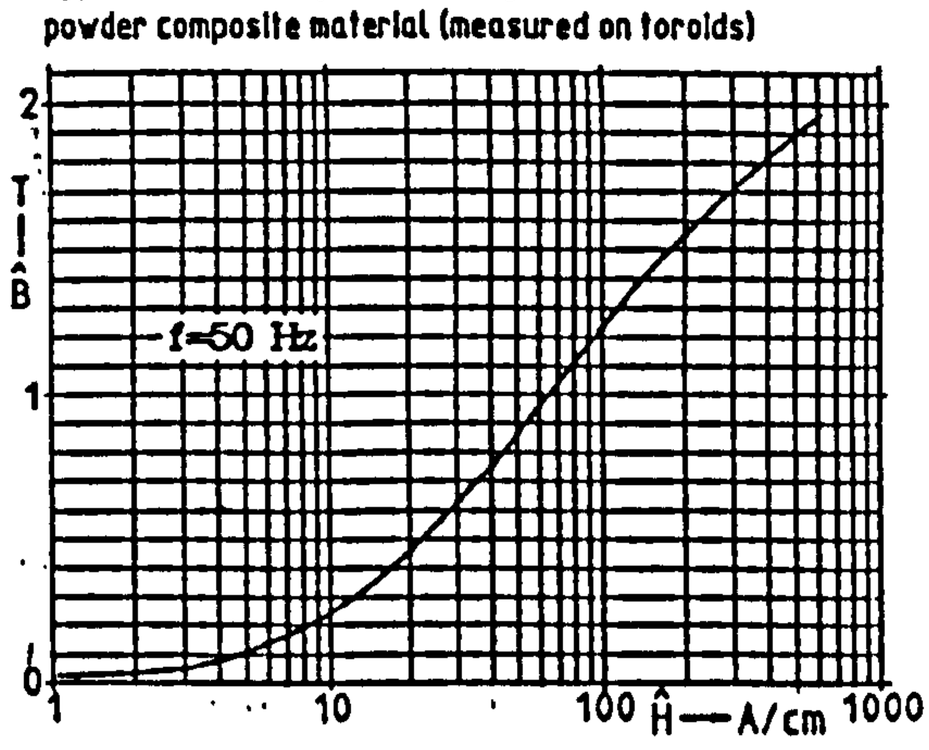
Incremantal permeability μ_Δ as a function of pre-magnetization H_{DC} (measured on toroids)



Losses versus modulation (measured on toroids)



Typical flux density/field strenght curves of powder composite material (measured on toroids)



physical properties		magnetic properties	
Density	7,37 g/cm³	Permeability (40mT)	115
Electrical resistivity	0,5-10⁴ Ωcm	Max. permeability	190
Young·s modulus	40-70 kN/mm²	Saturation flux density	1,90 T
Compression strenght	240 N/mm²	Coercivity	4,10 A/cm
Bending strenght	50-100 N/mm²	Specific core losses (0,1 T; 1 kHz)	23 μWs/cm³
Coeff. of linear expansion (20° bis 100°C)	12-14 10⁻⁶/K	Core losses (1 T;50 Hz)	60 mW/cm³
Thermal conductivity	6-15 W/m*K		
Permessible operating temperature	max. 150 °C		

Data sheets of other materials upon request !

CN

ABM100.32

ABM100.32 is an insulated iron powder developed for soft magnetic applications for frequencies < 10 kHz. The powders high compressibility and durable insulation gives high induction and low losses in combination with constant permeability over a broad frequency range.

The special insulation also allows annealing which gives very low losses at frequencies < 1000Hz. The distributed air gap in the material combined with the cost effective and three dimensional shape making capability of P/M technology offers flexibility for designers.

Mixes

1. ABM 100.32 + 0,5% Kenolube + 0.5% Phenolic Resin
2. ABM 100.32 + 0.5% Kenolube

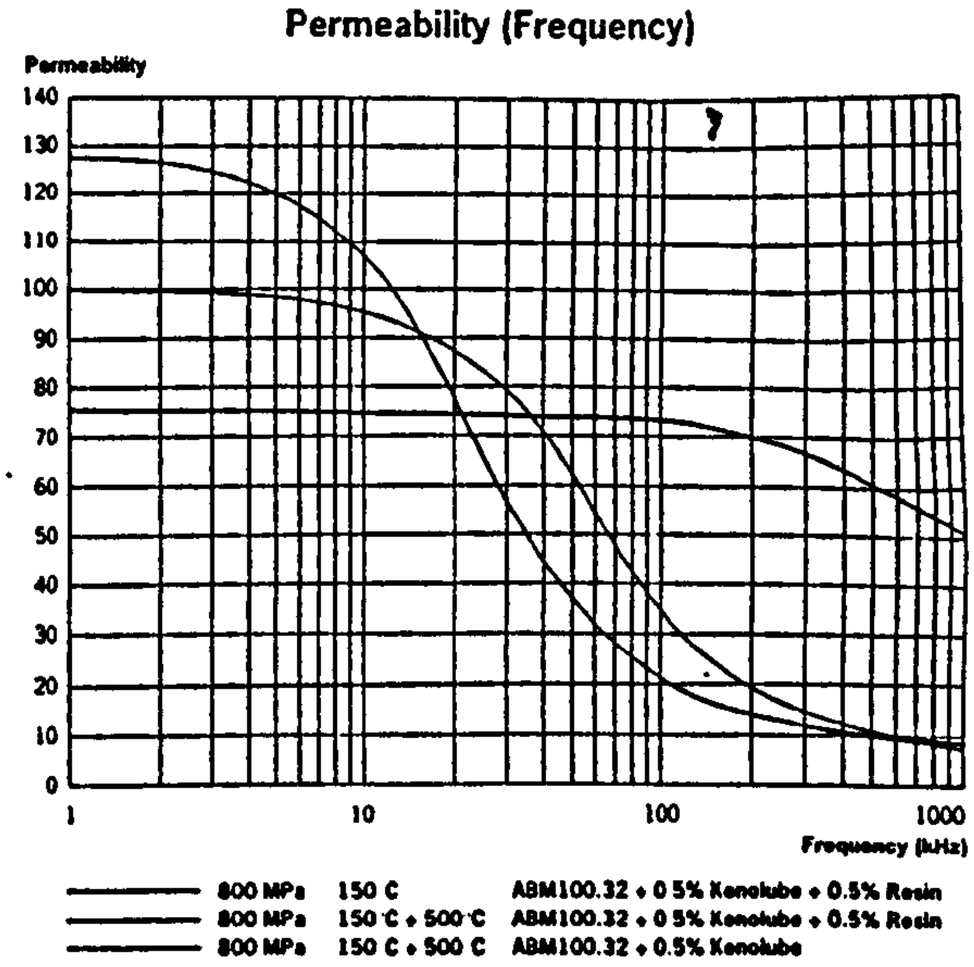
Typical data

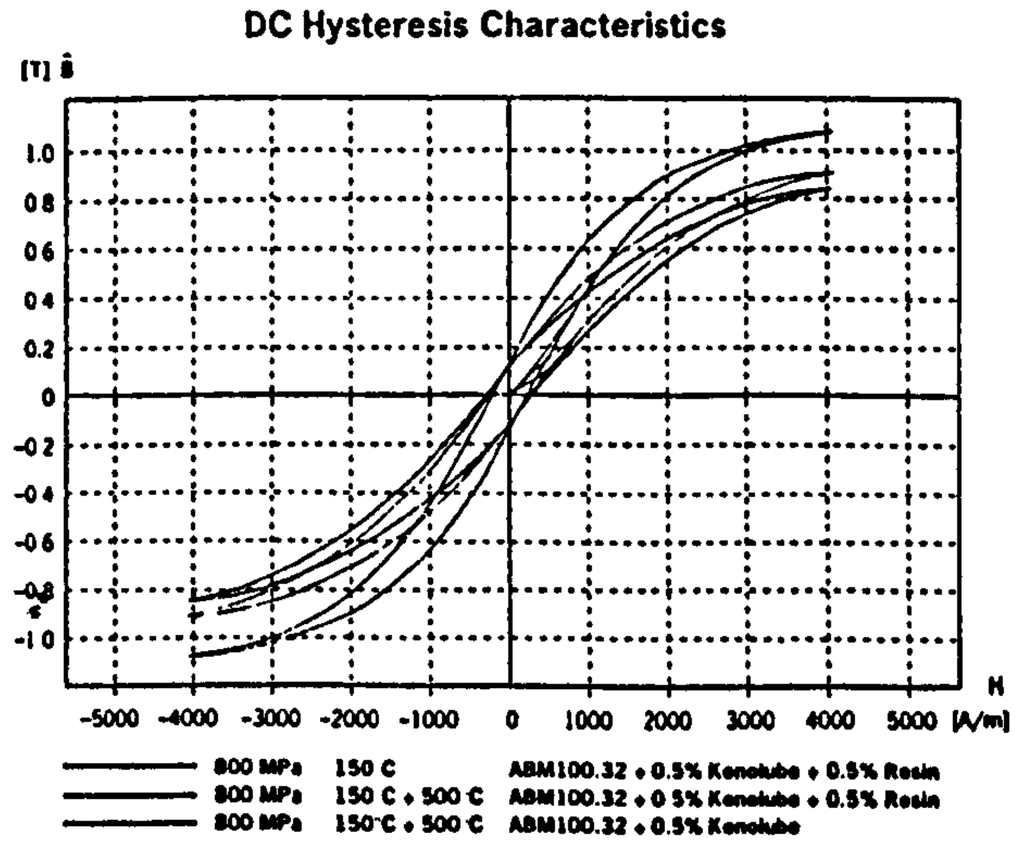
Apparent density	3.2 g/cm³
Flow	22.6 s/50g

	Mix 1	Mix 2
GD(800 MPa), g/cm³	7.18	7.34
GD(600 MPa), g/cm³	7.09	7.21

Heat treatment

Curing	150°C, 1 h in air		150°C, 1 h in air +	
Annealing			500°C, 0.5 h in air	
	Mix 1	Mix 2	Mix 1	Mix 2
Strength (TRS), N/mm²	27	20	63	31
Dimensional change, %	-0.025	-0.002	0.03	0.015
μ_{max} 800 MPa	225		290	390
600 MPa			265	365





Höganäs

The Difference is Knowledge

Manufacturing conditions

Powder

ABM100.32 is an insulated sub 150 micron powder.

Manufacturing

The test toroids were manufactured from mixes of ABM100.32, lubricant and phenolic resin as indicated.
Compacting pressure; 600 MPa and 800 MPa.

Heat Treatment

Curing

All test specimens were cured at 150°C for 1 hour in air.

Annealing

After curing an annealing was performed at 500°C for 30 minutes in air.

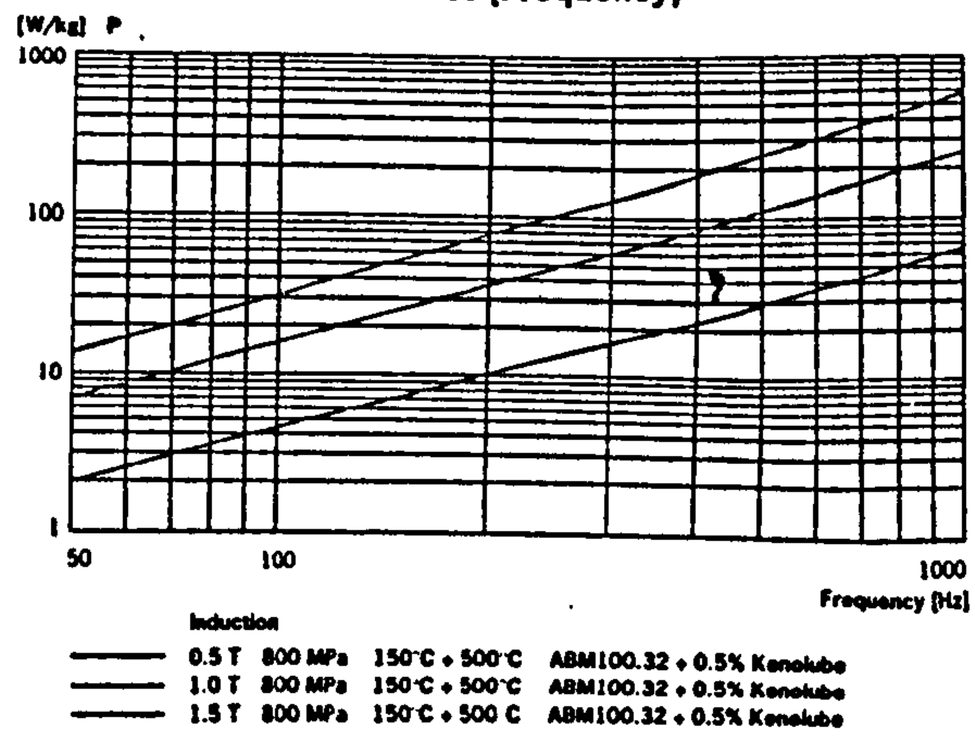
Testing conditions

Physical properties were determined in accordance with applicable ISO standards.

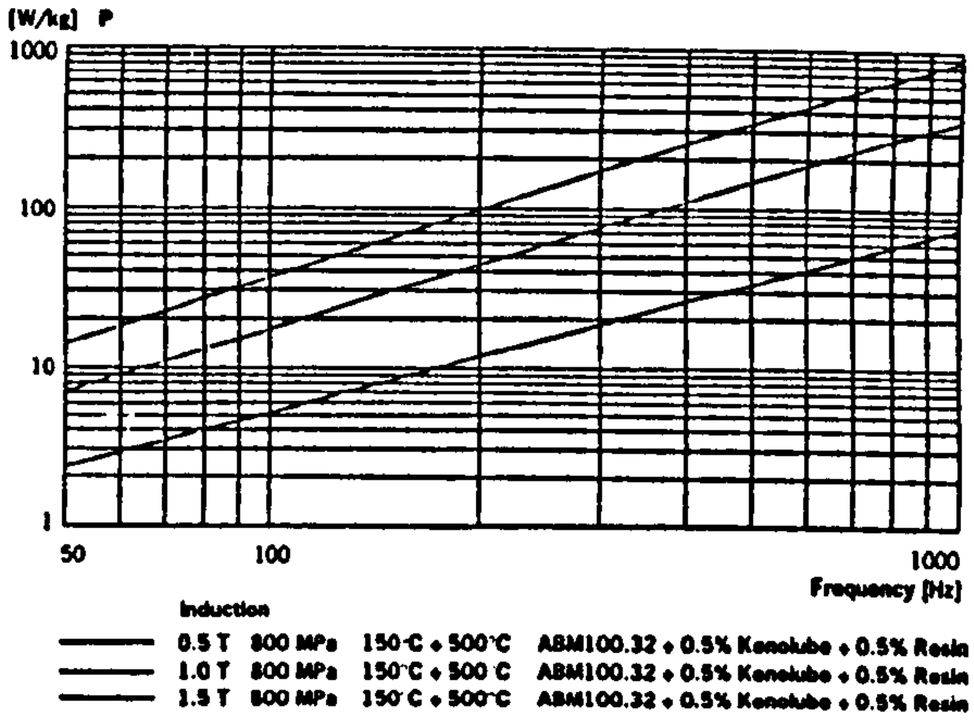
Magnetic properties were measured on toroids with following dimensions

Outer diameter	55 mm
Inner diameter	45 mm
Height	5 mm

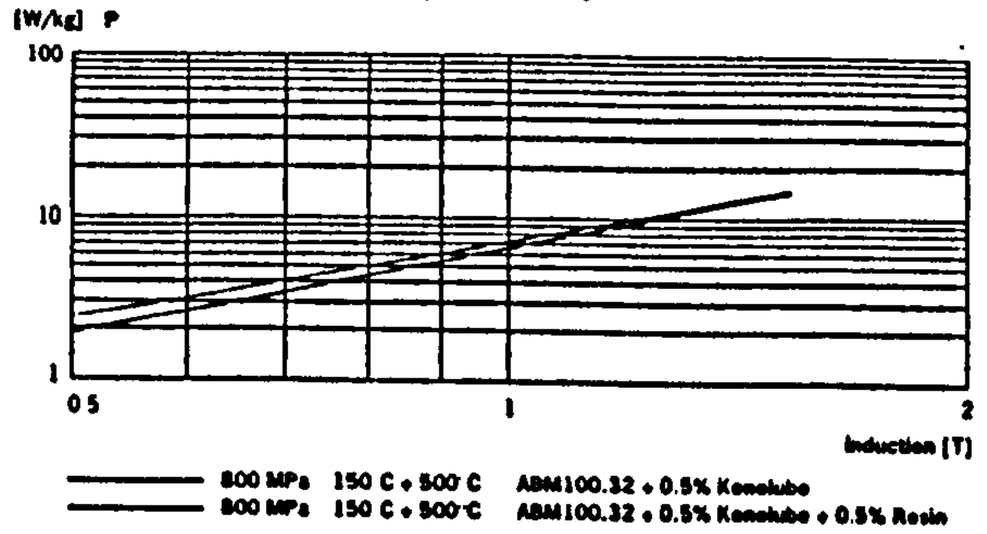
Core Loss (Frequency)



Core Loss (Frequency)



Core Loss (Induction) at 50 Hz



Höganäs

The Difference is Knowledge

Höganäs AB, S-263 83 Höganäs, Sweden
Tel. +46-42 338000, Telex 72368, Fax +46-42 338150