56664

OPTIMUM GEOMETRY FOR TORQUE RIPPLE MINIMIZATION OF SWITCHED RELUCTANCE MOTORS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

FUNDA ŞAHİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

JANUARY 1996

Tez
Dokümantasyon

Approval of the Graduate School of Natural and Applied Sciences

(or) Prof. Dr. İsmail Tosun
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Fatih Canatan
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. H. Bülent Ertan
Co-Supervisor

Assoc. Prof. Dr. Kemal Leblebicioğlu
Supervisor

Examining Comittee Members

Prof. Dr. Muammer Ermiş (Chairman)

Prof. Dr. H. Bülent Ertan

Prof. Dr Yıldırım Üçtuğ

Assoc. Prof. Dr. Kemal Leblebicioğlu

Assoc. Prof. Dr. Müjdat Tohumcu

ABSTRACT

OPTIMUM GEOMETRY FOR TORQUE RIPPLE MINIMIZATION OF SWITCHED RELUCTANCE MOTORS

Şahin, Funda

M.S., Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Kemal Leblebicioğlu

Co-Supervisor: Prof. Dr. H. Bülent Ertan

January 1996, 207 pages

For reducing torque ripple of SR motors two different approaches can be considered. One is to pursue a motor geometry which reduces torque ripple, the second is to manipulate motor current to improve performance. The second approach to the problem requires fast and costly microprocessors for control purposes. A number of papers reporting progress in this area exist. In any case, it is desirable to look for an ideal geometry even if following such an approach, further improvement is desireable.

Determination of optimum geometry that minimizes torque ripple is extremely difficult and requires the abililty to predict SR motor performance for all magnetic circuit parameters for a specified operating condition. Partly because of this reason, research in the literature concentrates on minimizing torque ripple for a specific

geometry through shaping of the phase current. The research reported in this thesis however presents a more general approach to the problem and inspects the effect of magnetic circuit parameters on the torque ripple while they are allowed to vary in a wide range. This is made possible since the authors have access to a set of dimensionless data, and a method which permits prediction of torque position and permeance position characteristics of a doubly-salient magnetic structure as a function of exciting MMF for a specified tooth width/tooth pitch (t/λ), tooth pitch/airgap length (λ/g) and normalized position (x_n).

In this study, a doubly salient geometry which minimizes torque ripple at low speeds is sought. To simplify the problem at this stage the excitation is assumed to be available without any restriction due to winding space. The tools available for this purpose are a finite element field solution program and a set of normalized data as described above. First the accuracy of the finite element program is tested. It is found that although accurate results can be obtained, this approach is very lengthy for searching wide range of parameters.

Than using the force and permeance data for the same purpose is investigated. A neural network based software is developed to extract the desired information from the data available. Than the problem of determining the optimum geometry is treated as a mathematical optimization problem. The results indicated that irrespective of the excitation level, the optimum points occur for large λ/g values (about 250) and for t_s/λ (stator tooth width/tooth pitch) about 0.4 and t_r/λ (rotor tooth width/tooth pitch) about 0.5.

To gain a physical insight to the problem various parameters are fixed and variation of torque ripple with one independent variable is also investigated. The results are presented in a table which identifies a range for each parameter where torque ripple is quite near the global minimum.

Finally neural network based predictions are compared with finite element field solution results. It is observed that, the findings from the two approaches agree quite well.

In this work, it is also shown that, when a single phase is excited, adjacent pairs of teeth to the excited pole play no role in torque production.

Key Words: Reluctance motors, torque ripple, optimum design,
finite element analysis, artificial neural networks.

ÖZ

ANAHTARLI RELÜKTANS MOTORLARINDA MOMENT DALGACIKLARINI ENAZLAMA AMAÇLI TASARIM

Şahin, Funda

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Kemal Leblebicioğlu

Ortak Tez Yöneticisi: Prof. Dr. H. Bülent Ertan

Ocak 1996, 207 sayfa

Anahtarlı relüktans (AR) motorlarında moment dalgacıklarının azaltılması amaçlı iki ayrı yaklaşım bulunmaktadır. İlki, moment dalgacıklarını azaltacak bir motor geometrisi tasarımı, ikincisi ise motor akımına bu dalgacıkları azaltacak şekilde müdahale edilmesidir. İkinci yaklaşım, ki bu konuda önemli düzeyde gelişmeler kaydedilmiştir, denetim amacına hizmet edecek hızlı ve pahalı mikroişlemcilere ihtiyaç duymaktadır. Her ne kadar bu ikinci yaklaşımla ulaşılabilecek çözümlerin maliyet problemlerinin giderilmesi için yapılacak olan çalışmalar bu konuyla ilgilenenlerce daha fazla ilgi görmekteyse de, ideal geometrinin tespiti hem

meselenin özünün anlaşılması, hem de yukarıda değinilen çalışmalara destek olarak önem arzetmektedir.

Moment dalgacıklarının enazlanmasını sağlayacak ideal geometrinin tespiti oldukça güç bir süreç olup, tanımlı çalışma koşullarında, tüm manyetik devre parametreleri için AR motor performansının kestirilmesini gerektirir. Bu çalışmada elde bulunan simetrik çıkık kutuplu geometriler için hazırlanmış boyuttan bağımsız veriler ve çifte çıkık kutuplu manyetik devrelerde moment-konum ve permeans-konum karakteristiklerini, uyarma MMK'nın (magneto motor kuvveti) diş genişliği/diş açıklıgı (t/λ) ve diş açıklığı/hava aralığı uzunluğu (λ/g) için tanımlı bir fonksiyonu olarak ifade eden bir metod kullanılmıştır. Böylece moment dalgacıklarını enazlayacak bir geometrinin tespiti mümkün olabilmiştir.

Bu tez düşük hızlarda AR motoru için moment dalgacıklarını enazlamayı amaçlayan bir çalışmayı kapsamaktadır. Problemin basitleştirilmesi açısından sargıların kapladığı alandan kaynaklanan fiziksel kısıtlar ihmal edilmiştir. Bu çalışmada AR motorunun moment eğrilerinin elde edilmesinde, elde bulunan bir sonlu elemanlar analiz programı (ANSYS) ve yukarıda değinilen normalize edilmiş veri araç olarak kullanılabilmektedir. İlk olarak sonlu elemanlar analiz programı doğruluğu açısından test edilmiş ve büyük ölçüde hassasiyet elde edilmiştir. Ancak, bir optimizasyon sürecinde geniş bir parametre aralığında çalışma yapılacağından, zaman kısıtı bu yaklaşımın kullanılmasını olanaksız kılmıştır.

Sonuç olarak elde bulunan moment ve permeans verisinin araştırma için kullanılmasına karar verilmiştir. İhtiyaç duyulan ara veriler, bir yapay sinir ağının eldeki verilerle eğitilmesiyle belirlenmektedir. Problemin doğrusal olmayan yapısının matematiksel modelleme imkanlarını ciddi bir şekilde sınırlaması, yapay sinir ağlarının kullanılmasını gerekli kılmıştır. Daha sonra, matematiksel bir optimizasyon problemi modellenmiştir. En iyi sonuçların çeşitli uyartım

seviyelerinde, λ/g'nın yüksek (yaklaşık 250), t_s/λ'nın (stator diş genişliği/diş açıklığı) 0.4 ve t_r/λ'nın (rotor diş genişliği/diş açıklığı) 0.5 civarında olduğu aralıklarda bulunduğu tespit edilmiştir.

Probleme fiziksel bir anlam getirebilmek amacıyla parametrelerin değişen değerlerinde moment dalgacık miktarının nasıl değiştiği araştırılmıştır ve parametrelerin çeşitli kombinasyonları için hangi aralıklarda dalgacık miktarının düşük olduğunu gösteren tablolar düzenlenmiştir.

Son olarak sinir ağlarıyla elde edilen moment tahminleri sonlu elemanlar analizi sonuçlarıyla karşılaştırılmış ve iki yöntemin yaklaşık sonuçlar verdiği görülmüştür. Buna ek olarak, anahtarlı relüktans motorlarında tek faz uyartıldığında komşu diş çiftlerinin moment üretimine etkisinin olmadığı gösterilmiştir.

Anahtar Sözcükler: Relüktans motorları, moment dalgacıkları, optimum tasarım sonlu elemanlar analizi, yapay sinir ağları

ACKNOWLEDGMENTS

I express sincere appreciation to Prof. Dr. H. Bülent Ertan and Assoc. Prof. Dr. Kemal Leblebicioğlu for their guidance and insight throughout the research. The technical assistance of Hüsnü Yıldız and Hamiz Erdem is gratefully acknowledged. I offer sincere thanks to Balkan Şimşir, Bülent Özhorasan and Ayşe Kabasakal for their amity. To my parents, I thank them for their support. And special thanks to my husband Önder, for grammatical corrections and his desirous endurance to switched reluctance motors.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 General

Switched-reluctance motors (srm) have gained attention in the variable speed drive market. The savings in motor manufacturing cost due to the simplicity of construction and usage of minimum number of switching devices in the drive circuit are two important factors in its favor compared to any other motor drive. Moreover, an attractive feature of srm is its higher torque-inertia ratio compared to similarly sized induction motors or permanent magnet synchronous motors. The advantages and disadvantages of the motor are listed below:

Advantages:

a) High torque to inertia ratio,

b) High speed operation,

c) Fast step response,

f) Mechanically simple with long life,

g) Bidirectional rotation,

Disadvantages:

a) No holding torque when windings are not excited,

b) Oscillations and overshoot in step response,

c) Acoustic noise and speed ripples caused by high torque ripple.

The srm is inherently a nonlinear machine. The salient stator and rotor poles give rise to position dependent airgap reluctance which is harnessed to create useful torque. This torque production mechanism is highly dependent on the geometry of the poles and is characterized by static dependence on both stator coil current and rotor position. These nonlinear nature of srm causes torque ripple which is greater than that of other conventional electric machines, resulting in increased acoustic noise and speed ripples.

1.2 The Switching Reluctance Motor

1.2.1 Construction and Principle of Operation

The motor is designed to have salient teeth on the stator and rotor, which are constructed by stacking together steel laminations. Common configurations have two, three or four-phase windings on the stator. An example of a four-phase motor having 8 stator and 6 rotor teeth is shown in Fig.1.1.

In typical operation, energizing a phase of the motor makes the rotor turn in such a way that a pair of rotor teeth are aligned with the teeth of the energized phase. Therefore, the reluctance of the energized magnetic circuit is minimized, or equivalently, the inductance is maximized. The torque that is developed is independent of the current polarity, so operation from a single power supply is common. Sustained rotating motion is achieved by sequentially energizing the various phases as the rotor turns, so the well-known name is "switched reluctance motor". Motor construction is very rugged, since there are neither field windings nor magnets on the rotor and with proper design, the motor can have very high efficiencies.

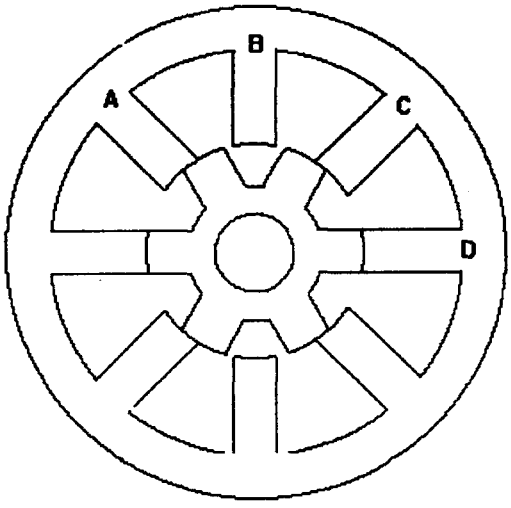


Figure 1.1 Crossectional view of a srm

In step motor structures, each set of coils which are excited separately for applying a specific pattern of excitation is called a 'phase'. The number of discrete positions in which the specific pattern of excitation can be established relative to the stator structure equals to the number of phases and denoted by 'q'. When excitation sequence is changed, the rotor moves by an angular displacement which is termed as 'step angle', α. The step angle of a srm is determined by the number of teeth on the rotor and stator, as well as the number of phases and can be calculated as follows;

$$\alpha = \frac{2\pi}{q.N_r} \qquad (1.1)$$

where N_r denotes the number of poles.

The operating principle of srm just described is rather straightforward. Let one phase of the windings, phase A (Fig.1.1) be energized by a constant voltage. The magnetomotive force setup by the current will position the teeth of the rotor section

under the excited phase, aligned with the teeth on the same phase of the stator. This is the position of minimum reluctance and the motor is in a stable equilibrium. For instance, when phase B is energized, rotor will rotate 15 degrees, counter-clockwise, and the rotor teeth will be aligned with the teeth of phase B. Continuing in the same way, the input sequence of phase of ABCDA.. will rotate the motor in counter-clockwise direction.

If the motor is energized with its rotor at the equilibrium position, no torque will be developed on the rotor shaft. When the rotor is displaced from the equilibrium position, a restoring torque is developed which tends to restore the rotor to its stable equilibrium position. This restoring torque is referred to as the "static torque". A typical static torque curve of one phase of a 8/6 (number of stator teeth / number of rotor teeth) structure is shown in Fig.1.2. The zero-degree position shown on the torque curve represents the stable equilibrium position of the rotor.

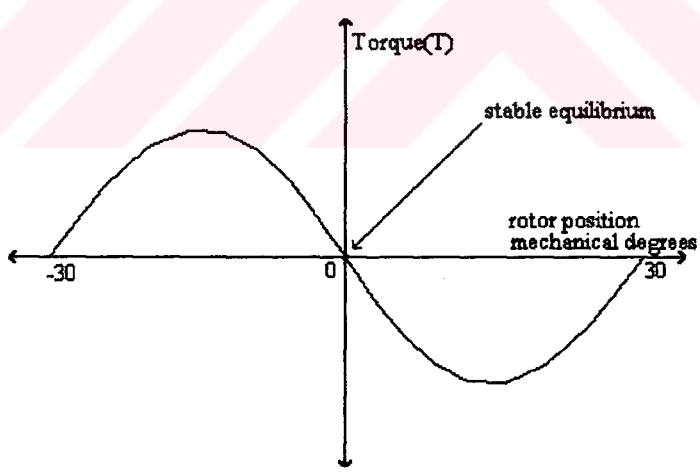


Figure 1.2 Typical static torque of a 8/6 srm

Fig1.3 illustrates typical static torque curves of a four phase srm. These curves are quite useful in the determination of the best control sheme for the motor. There is

a displacement between the characteristics corresponding to one step length, so that, for example, the equilibrium position for phase A excited is one step length away from the equilibrium position of phase B excited.

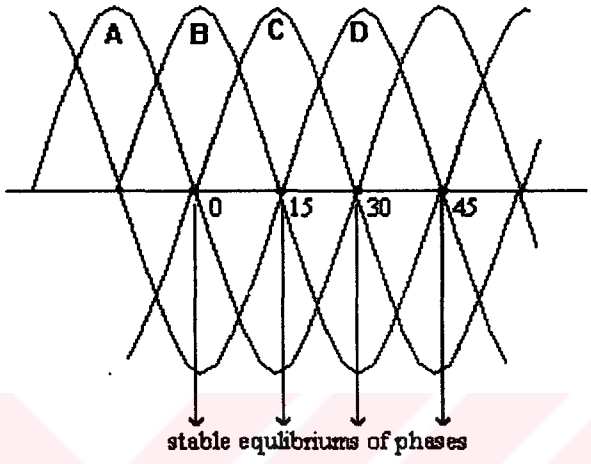


Figure 1.3 Typical static torque curves of a 8/6 srm

The shape of the static torque/rotor position characteristic depends on the dimensions of the stator and rotor teeth as well as the operating current. Static torque characteristic of the motor may be predicted by using finite element analysis software (ANSYS or similar) as described in chapter-2 as well as by using semi-numerical methods [16,17].

1.2.2 Torque Ripple of Switched Reluctance Motor

Torque ripple may be determined as the variations in the output torque. Since the amount and level of torque ripple are dependent on the speed of the motor, torque ripple should be determined according to the operating mode of a srm. Switched reluctance motors generally have three operating modes; low speeds (where phase

current is assumed to be constant), chopped current mode at medium speeds and high speed mode where phase current is dependent on many variables such as back emf . In order to predict the amount of torque ripple, current waveforms as seen in Fig.1.4 should be considered as well as static torque characteristics of the motor. Note however, the effect of torque ripple is most significant at low speeds where torque variations considerably affect instantenous speed of the shaft. At higher speeds inertia of the rotor helps to reduce these speed variations and hence vibration. Torque ripple determination in these operating modes is explained as follows.

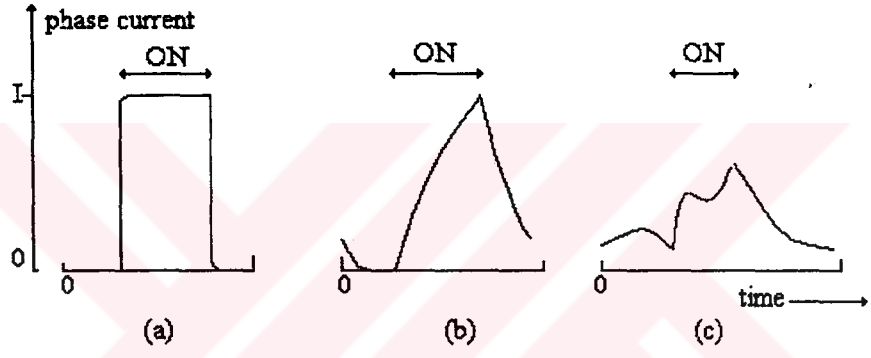


Fig.1.4 Typical current waveforms for a srm
a) low-speed b)medium-speed c)high-speed

A) Low-Speed Mode

Typical phase current waveforms for srm are shown in Fig.1.4. At the lowest operating speeds (Fig.1.4.a) the current waveforms are nearly rectangular, the build-up of current to the rated level occupies a minor portion of the excitation time.

For low speed applications, assuming that the overlapping of the phase currents is negligible, and phases are turned on and off at the rotor positions (x) corresponding to the intersection points of the torque curves of the phases (x corresponding to

T_{min}), the amount of torque ripple may be determined easily as the percentage ratio of peak torque to minimum torque (Eq.1.2). The maximum value of the static torque is known as the "peak static torque" (T_{max}) and for torque ripple prediction, the intersection points, as seen in Fig.1.5, of the torque curves of the phases is denoted by T_{min}.

$$\%\text{ripple} = \frac{T_{min}}{T_{max}} \cdot 100 \qquad (1.2)$$

The overall torque/rotor position characteristic of the motor is shown in Fig.1.5. In this thesis, an optimum set of dimensions of the motor (stator and rotor teeth and airgap lenght) are sought for different levels of excitation for low speeds to minimize torque ripple as discussed in chapter 4.

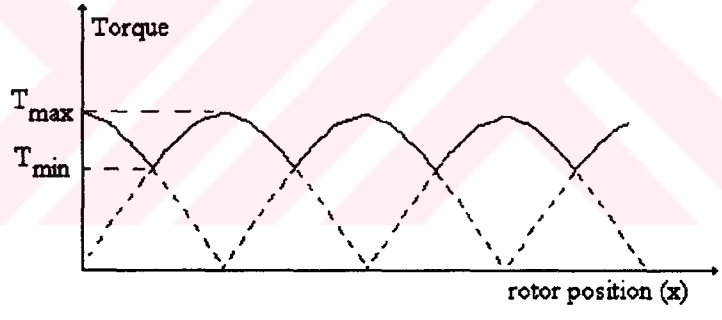


Fig.1.5 Torque ripple characteristic of a srm at low speeds

B) Medium-Speed Mode

For stepping rates where the phase current is only excited for a time comparable to the winding time constant, however, the waveform (Fig.1.4.b) is considerably distorted by the nearly exponential rise and decay of the phase current. Therefore,

for medium speeds, at a given speed, phase inductance (L) and resistance (R), output torque ripple characteristic of the motor may be obtained from the relevant static torque characteristic.

C) High-Speed Mode

At high speeds, each phase is excited for only a short time interval and the build-up time of the phase current is some significant proportion of the excitation interval. When a motor is operating at high speeds the current in each phase may not even reach its rated value before the excitation interval finishes and the phase is turned off. In addition, the time taken for the phase current becomes important at high speeds because the phase current continues flowing (through the freewheeling diode) beyond the excitation interval dictated by the drive transistor switch. The calculation of the pull-out torque at high speeds is complicated by the variations in current during the excitation time of each phase, which means there is no longer a simple relationship between static torque/rotor position characteristic and pull-out torque.

At high speeds, the voltage induced in the phase windings by the rotor motion should also be considered. The effect of these induced voltages can be seen from the high speed waveform on Fig.1.4.c, in which the waveform can no longer be described in terms of a simple exponential rise and decay. Even while the phase is switched on it is possible for the current to be reduced by the induced voltage, which is at its maximum positive value when the phase is excited. Similarly when the phase is turned off the decay of current can be temporarily reversed as the induced voltage passes through its maximum negative"[6]. Therefore, analysis of torque ripple completely depends on the analysis of the circuit model (Fig.1.6) of the one phase of a srm, that includes voltage induced in the windings at high speed operation. Torque ripple analysis and minimization of srm at high speeds is left aside as future study.

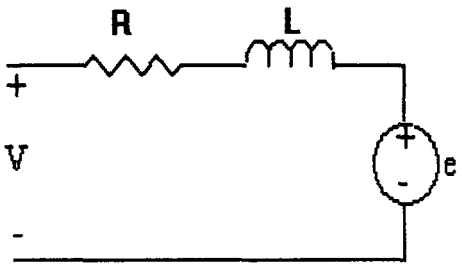


Fig.1.6 Circuit model for one phase of a srm

1.2.3 Torque Ripple Minimization

The form and level of torque ripple are dependent on the form of magnetic circuit, the level of its saturation, and the phase current waveform. There are two approaches that can be used for minimization of torque ripple of switched reluctance drive:

a) designing magnetic circuit geometry with minimum torque ripple
b) manipulate motor current to improve performance

Second approach to the problem requires fast and costly microprocessors for control purposes. In any case, it is desirable to look for an optimum geometry even if further improvement is desired through such an approach. The aim of this thesis is to find the optimum geometry with minimum torque ripple at low speeds.

1.3. Existing Research in the Literature

Up to now, plenty of studies have been conducted on switched reluctance motors. The reason of this interest is the advantages of the srm listed before. But, there are

still some unsolved problems inherent to srm which arise due to their working principles that are highly dependent on the heavy saturation of the magnetic material. A major problem that the researchers are still working on is the srm's torque ripple. In the following subsections, these research on srm will be reviewed by splitting the existing work on srm into two, reports on srm's in general and studies specialized on srm's torque ripple.

1.3.1 Brief Review of Previous Work on SRM's

Earliest published reports on srm belong to Unnewehr and Koch, Blenkinshop, Byrne and Lacy, Baushand and Rieke. Unnewehr and Koch presented a disc type srm in 1974 [35]. In 1976 a phase single stack srm is analyzed by Blenkinshop [9]. This work presented only the basic operation principles of srm. In 1977, in order to investigate the performance of srm's, Koch developed a linear theory of srm. In 1979, Corda presented his results on the design of a 0.75 kW srm [13]. Corda, in his work, formulated linear analysis for the performance calculations of a srm in a very systematic manner. He set out a number of different design criteria which are necessary for a reversible drive with self starting capability.

Lawrenson discussed the recent significant developments in srm drives in his paper in 1983 [23]. In his work the importance and advantages of such drives were emphasized.

In 1984, Chappell, Ray and Blake published a paper which discusses applications where microprocessors have been used in the control unit of srm drives [31].

In the following years many noteworthy papers were published on design and performance calculations. Some of these deserved more attention such as Unify and

Miller. Research results and methods described in the following work is most utilized here:

In 1982, Ertan showed an analytical method of static torque and inductance curve prediction for symmetrically slotted doubly salient devices in his published work [19].

In 1985, Tohumcu offered a method for optimizing the asymmetrically slotted srm in his Ph.D. thesis [33].

In 1987, Diriker sought the optimum airgap geometry of the asymmetrically slotted geometry in his M.S. thesis [15].

1.3.2 Brief Review of Studies Specialized on SRM's Torque Ripple

Determination of optimum geometry to minimize torque ripple is extremely difficult and requires the prediction of SR motor performance for all magnetic circuit parameters for a specified operating condition. Partly because of this reason, research in the literature concentrates on minimizing torque ripple for a specific geometry through shaping of the phase current[34] However, a few examples exist in the literature, which attempt to investigate the affect of pole arc on the torque ripple with the aid of finite element field solutions[36]. Brief review of studies in the literature specialized on srm's torque ripple is presented below:

B.Amin presented a paper about the analysis of the torque ripple and design considerations for improving the rate of the torque ondulations in variable reluctance motors in 1988 [2].

In 1990, Corda published a paper and discussed the effects of the magnetic circuits on torque pulsations of switched reluctance motor. In his paper several motor

structures are considered and by using field solutions of the motors, torque ripple levels are compared [14].

Wallace and Taylor, in 1990, investigated the effects of pole arcs on torque ripple by using finite elements method and control methods for minimization of torque ripple of three-phase srm [36].

Tormey and Torrey, in 1990, published a paper presenting an explication of the issues related to the design of a srm drive for an application which is sensitive to the supply current drawn by the drive and the torque ripple produced by the drive [34].

In 1991, Wallace and Taylor published a paper on low torque ripple switched reluctance motor for direct-drive robotics. A contribution of this paper is the discovery that, although a trade-off between peak torque and torque ripple exists, the maximum smooth torque and minimum torque ripple can be achieved by the same direct-drive motor [37].

Optimum commutation-current profile on torque linearization of switched reluctance motors is discussed by Schramm, Williams and Green in 1992 [32].

In 1993 Le Chenadec, Multon and Hassine published a paper about the current waveform optimization to minimize torque ripple [24].

Hung, in 1993, published a paper on the design of current waveforms that minimizes torque ripples in variable reluctance motors at low excitation levels [20].

O'Donavan et al, in 1994, published a paper presenting an artificial neural network solution to torque ripple reduction in a switched reluctance motor. They showed a very simplified controller resulting from the combination of neuro-learning and analytical insight [30].

1.4 Content of the Thesis

The aim of this thesis is to find an optimum switched reluctance motor geometry with minimized torque ripple for excitation levels where the current waveform is nearly constant during stepping. Determination of optimum geometry that minimizes torque ripple is extremely difficult and requires the prediction of SR motor performance for all magnetic circuit parameters for a specified operating condition. The research reported in this thesis however presents a general approach to the problem and inspects the effect of magnetic circuit parameters on the torque ripple while they are allowed to vary in a wide range. This is made possible due to having access to a set of dimensionless data [16,17], and a method which permits prediction of torque position and permeance position characteristics of a doubly-salient magnetic structure as a function of exciting MMF for a specified tooth width/tooth pitch (t/λ), tooth pitch/airgap length (λ/g) and normalized position (x_n).

In optimum design of a srm, knowing the steady state average torque is of vital importance. In srm's analytical expression of steady state average torque is not available due to heavy saturation and unusual nonlinear nature of srm's. Then instead of using analytical expressions, torque can be computed by means of various numerical field solutions. However, such methods are too long to be introduced into an iterative optimization process. Therefore, an easier method [19] of calculating the static torque curve, should be used. In this thesis, normalized permeance and force are calculated from the numerical magnetic data[16-19] for doubly salient geometries with identical teeth pairs.

In chapter-2, torque and flux linkage calculations of srm by finite element field analysis is introduced. The usage and abilities of finite elements analysis software (ANSYS) are presented. Method of measurements of torque and flux linkage

characteristics of srm is explained in detail and calculation results and measurements are compared in this chapter for each of two different test motors.

In chapter-3, first the essentials for artificial neural networks are cited. A method for using neural networks in the process of relating the design variables with torque ripple according to the data on hand is presented. Force data calculation for asymmetrically slotted structures from the force and permeance data of symmetrically slotted structures calculated by Ertan [19], are explained. Algorithms for neural network simulations and prediction of asymmetrically slotted structures is also contained in this chapter.

In chapter-4, a background for the torque ripple optimization problem is formed. First the general optimization problem is introduced, then the design problem is stated as a mathematical optimization problem, design variables are defined and the constraints that should be imposed on the optimization problem are set out. Torque ripple function as an objective is introduced and results of optimization problem for different excitation levels are presented.

In chapter-5, optimization results, namely optimum motor designs with minimum torque ripple, are verified by finite element field solutions. In other words, motor designs with optimum parameters found are modeled in ANSYS and static torque curves of these motors are obtained for the prediction of the amount of torque ripple. Results are evaluated and comparisons are made.

Chapter-6 covers an overall discussion of the thesis and the conclusions. Moreover, possibilities for future research are discussed.

CHAPTER 2

SWITCHED RELUCTANCE MOTOR FINITE ELEMENT FIELD CALCULATIONS AND TORQUE AND FLUX LINKAGE MEASUREMENTS

2.1. Introduction

Numerical solution of Magnetic fields is an important engineering tool for prediction of performance of various devices. The use of this technique for magnetic design purposes is only about 30 years old. It was first pioneered by Elderly, and is growingly becoming a very important modern tool for magnetic design. Finite Element method is one of the solution techniques and it has found a more general acceptance than the others. Finite element modelling is much newer in electrical engineering than in civil and mechanical engineering. While stress and temperature fields are the key distributed parameters obtained by finite elements for civil and mechanical engineers, electrical engineers need to obtain electric and magnetic fields. This fields must then be used to predict other performance parameters such as torque, current, voltage, power loss etc. ANSYS is one such professional Finite Element Field solution tool used by industry.

In this chapter, the use of this software for predicting the performance of switched reluctance motors will be described. In Finite Element field solution program the field is generally divided into number of discrete elements. For each element a set of

equations valid for that element is written. Then, the resulting set of equations are solved simultaneously while satisfying the boundary conditions.

The solution therefore greatly depends on how the user discretizes the problem and imposes boundary conditions. For that reason it is essential to test the program and its results for obtaining desired accuracy without excessively increasing the computation time. Although professional programs provide a field solution, often essential software for calculating the desired performance is not readily available. The user develops such 'macro's for the specific problem. Development of such tools for SR motor performance analysis and testing their accuracy has been the other purpose.

The work was started by solving the field for SR motors readily available in the laboratory. The tests on these motors (called SR1 and SR2 in this thesis) for finding their torque-position and flux linkage-position-current characteristics have also been carried out. The effect of distribution of meshes in the solution area is investigated also on this motors. Another issue investigated was to find out how to use a given mesh distribution for obtaining field solution for different rotor positions. This is extremely important for minimizing the effort to obtain the desired characteristics. The purpose was to identify factors affecting the prediction accuracy, and to find out how to minimize solution time while achieving desired accuracy.

For magnetic field analysis of the sr motors computer system with the following specifications was used:

HP series 720 Computer Workstation
1 Gbyte SCSI Disk
32 Mbytes RAM
HP - UX 8.07

Moreover PC based version of the program is also studied. But, because of the huge computational time, the system described above was preferred.

In this chapter, the finite elements analysis method and, ANSYS program are briefly described. Then the test and computation results for each motor is given in separate subsections. Measurement techniques are also discussed.

2.2 Finite Elements Method

The finite element method is a numerical procedure for analyzing structures. Usually the problem is too complicated to be solved satisfactorily by classical analytical methods. The finite element procedure produces many simultaneous equations, which are generated and solved on a digital computer.

> In general, the finite element method models a structure as an assemblage of small parts (elements). Each element is of simple geometry and therefore is much easier to analyze than the actual structure. In essence, a complicated solution is approximated by a model that consist of piecewise continuous simple solutions. Elements are called 'finite' to distinguish them from differential elements used in calculus. Discretization is accomplished simply by sawing the continuum into pieces and then pinning the pieces together again at node points. [11]

An important issue in a finite element analysis is the behaviour of the individual elements. A few good elements may produce better results than many poorer elements. We can see that several element types are possible by considering Fig.2.1. Function Φ, which might represent any of several physical quantities, varies smoothly in the actual structure. A finite element model typically yields a piecewise smooth representation of Φ. Between elements there may be jumps in the x and y derivatives of Φ. Within each element, Φ is a smooth function that is usually represented by a simple polynomial. For the triangular element

$$\Phi = a_1 + a_2x + a_3y \tag{2.1}$$

is appropriate it where the a_i are constants. These constants can be expressed (solved) in terms of Φ_1, Φ_2, Φ_3 which are the values of Φ at the three nodes. Triangles model the actual Φ by a surface of flat triangular facets. For the four node quadrilateral the bilinear function

$$\Phi = a_1 + a_2x + a_3y + a_4xy \tag{2.2}$$

is appropriate. The eight-node quadrilateral in Fig.2.1 has eight a_i in its polynomial expression

$$\Phi = a_1 + a_2x + a_3y + a_4xy + a_5x^2 + a_6y^2 + a_7x^2y + a_8xy^2 \tag{2.3}$$

and can represent a parabolic surface.

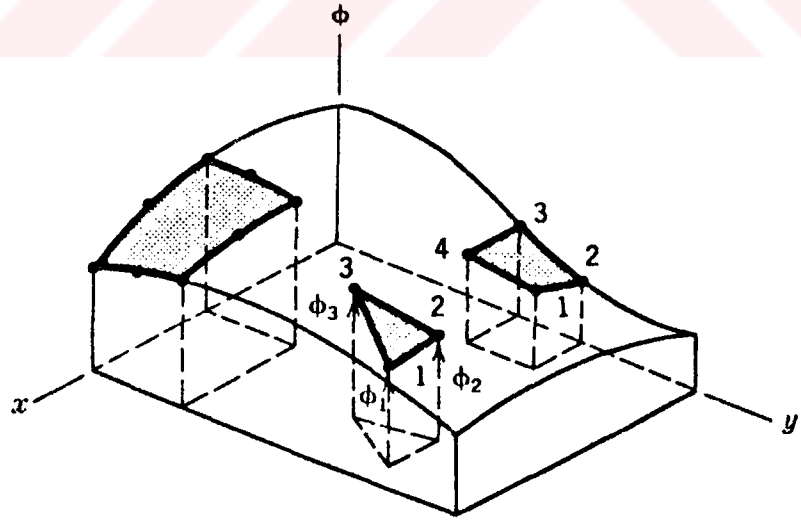


Figure 2.1 "A function $\Phi = \Phi(x,y)$ that varies smoothly over a rectangular region in the xy plane and typical elements that might be used to approximate it."[11]

Equations 2.1 to 2.3 are interpolations of function Φ in terms of the position (xy) within an element. That is, when the a_i have been determined in terms of nodal values Φ_i, Eqs. 2.1 to 2.3 defines Φ within an element in terms of Φ_i and the coordinates. If the mesh of elements is not too coarse and if Φ_i happened to be exact, then Φ away from the nodes would be a good approximation. Nodal Φ_i values are closed to exact if the mesh is not too coarse and if element properties are properly formulated. The finite element method solves for the unknown function Φ only at nodes Φ_i , and once this is accomplished the function Φ within the region between the nodes can be determined by interpolation.

Unfortunately, the choice of element type and size to be used in various part of the entire region in question is not simple and requires experience. But fortunately, as will be elaborated later in this chapter, professional finite element programs such as ANSYS has its own mesh generation program that divides the continuum into finite elements according to the basic guidelines defined by the user, automatically.

The importance of the choice pertaining to the type and size of elements may be visualized on some simple figures sketched for a one dimensional region. In Fig.2.2.a the true function Φ (quadratic) to be estimated is sketched as a function of x within the interval [0,a]. A simple element of size [0,a] shown in Fig.2.2.b with a linear interpolation function would do a poor job of capturing the actual quadratic nature of the response. Better results could be obtained with a greater number of linear elements (Fig.2.2.c) or a single element of size [0,a] that has a quadratic interpolation function could capture the actual distribution (Fig.2.2.d). The insight out of this example may easily be associated with the elements of two dimension, considering that for a triangular element, the interpolation function is linear (a plane) and for the quadrilateral it is bilinear (see Fig.2.1).

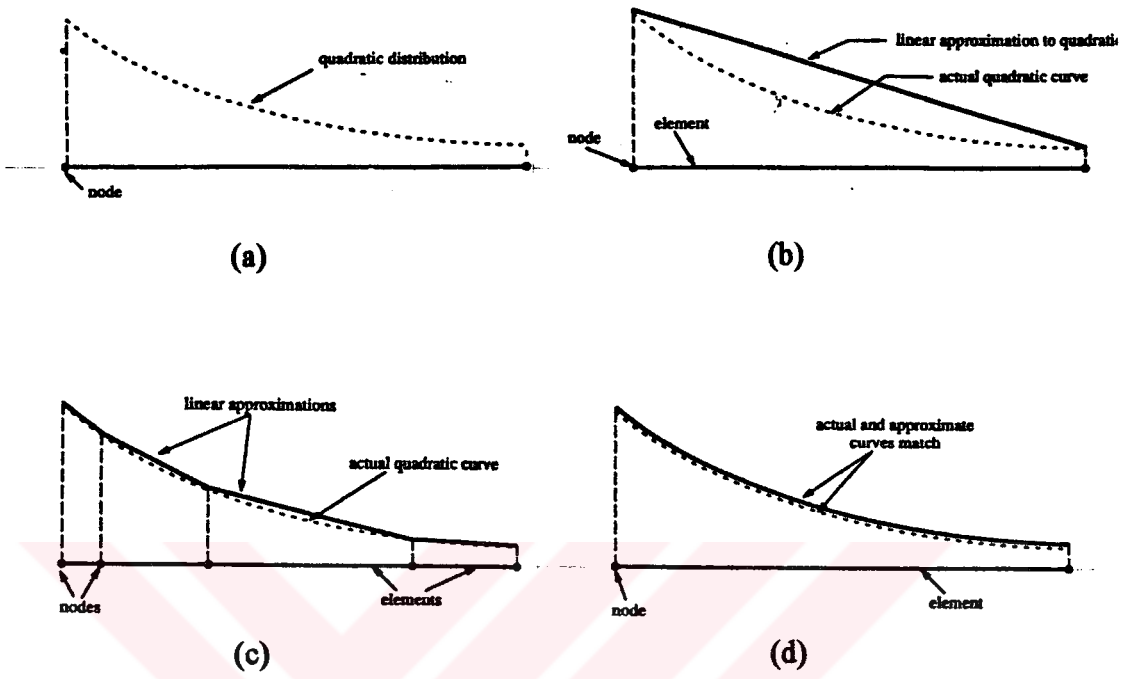


Figure 2.2 Simplified example pertaining to the importance of element type and size

As cited above, the primary task of the finite element method is solving the unknown function Φ, defined by some differential equations and boundary conditions, at the nodes specified. For any single element the problem is eventually reduced to a form such as

$$[K]_i \overline{\Phi_i} = \overline{A_i} \tag{2.4}$$

where $\overline{\Phi_i}$ is the unknown vector for the ith element with entries $\overline{\Phi_{ij}}$ denoting the unknown value of Φ at the jth node of element i (Fig.2.3.a). $[K]_i$ is a matrix and $\overline{A_i}$ is a vector obtained from several transformations of the differential equations and

boundary conditions that define the problem. These transformations involve the representation of the differential equations in their weak forms and some rearrangements according to the nature of the problem.

After equations of type Eq.2.4 are obtained for each element these should be combined in such a way that their relative locations on the entire space and their relations could be captured. To maintain this the nodes are labelled not only locally (in element level as in Fig.2.3.a), but globally (as in Fig.2.3.b) as well.

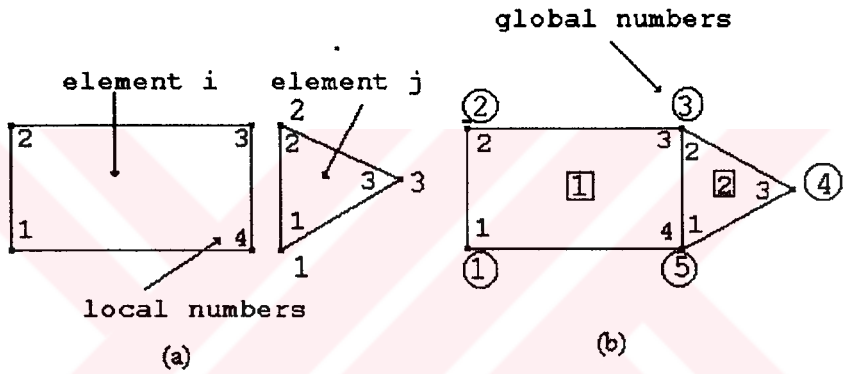


Figure 2.3 a. Local labelling of elements
b. Global labelling of elements

Then for element 1, in Fig.2.3.b the matrix form of the problem according to global coordinates is

$$\begin{vmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{21} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{vmatrix} \cdot \begin{vmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_5 \end{vmatrix} = \begin{vmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{vmatrix} \qquad (2.5)$$

and for the second element;

$$\begin{vmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{vmatrix} \cdot \begin{vmatrix} \Phi_5 \\ \Phi_3 \\ \Phi_4 \end{vmatrix} = \begin{vmatrix} g_1 \\ g_2 \\ g_3 \end{vmatrix} \tag{2.6}$$

As seen the only difference with the original formulation Eq.2.4 is that the entries of the vector $\overline{\Phi_i}$ is labelled globally rather than locally.

The combined formulation of this two element system is

$$\begin{vmatrix} k_{11} & k_{12} & k_{13} & 0 & k_{14} \\ k_{21} & k_{22} & k_{23} & 0 & k_{24} \\ k_{31} & k_{32} & (k_{33}+l_{22}) & l_{23} & (k_{34}+l_{21}) \\ 0 & 0 & l_{32} & l_{33} & l_{31} \\ k_{41} & k_{42} & (k_{43}+l_{12}) & l_{13} & (l_{45}+l_{11}) \end{vmatrix} \cdot \begin{vmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \end{vmatrix} = \begin{vmatrix} f_1 \\ f_2 \\ f_3+g_2 \\ g_3 \\ f_4+g_1 \end{vmatrix} \tag{2.7}$$

Although, it may not be observed in the simple two element system, in a model with large number of elements, a large but sparce matrix is obtained.

There exist special methods to store and solve these kinds of sparce matrix equations. Because, in large size problems the unnecesity of storing null entries is a very important advantage in terms of the storage capacity.

The matrix equation is finally solved either by direct methods that are based on Gaussian elimination (Cholesky decomposition, Front method) or by iterative methods where the solution vector Φ is solved iteratively (Gauss-Jacobi, Gauss-Seidel, Conjugate gradient method). Additionally, in professional packages such as

ANSYS, output interpretation programs called postprocessors, help the user sort the output and display it in graphical form.

2.3 Finite Elements Software: ANSYS

ANSYS is a computer program for finite element analysis and design. The program may be used to find out how a given design (e.g. a machine) works under operating conditions. It can be also used to calculate a proper design for given operating conditions.

ANSYS program can be used in all disciplines of engineering such as structural, thermal, mechanical, electrical, electromagnetic, electronic, fluid and biomedical. In this study only the static magnetic analysis facilities of the program is used.

In the following subsections basic features of ANSYS and the steps to be followed to build a model and solution of the field are described.

2.3.1 Static Magnetic Analysis Option of ANSYS

Two dimensional static magnetic field analysis is used to determine the solution of static magnetic fields of devices that can be modeled planar or axisymmetric configurations. The term "static" indicates the magnetic field is not a function of time; that is, transient effects are not considered. "Planar" indicates a device which can be modeled by a 2-D cut through its center. A device is set to be "axisymmetric" if a 2-D section can be revolved 360° about an axis. Planar or axisymmetric magnetic fields are important phenomena in many engineering applications; solenoids, actuators, motors, permanent magnet devices, transformers, etc.

Two-dimensional magnetostatic field problems are solved by minimizing a nonlinear magnetic energy functional containing a vector magnetic potential (A. In other words, the vector potential formulation is used. The ANSYS program solves these simultaneus equations using the frontal solution technique and this solver simultaneously assembles and solves the global set of simultaneous equations as follows:

1. The degrees of freedom (unknowns) and the [K] matrix for the element with the lowest order number are added to the global matrix which is initially empty.

2. Those degree of freedom that are not shared by any other elements yet to be added to the global matrix are expressed interms of the other degrees of freedom already in the matrix. The resulting equations are stored, and the degrees of freedom just processed are removed from the global matrix.

3. The degrees of freedom and the [K] matrix for the next element, in order number sequence, are brought into the global matrix.

4. Steps 2 and 3 repeated until all elements are processed. The last expression that was stored can be used to solve explicitly for the last degree of freedom.

5. The known degrees of freedom can then be used in stored expressions to solve for the remaining degrees of freedom. This action is called 'back substitution'.

The number of degrees of freedom (unknowns) in the global matrix at any given time constitute the 'wavefront'. The wavefront swells and shrinks as the elements are processed. Large wavefronts require more computer resources than small wavefronts. The wavefront is smallest if degree of freedom can be removed as soon as possible from the global matrix after the degrees of freedom are added. Since degrees of freedom are added based on the element order, adjustment of the

element order can be used to reduce the wavefront. Once the degrees of freedom are determined, derived results, such as flux densities are calculated within each element using its shape functions.

A single iteration is required if element matrix has constant components(i.e., constant material permeability). Multiple iterations are required (i.e., the analysis is nonlinear) under the following conditions:

- The B-H material curve is nonlinear
- The permanent magnet demagnetization curve is nonlinear

For a nonlinear analysis, the iterative procedure is based on the Newton-Raphson method to obtain a solution.

Whether the problem is linear or nonlinear, the 2-D static analysis solves for values of the out-of-plan, or z-directed, component (AZ) of the vector magnetic potential at each node, and for the magnetic flux density ($\overline{B}$). Nodal vector magnetic potentials are solved for in the planar case and circumferential directed magnetic potentials are solved for in the axisymmetric case. Other elements are calculated on a per-element basis.

2.3.2 ANSYS Modules

ANSYS contains three main modules (Fig.2.4): a preprocessor for model entry and generating a finite element mesh, a solver which generates the main numerical solution, and a postprocessor for deriving various parameters or creating displays. These steps of analysis are as follows:

PREPROCESSING:

The preprocessing phase is where all relevant data, such as material properties, geometry, and boundary conditions, are entered into the database in preperation of solution. Preprocessing steps are as follows:

A. Defining element types (four nodded magnetic elements are used for meshing operation)

B. Defining material properties. Options are as follows

Linear material properties

Nonlinear material properties (BH data of magnetic material is loaded)

C. Creating the model geometry (dimensions of the motor)

D. Defining analysis type (magnetic,thermal,..) and analysis option

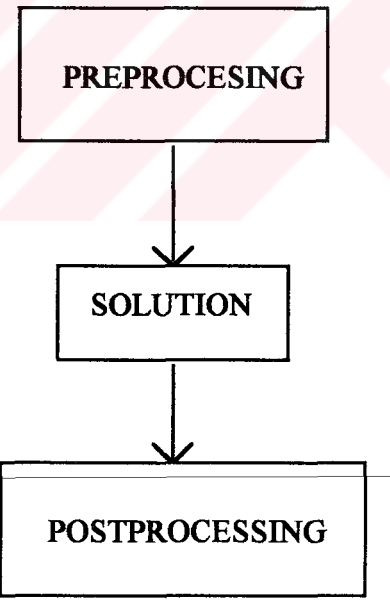


Figure 2.4 ANSYS Modules

SOLUTION:

In solution phase following steps are included:

A. Boundary Conditions;

*Boundary conditions are considered to be at symmetry planes of a model or at the far-field boundary of the model . Boundary options includes flux-normal, flux-parallel , far-field , far-field zero and periodic.

*Flux normal conditions are used to force flux to flow perpendicular to a surface. This is imposed naturally by the finite element method when no specifications are made at the surface of interest.

*Flux parallel conditions are used to force flux to flow parallel to a surface . This is imposed by setting the magnet vector potential , Az , to a constant value. The constant value is usually zero unless a non-zero field is being imposed.

In our applications flux parallel boundary conditions (A=0) are chosen on the exterior parts of the motor that surrounded by air. (Drichlet type or flux parallel boundary conditions)

B. Loads

*Loads that can be applied to a model include nodal potentials, element current density and nodal current segments.

*Nodal vector potential loads (A_Z) are used to apply nodal vector potentials to a node. This is generally done to apply an external field to a model by applying potentials at the model boundary.

*Source current density (J_S) loading is used to apply a uniformly distributed source current to an element. The loading is expressed in terms of current "density".

C. Specifying load step options:

Due to the nonlinear nature of the model, a two load step procedure is used. The first load step is used to ramp on the boundary conditions (nodal potentials and loads) . The second load is used to converge the iterations. Convergence controls used during solution also have to be defined . The specificiations of solition phase is as follows:

1. ANSYS- Analysis Options:

- Analysis options includes, STATIC, SUBSTRUCTURE analysis, etc.
- Newton-Raphson solution procedure may be selected. (In most cases, the program will choose by default the appropriate option) (NROPT command, adaptive descent procedure)
- Solver options include a FRONTAL EQUATION SOLVER and a JACOBIAN CONJUGATE GRADIENT SOLVER. (EQSLV-command).

2. ANSYS-Solution Options:

- Number of substeps within a load step may be specified (NSUBST-command)
- Number of equilibrium iterations within a substep to allow for nonlinear convergence may be specified by (NEQIT)-command.
- CNVTOL command is used to set the nonlinear convergence criteria for the analysis.
  - Convergence may be based on the out-of-balance load for the corresponding forcing load (Label=CSG)
  - Convergence may be based on the degree of freedom. (Label=A)
  - Program calculated default reference is recommended
  - Solution accuracy has to be set to at last a value 0.001 (0.1%) for the accuracy of the analysis. (Default value is 0.001).
- If convergence is not obtained, program may be terminated (NCNV command)

3. Linear Analysis:

- Linear analysis accurs when only linear properties are defined (i.e, relative permeability input only, no B-H curve).
- Solution requires only 1 substep.

4. Nonlinear Analysis:

- Nonlinear analysis occurs when B-H curves are input.
- Recommended solution sequence is to solve the problem over two load steps as follows:

  a) First load step:

  -Obtain approximate solution

  -1 equilibrium iteration per substep

  -Number of substeps (5-15) is chosen

  b) Second load step:

  -Obtain converged solution

  -Set number of equilibrium iteration (10-20)

  -Set convergence tolerance (default value 0.001)

For SR motor analysis here, nonlinear analysis solution sequence is used; Number of equibrium iteration is defined 20 , number of substeps is 5 and, convergence tolerance taken is 1.10^{-6}.

POSTPROCESSING:

Once the solution has been calculated ANSYS postprocessor can be used to review the results. Contour displays , tabular listings and many other postprocessor options are available and there are commands to make some calculations (e.g. line integral, cross product .. etc.). Some specifications of postprocessing phase are:

A. Nodal vector potential (A_Z) is available for processing.

-Flux lines are lines of constant A_Z in a planar analysis , and lines of constants A in an axisymmetric analysis (r = x coordinate)

- A_Z can be used to visualize flux lines

- Between any two nodes in a model , the flux passing between the nodes may be calculated as :

$$\phi = \oint \bar{A} \cdot \bar{dl} \qquad (2.8)$$

B. Nodal magnetic flux density (B) and field intensity (H) are available for processing.

C. Nodal Lorentz forces or Maxwell surface forces may be displayed.

2.4 Calculations of Torque and Flux Linkage Using Field Solutions

2.4.1 Torque Calculation:

A 'Macro' software is used for calculation of torque produced by the motor at a given position, once the field solution is obtained. The approach used in this software is briefly described below.

The torque on a rigid body can be obtained using the Maxwell stress tensor and is given by;

$$T = \int \left((\bar{B}.\bar{n}).(\bar{r} \times \bar{B}/\mu_o) - |\bar{B}|.2.(\bar{r} \times \bar{n})/2\mu_o \right) ds \qquad (2.9)$$

where r is the position and n is the normal vector.

For the two dimensional planar case $\bar{r} \times \bar{B}$ has only a single out of plane vector component and can thus be treated as the scalar

$$K = (r_x . B_y - r_y . B_x) \qquad (2.10)$$

therefore equation 2.9 becomes :

$$|T| = \int \left(K . \bar{B} / \mu_o . \bar{n} \right) . ds \qquad (2.11)$$

In all our applications a circular integration path is chosen through the center of the air gap and for different rotor positions (rotating the rotor nodes without modeling the geometry again as explained in the following sections) torque is calculated . The path of integration is circular about the origin and number of nodes used to define the path is chosen to be 36, it means the angle between nodes is 10 degrees. Integration path is shown in Figure 2.5 . It has to be remembered that the SI unit of calculated torque in this method is Nm/m and resultant torque is calculated by multiplying it with stack length.

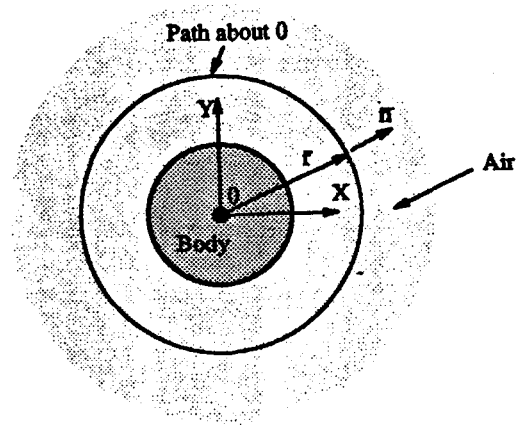


Figure 2.5 Path of integration for torque calculation

2.4.2 Flux Linkage Calculations:

As discussed in the previous sections, nodal magnetic potential (A_Z) is available in ANSYS postprocessing phase. A_Z can also be used to visualize the flux lines . (As seen in the A_Z graphs) . In this study there are two methods used for flux linkage calculations.

A) METHOD 1:

Between any two node in the model, the flux passing between the nodes may be calculated as in Eq 2.12:

$$\phi = \oint \overline{A} \cdot \overline{dl} \qquad (2.12)$$

In this case the integration is carried out following a closed loop extending into the z direction since A_Z has no component in the x direction. Contribution to the integral part of the contour along x direction is zero. If a unit length of magnetic core is considered in the z direction, it may be concluded that flux through the stator teeth as seen from Figure 2.6, may be calculated from the equation

$$\phi = (A_1 - A_2) \qquad (2.13)$$

where;

A_1 is the magnetic vector potential of node 1 in Fig 2.6

A_2 is the magnetic vector potential of node 2 in Fig 2.6

ϕ_2 is flux per pole per turn

The flux linkage per phase (Wb-Turns) may be calculated from the equation:

$$\psi = (2 * N) * (\phi_2) * L \qquad (2.14)$$

where ;

L : Stack length

N : Number of turns

On the other hand flux leakage may be assumed to be;

$$\text{Leakage flux} = 2\phi_1 - \phi_2 \qquad (2.15)$$

where ϕ_1 is the flux through surface shown in Fig.2.6. Leakage fluxes can be seen on A_Z graphs.

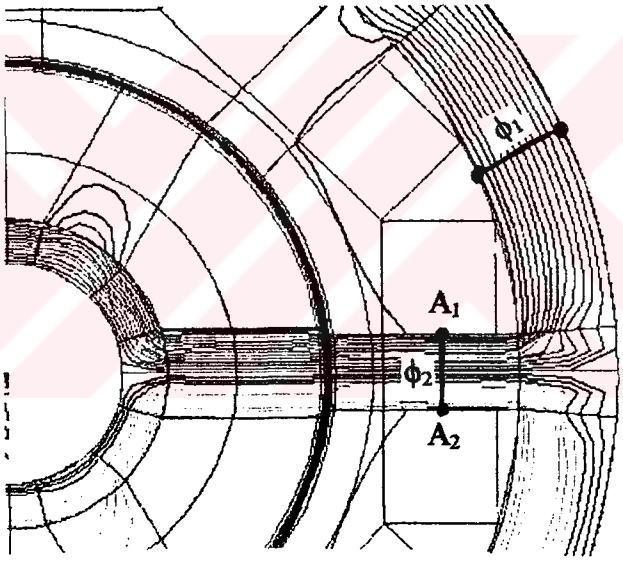


Figure 2.6 Calculation of flux per pole

Using this method, it is assumed that leakage flux will not be contained through the integration path, if the nodes on the edges of the slot are chosen . But, it is very difficult to predict where leakage flux deviates from the slot region. Another assumption, explained as the second method aims averaging the linkage flux values calculated between different node patterns placed on the coils.

METHOD 2:

A second method recommended by Prof. Jack is also tested . For this purpose a macro file is written to find average magnetic flux density over a coil.

In this approach flux may be calculated from;

$$\text{flux linkage /pole / turn} = \Phi = A_1 - A_2 \qquad (2.16)$$

where,

A_1 is the average flux of coil 1

A_2 is the average flux of coil 2.

Note that this method is similar to method 1. The only differenece in this case is an average node potential is considered for the calculations. In this study both are found to give almost identical results.

Macro file procedure:

The macro file written for this purpose traces the following steps.

1. From ANSYS magnetic field solution find nodal magnetic vector potential A_i for every node on coil 1 (Fig. 2.7)
2. From ANSYS magnetic field solution find nodal magnetic vector potential A_i for every node on coil 2.
3. Find average magnetic vector potential of coil 1 from

$$A_1 = (\sum A_i) / n \qquad (2.17)$$

where i= 1, .. n is the node number.

4. Find average magnetic vector potential of coil 2 from

$$A_2 = (\Sigma A_i) / n \qquad (2.18)$$

where i= 1, .. n is the node number. (coils have same number of nodes)

5. Then;

$$\text{linkage flux/pole/turn} = \Phi = A1 - A2 \qquad (2.19)$$

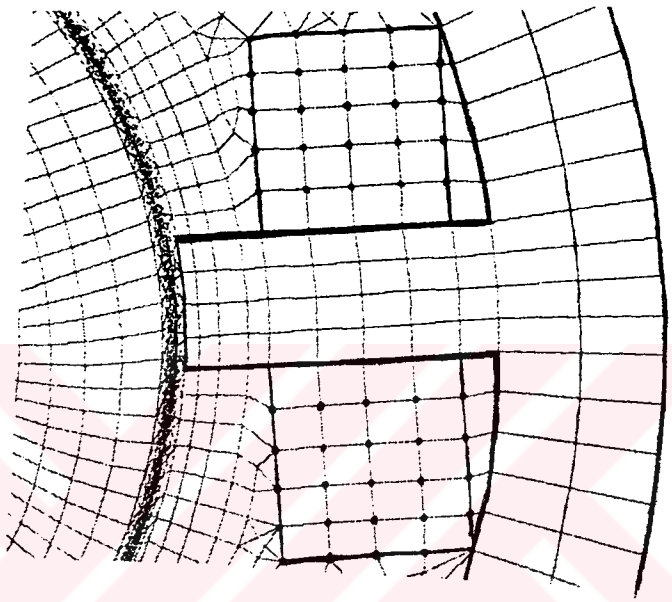


Figure 2.7 Nodes of Excited Coils

2.5 Solution of the Field and Calculation of Torque and Flux Linkage for Test Motor 1 (SR1)

The program used for the solution of the magnetic field with the purpose of calculation of the torque and flux linkage is briefly described in the previous sections.

In this section the application of the field solution technique to the switched reluctance motors described. The method is studied on two different switched reluctance test motors.

The effect of the mesh distribution and the factors affecting the accuracy of torque position (T vs θ) and flux linkage-position characteristics (λ vs θ) was first investigated on SR1. Measured T ve θ and λ vs θ characteristics of this motor was available from previous works. Solution results are compared with these measurements.

2.5.1 The specifications of the motor

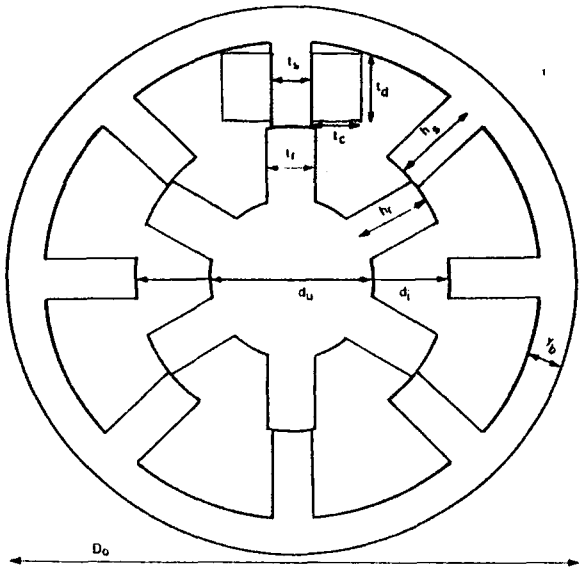
Dimensions of the test motor SR1 are as follows:

Outer diameter of rotor	:	70	mm
Outer diameter of stator	:	135	mm
Core length	:	91.5	mm
Backcore width	:	11	mm
Air gap length	:	0.2	mm
Stator pole width	:	8.5	mm
Rotor pole width	:	10	mm
Stator pole depth	:	21.3	mm
Rotor pole depth	:	17.5	mm
Number of turns	:	300	
Rated Current	:	6 Ampere/pole	
Wire size	:	0.7 mm	

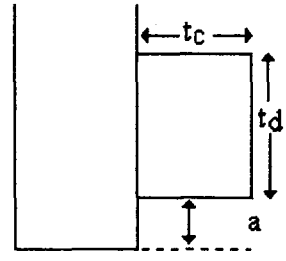
Maximum area for placing the coil is

a= 6 mm, t_c = 12.8 t_d = 12.6

The motor structure is shown in Figure 2.8



a) SR1 Motor lamination



b) A Phase coil of SR1

Figure 2.8 SR1 motor structure

STEP ANGLE

The step angle of the motor is simple to calculate as described in chapter 1 as follows;

$$\alpha = 360 / n.p = 360 / 4.6 = 15°$$

where n is number of phases and p is the number of rotor teeth.

POSITION NORMALIZATION

The torque curve of the motor is periodic over rotor tooth pitch (RTP)

$$RTP = n.\alpha = 4*15 = 60° \qquad (2.20)$$

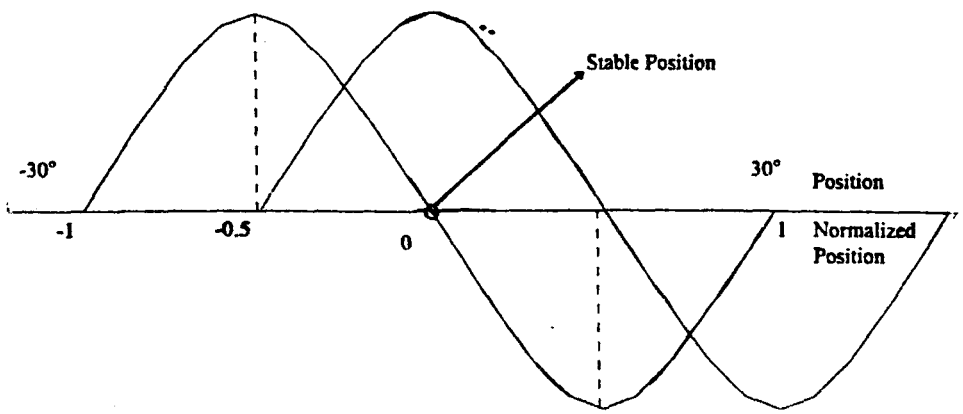


Figure 2.9 Torque-Normalized Position Curves of SRM

In this study rotor position is expressed in normalized form. The normalization is done over (RTP/2). In this notation, the normalized step angle (α_n) is calculated as;

$$\alpha_n = 15 / 30 = 0.5$$

LOADING

For the exterior nodes of the motor (exterior parts of stator and rotor) flux parallel boundary conditions are applied. It means parallel component of the magnetic vector potential is defined as zero. It is supposed that only one phase is excited at the time of analysis. Exciting conductors are seen in Fig.2.10.

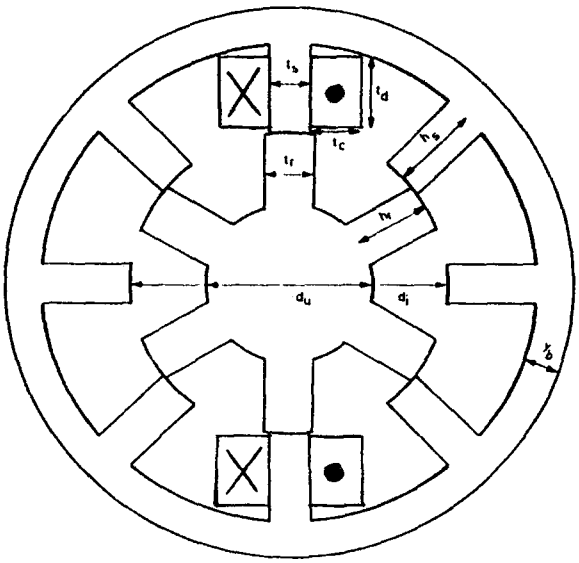


Figure 2.10 SRM with Excited Conductors

Current density is applied on every element in the area representing the coils (Fig.2.11) of a pole. For SR1 motor for 4 Ampere phase current, current density is:

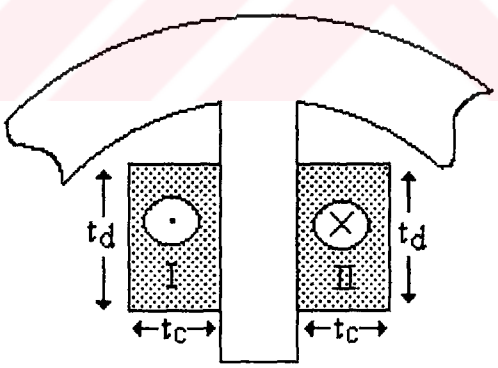


Figure 2.11 Conductors of SR1

$J = N.I / Area = 300.4 / (t_c . t_d) = 1200 / 1.61.10^{-4} = 7.44.10^6 A/m^2$

For Part II the direction is reversed. $J = -N.I / Area = -7.44.10^6 A/m^2$

2.5.2 Type of Element and the Distribution of Elements

Element type used for the solution of the problem here is PLANE -13/2-D Coupled-Field Solid. Plane 13 coupled field solid element is defined by four nodes with up to four degrees of freedom per-node. Magnetic, thermal, electrical and structural field capability may exist. The element has nonlinear magnetic capability for modeling B-H. curves or permanent magnet demagnetization curves. Plane 13 may be used as a four nodded or three nodded (rectangular or triangular options). element. Meshing properties of the part of the motor may be described as follows:

A) Airgap Meshing

For airgap meshing, rectangular elements are used. This is called mapped meshing. It requires that an area or volume be "regular" that is, it must meet certain special criteria:

a) The area must be bounded by either three or four lines,

b) It must have equal numbers of element divisions specified on opposite sides of the rectangular elements. They are preferred for meshing the airgap because;

i) For torque calculation, nodes along the same circular path have to be taken (accuracy of integration)

ii) For rotation of the rotor without remeshing, there must be equal size of elements on the surface of rotation. This process is explained below:

* Rotation of the Rotor: Rotor/Stator Stitching Operation

In order to obtain torque / position curves of the switched reluctance motor, several analyses are needed. To avoid generation of a mesh distribution for each individual position, ANSYS stitching operation is used. The procedure is summarized as follows.

1. Create stator solid model

2. Create rotor solid model

3. Mesh rotor and stator areas: On the border of the rotating part and stationary parts in the airgap, there are nodes generated at the same coordinate which are detached. For a quarter geometry, 180 nodes are generated on the border, so that the interval between nodes corresponds to a 0.5° angle and intervals are equal. For this reason, we may rotate the rotor part with an angle which is multiple of 0.5°.

4.Define rotor nodes at surface as detached components (CM Command) of the nodes on stator side

5. Define stator elements at surface as detached components (CM Command) of the nodes on rotor side

6. Select rotor nodes and stator elements at the surface.

7. Generate constraint equations. (CEINTF)

8. Apply boundary conditions

9.Solve

*If another analysis is needed for different position of the rotor;

1. Delete constraint equations by CEDELE command,

2. Select rotor nodes,

3. Detach solid model,

4. Rotate rotor nodes by a specified amount (NGEN-command),

5. Select rotor nodes and stator elements at the surface,

6. Generate constraint equations (stitching),

7. Solve.

B) Meshing Stator and Rotor Regions

In these parts of the motor rectangular elements are prefered. But, for non-regular areas such as triangular areas generated between the coilar and back core areas, triangular elements may be used.

For meshing other parts of the motor, the shape and distribution of the elements are carefully selected. Using a mesh that has low aspect ratio helps to minimize the discontinuities in the flux density components and the errors in the computed torque of a srm. For this reason the ANSYS program performs element shape checking (based on aspect ratio and shape angle) to warn user whenever a meshing operation creates an element having a poor shape. Here are some suggestions to decide whether element shapes are acceptible:

*Regions that are flattened or have excessively sharp corners generally cause mesh failure. Hence, mesh should be made dense enough so that elements with high aspect ratio would not be produced.

*Low aspect ratio elements should be used to avoid long and sharp triangles. "15°" is a proper lower bound for the internal angles of any element"

In designing the mesh structure of SR motor, the above mentioned principles have been followed. Additionally, since the airgap plays a crucial role in the determination of torque, a special emphesis is given to the mesh design of this region. The design is made such that the elements are smallest among of the whole mesh of the motor structure. Radially moving in both directions away from the airgap (along the magnetic circuit) the motor is first divided into eccentrical circles such that the radial distance between each successive circle is an increasing multiple of airgap length (x2, x3,....) to define ring areas. The borders of these areas are divided such that, the size of the elements increased radially outwards within the stator region. Then each ring is meshed by ANSYS in either triangular or rectengular elements. The perimeter of each ring is divided such that the above mentioned 15° constraint is eventually satisfied. This procedure allows us to have a satisfactory accurracy in torque and inductance with a considerably low number of elements.

2.5.3 The Effects of Mesh Distribution on Calculations

The ultimate aim of the field solution is to be able to predict the torque and inductance characteristics of the motor with good accuracy. However, field solution is time consuming even on fast workstation. Therefore, it is highly desirable to minimize number of elements, without loosing the accuracy of the solution. To study this issue three different mesh models are created for SR1. Mesh A and Mesh B as shown below have similar number of elements on the main structure. The major difference is in the number of elements in the airgap. Mesh B has about 50% less elements in this region.

In the case of Mesh C the number of elements in the airgap are the same as Mesh A. In this case, however, the number of elements in other regions is increased by about 50% to find out its effect on the solution time and accuracy. The specifications of the meshes are as follows:

MESH A: 8180 elements (total) Fig. (2.12,2.13,2.17)
2160 elements (in the air gap)
**3 ROWS of elements used for air gap mesh
*** Computation time is approximately 30 minutes

MESH B: 7960 elements (total) Fig . (2.12,2.14,2.17)
1440 elements (in the air gap)
**2 ROWS of elements used for air gap mesh
*** Computation time is approximately 20 minutes

MESH C: 12140 elements (total) Fig.2.15
2160 elements (in the air gap)
**3 ROWS of elements used for airgap mesh
*** Computation time is approximately 45 minutes

Moreover, accuracy of five rows of elements in the airgap region was also investigated (Fig.2.16). The type of element used for meshing and the rules followed in creating the mesh are discussed in the previous section and the resulting torque and flux linkage values for different mesh types are explained in the next section. Related flux contour graphs are shown in Figures 2.18,2.19 and 2.20.

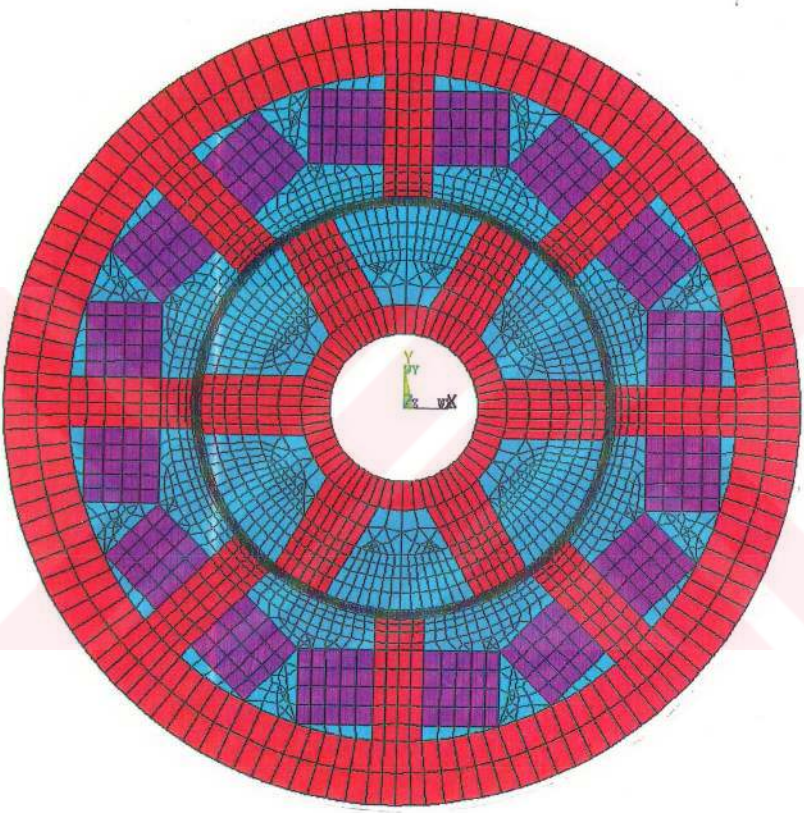


Figure 2.12 Mesh A,B general distribution

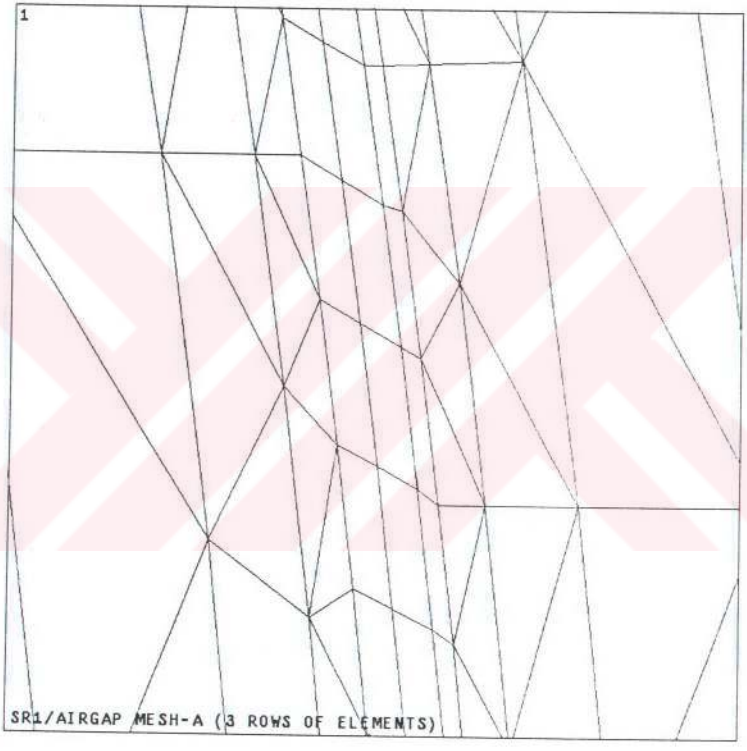


Figure 2.13 Mesh A, airgap region

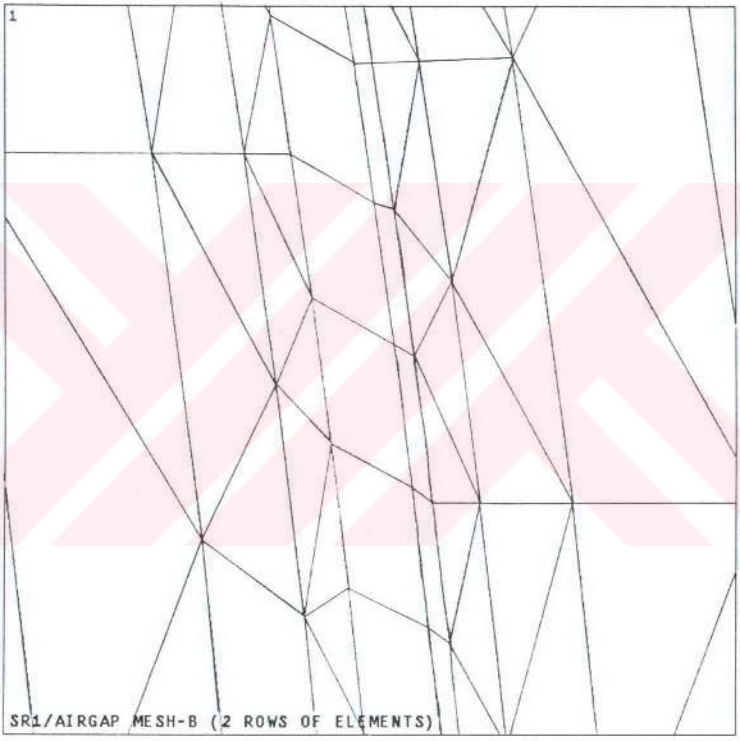


Figure 2.14 Mesh B, airgap region

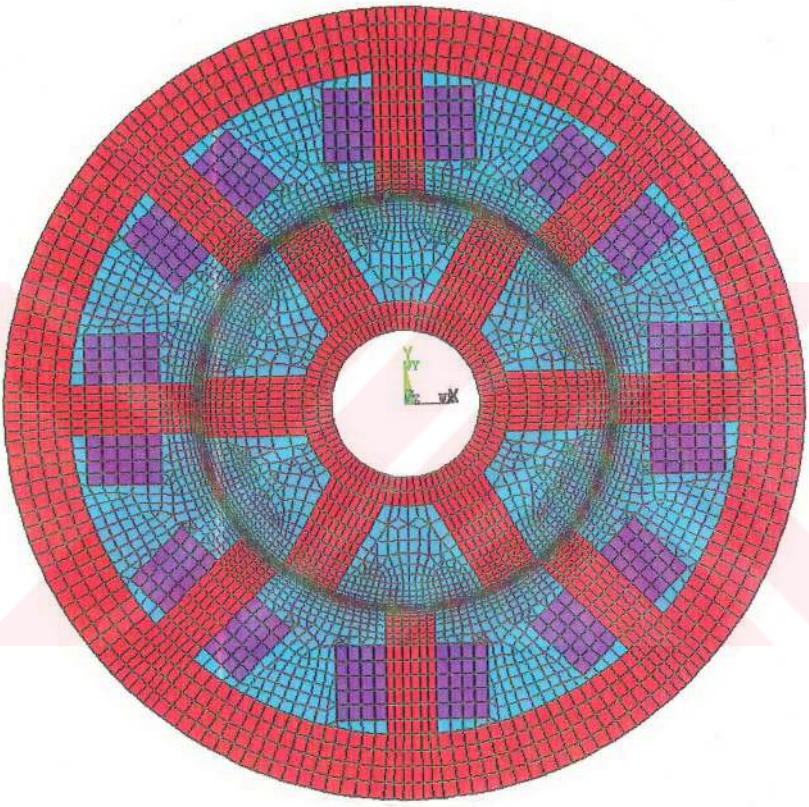


Figure 2.15 Mesh C general distribution

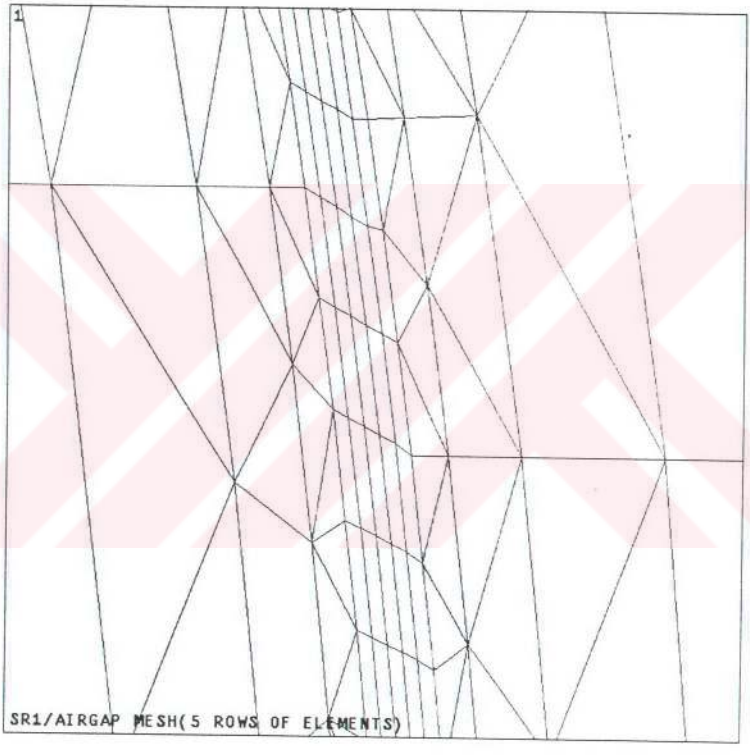


Figure 2.16 5-Rows of elements in the airgap

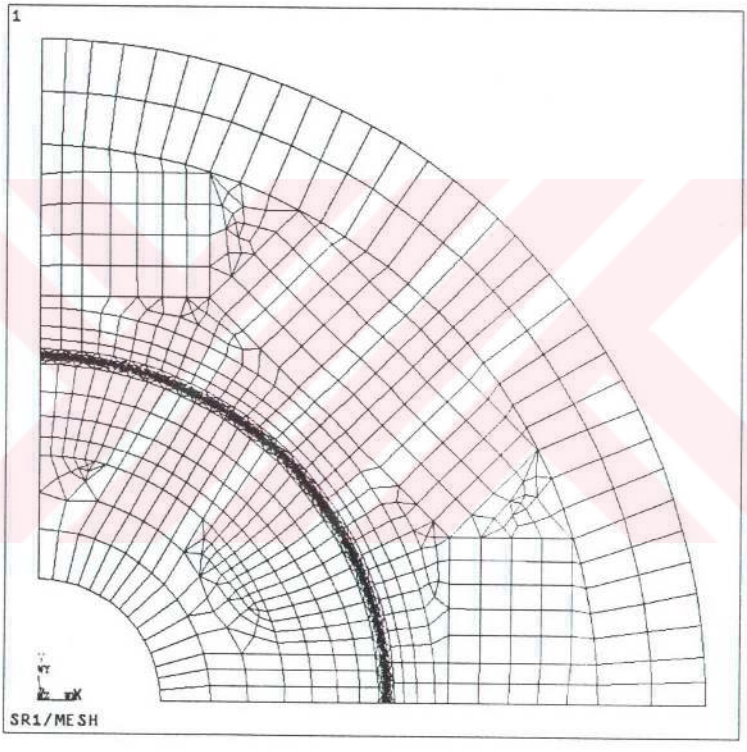


Figure 2.17 Detail of mesh A,B

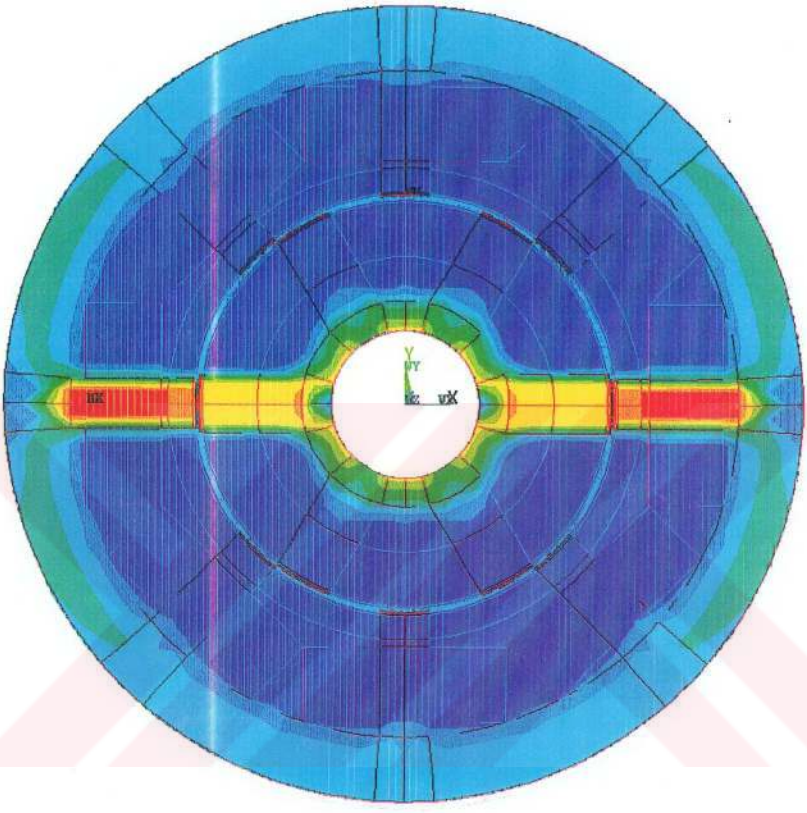


Figure 2.18 SR1 magnetic flux density distribution

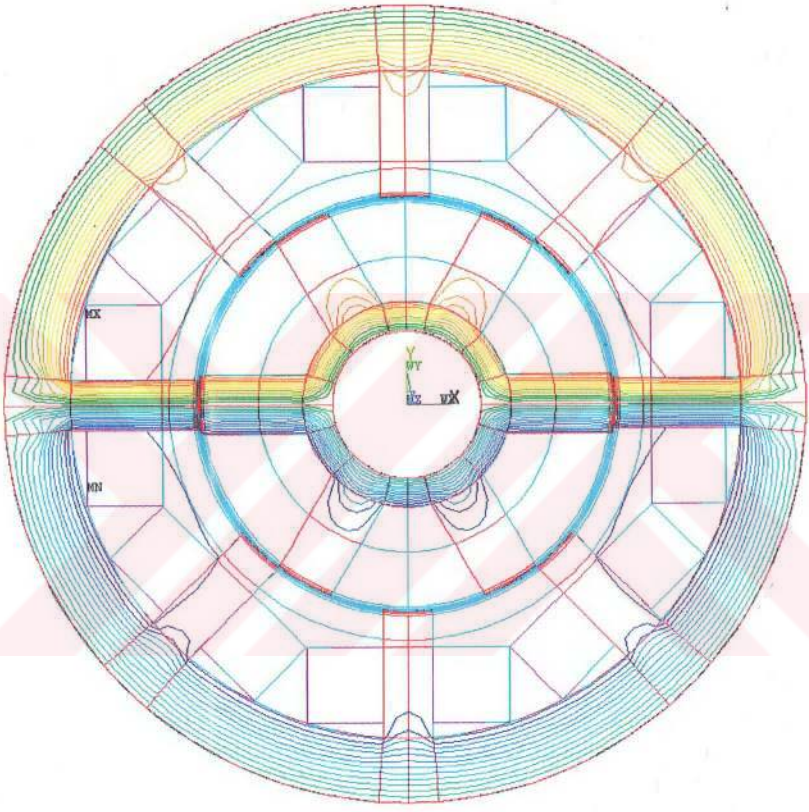


Figure 2.19 SR1 flux contours

2.5.4 Comparisons of Measured and Calculated Torque and Flux Linkage for SR1

A) Torque Prediction

The solution of the field is obtained for all three meshes described before and the static torque curve is computed from Maxwell Stresses for 6 normalized positions of the rotor teeth. Torque measurement of SR1 was made by Besenek[7] and the results are used. The measured and computed torque characteristic of the motor (or one phase on excitation at phase current of 4A) is shown in Figures 2.20 and 2.21 respectively, and is also tabulated in Table 2.1.

From these figures it can be observed that the solution obtained for reduced number of elements in the airgap is unacceptably inaccurate (Mesh B). However, Mesh A provides a very good accuracy. The error in computations from mesh B is obviously due to insufficient number of elements in the air gap. The rest of the mesh distributions for B is the same as mesh A.

Table. 2.1 Predicted and measured torque for SR1 at various positions at 4A excitation.

Normalized Position (x_n)	Measured Torque (Nm)	ANSYS (Mesh A)	ANSYS (Mesh B)
0.1	3.3	3.16	3.29
0.2	8.0	7.8	4.3
0.3	10.6	10.2	8.27
0.4	11.2	11.9	11.96
0.5	9.1	9.6	13.5
0.6	3.0	3.003	3.07

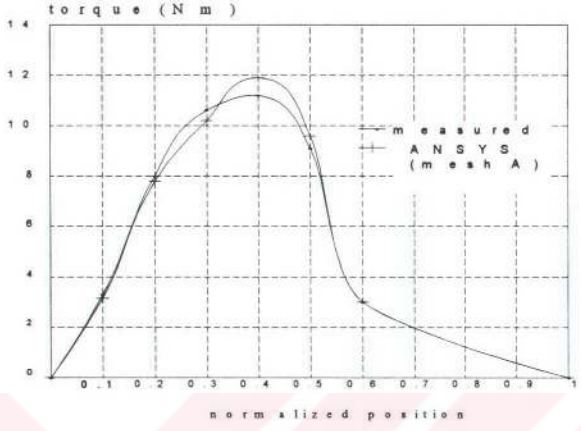


Figure 2.20 Predicted (Mesh A) and measured torque curves for SR1 at 4A

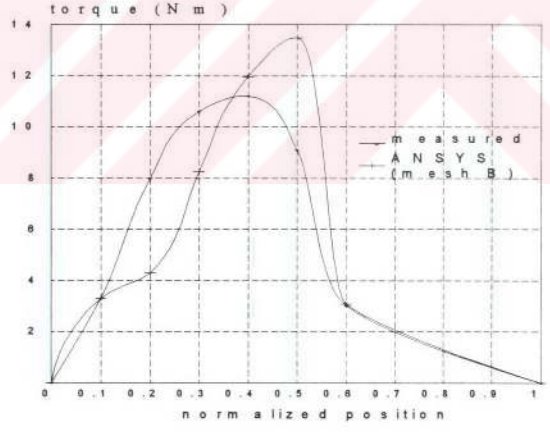


Figure 2.21 Predicted (Mesh B) and measured torque curves for SR1 at 4A

The solution obtained for Mesh C is also found to have a similar accuracy to that of obtained from Mesh A. However, the computation time was 50% more in this case. For this reason Mesh A is used for further investigation.

From the above results it may be concluded that the number of elements in the airgap is extremely important for obtaining acceptable accuracy and 3 rows of elements is the minimum number recommended for the airgap region for an acceptable torque curve prediction.

B) Flux Linkage Prediction

Measured (from Besenek's study [7]) and computed flux linkage position curve is given for the test motor (in one phase on mode at a phase current of 4A) for Mesh A and Mesh B in Table 2.2. and plotted in Fig.2.22. The method of flux linkage calculations is as described before and Table 2.2 shows that flux linkage is not very sensitive to mesh distribution in the airgap region.

Table 2.2 measured and predicted flux linkage for SR1 for 4A excitation

Normalized Position	Measured (I=4A) Wb.T	ANSYS (Mesh A) Wb.T	ANSYS (Mesh B) Wb.T
0	0.9	1.02	1.025
0.1	0.89	0.99	1.01
0.2	0.84	0.94	0.956
0.3	0.775	0.86	0.884
0.4	0.655	0.75	0.765
0.5	0.53	0.502	0.504
0.6	0.37	0.346	0.348

Investigation of Fig.2.22 displays a constant discrepancy between measured and computed flux linkage values. This is expectable since the end winding leakage which is independent of rotor position is neglected. However, this disprepancy seems to dissapear in normalized positions 0.5 and 0.6. This is likely to be due to a flux or position measurement error for these positions. On the other hand, assuming the flux distribution is same in axial direction and modelling the three dimensional problem as two dimensional may also be a source of error.

This point however, is not pursued further since the purpose of the exercise at this stage was to gain experience with the field solution programme. In the next section the results obtained for the 2nd test motor (SR2) are given. In this case the error sources are eliminated with the experience gained by the previous study and more accurate results are obtained.

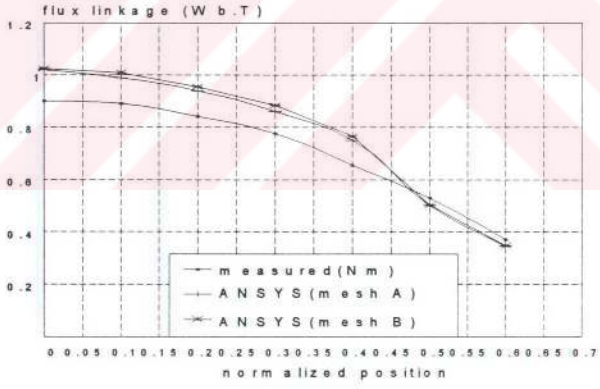


Figure 2.22 Measured and predicted flux linkage/position curves for SR1 at 4A

2.6 Torque and Flux Linkage Measurements and Calculations for Test Motor-2 (SR2)

2.6.1 Specifications of the Motor

The dimensions of the motor are given below and motor structure is shown in Fig.2.23. The coil region of SR2 is shown in Fig.2.24 and currents applied for the solution is shown in Table 2.3.

Outer diameter of rotor	:	38.6 mm
Outer diameter of stator	:	99.9 mm
Core length	:	40 mm
Backcore width	:	10.1 mm
Air gap length	:	0.255 mm
Stator pole width	:	8.2 mm
Rotor pole width	:	8.2 mm
Stator pole depth	:	30.395 mm
Rotor pole depth	:	7.2 mm
Number of turns	:	322
Rated Current	:	3 Ampere/pole
Wire size	:	0.7 mm

Table 2.3 Currents applied to the conductors of SR2

For I = 1A		For I = 2A		For I = 3A	
I	$J = N.I / Area$ $J = (1.322) / 1.12.10^{-4}$ $J = 2.8.10^{6}\ A/m^2$	I	$J = (2.322) / 1.12.10^{-4}$ $J = 5.75.10^{6}\ A/m^2$	I	$J = (3.322) / 1.12.10^{-4}$ $J = 8.6.10^{6} A/m^2$
II	$J = -2.8.10^{6}\ A/m^2$	II	$J = -5.75.10^{6}\ A/m^2$	II	$J = -8.6.10^{6}\ A/m^2$

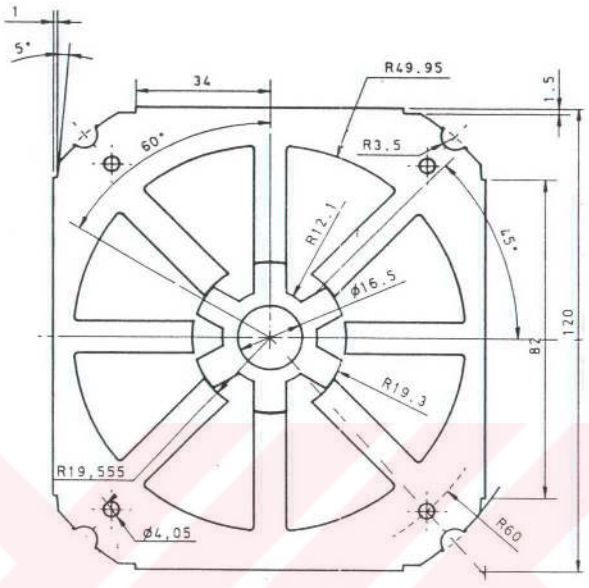


Figure 2.23 Lamination of SR2 (air gap length= 0.255mm stack length= 40mm)

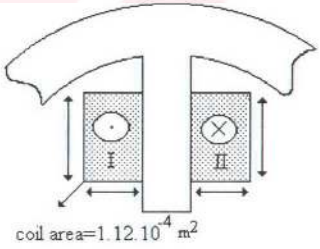


Figure 2.24 Conductors of SR2

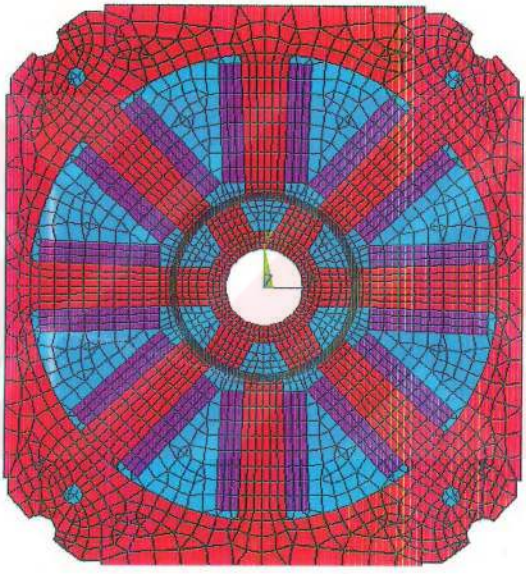


Figure 2.25 SR2 mesh distribution

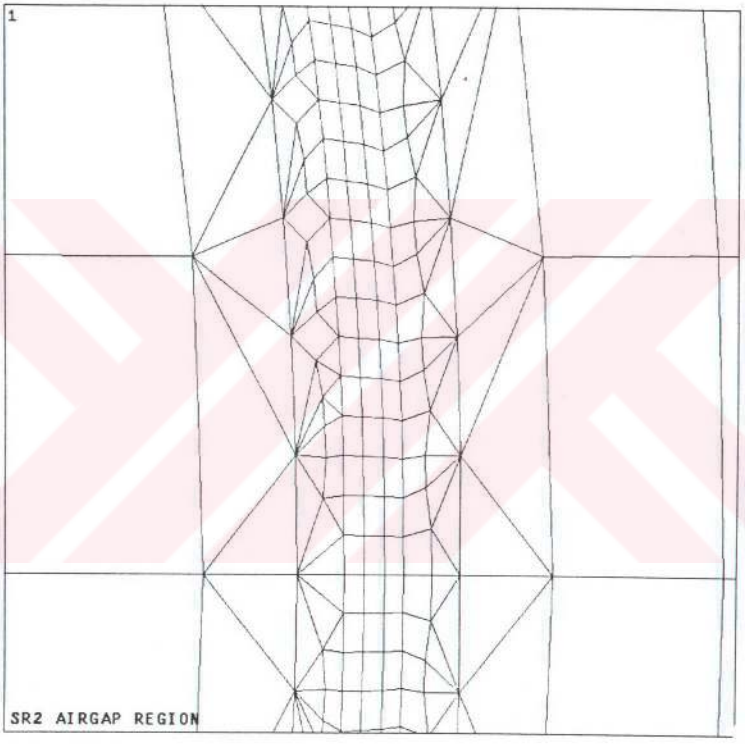


Figure 2.26 SR2 airgap mesh

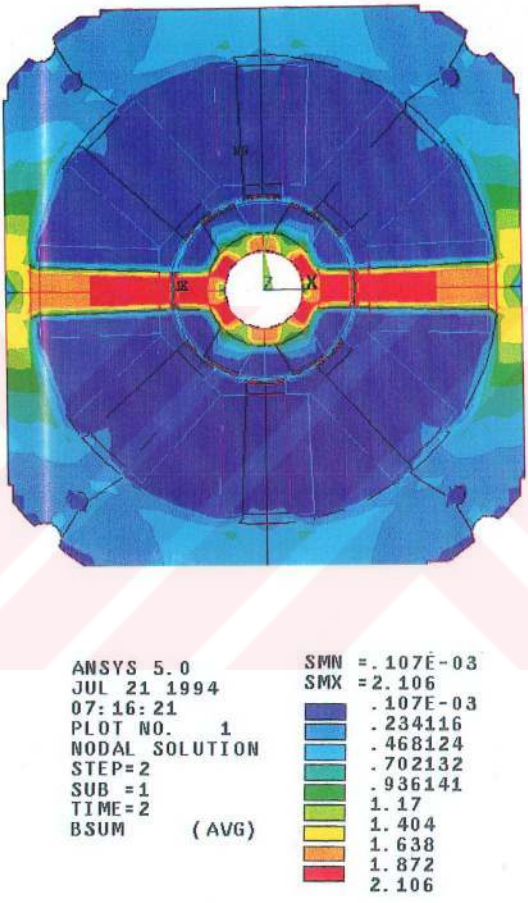


Figure 2.27 SR2 magnetic flux density distribution

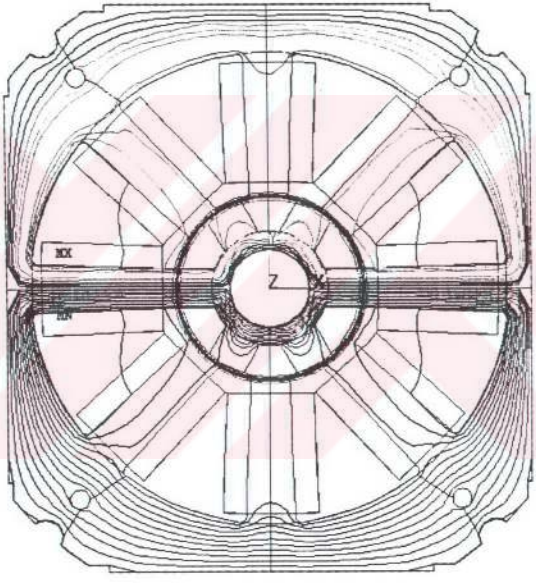


Figure 2.28 SR2 flux contours

2.6.2 The Field Solution and the Mesh Distribution

In the light of the experience obtained, the mesh distribution shown in Figures 2.25 and 2.26 is generated. Note that, in order to minimize the number of elements and obtain a reasonable accuracy, the element sizes along the pole are increased in a geometric sequence (g, 2g, 4g etc). Elements are generated as described in 2.5.2.

Number of Elements: 8704
Number of Nodes: 7918
Type of Element: rectangular - triangular
rows of elements in air gap: 3
solution accuracy: 10^{-6}

The field solution is obtained with this mesh for 9 different rotor positions at different excitation levels. The solution times changed between 25-35 minutes. A typical result for 3A excitation level is shown in Figures 2.27 and 2.28 when one phase is excited.

2.6.3 Torque and Flux Linkage Measurements for SR2

A) Torque Measurements

The test set up used for torque measurements is shown schematically in Fig.2.29 The device designed is simple but effective. Fixing rod in the figure stops rotation of the shaft. When one of the phases is excited, the rod on which micrometer rests (positioning arm) is set in a horizantal position. The motor is now in the IN position. Then the positioning arm in Fig.2.30 is adjusted with the aid of a screw until it deflects as much as desired. The new position of the rotor can be simply calculated as shown in Fig 2.31.

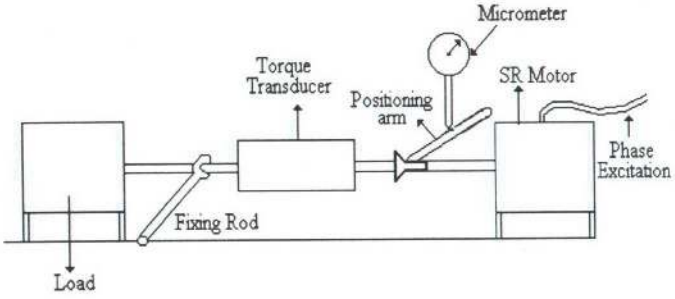


Figure 2.29 The set-up for measuring static torque

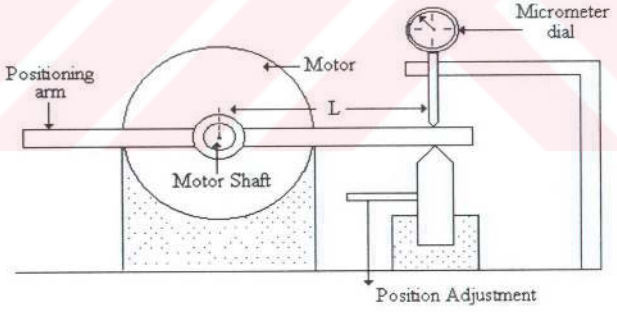


Figure 2.30 Shaft positioning set-up for torque measurements. (A section of the set-up in Fig.2.31)

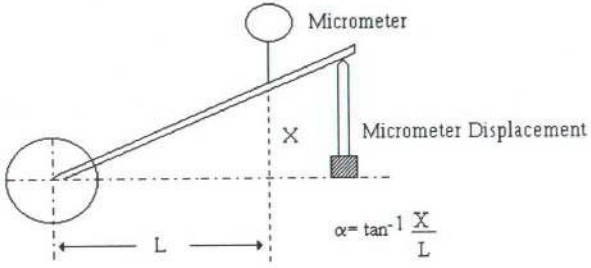


Figure 2.31 Position measurement

The reading of the torque transducer is then recorded and converted to torque at 1m if necessary. The torque transducer used in the tests is

Brandname: SHC / Himmelstein
Model: 24-02T(1-2)
Range: 100 lb-in (11.276 Nm)
Max.speed. 15000 rpm

The measured torque/position curves of this motor are given Fig.2.35 and Table 2.4 for one phase on excitation at three different excitation levels.

Table 2.4 Measured torque values for SR2 for different excitation levels

Normalized Position	1A Torque (Nm)	2A Torque (Nm)	3A Torque (Nm)
0.1	0.36	0.52	0.61
0.2	0.40	0.72	0.92
0.3	0.41	0.88	1.20
0.4	0.43	1.06	1.50
0.5	0.49	1.16	1.75
0.6	0.45	1.14	1.88
0.7	0.45	1.12	1.85
0.8	0.37	0.91	1.71
0.9	0.20	0.36	0.58

B) Flux Linkage Measurements

For flux linkage measurements again the positioning device in Fig.2.31 is used. For measurement of flux linkage a circuit shown in Fig.2.32 is used.

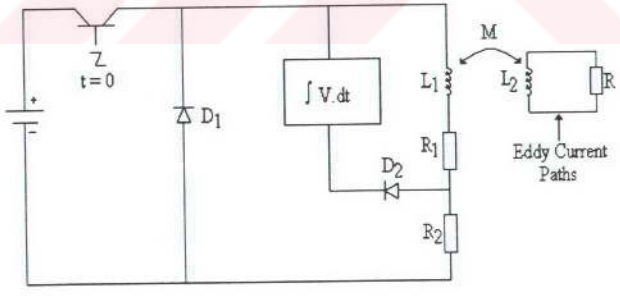


Figure 2.32 The circuit for measurement of flux linkage

The transistor switch in the circuit was designed after a considerable amount of experimentation with mechanical switches. It is observed that mechanical switches fail to give satisfactory results because of bouncing problems.

The integrator is also very important for obtaining accurate results. Various devices were tested to find out their suitability as an integrator. Drift and noise in the signal received was found to be an important problem. To avoid noise a coaxial cable is used to carry the signal to the integrator

A Gould 1602 digital storage scope with integration facility was found to be the most suitable device as integrator in this circuit.

When transitor in the circuit is turned on (manually by supplying its base) current circulates through the loop formed by the phase coil, resistance R_2 and the clamping diode. The indicated voltage-time integral in Fig.2.33 can be found in terms of the circuit parameters as shown below

$$\int V.dt = -Li + Li\frac{R_1}{R_1 + R_2} \tag{2.21}$$

$$\psi = L\,I \tag{2.22}$$

$$\psi = -\frac{(R_1 + R_2)}{R_2}.\int V.dt \tag{2.23}$$

The measured flux linkage data for different positions and different excitation currents is tabulated in Tablo 2.4 and shown in Fig.2.34

Table 2.4 Measured flux linkage values for SR2

Normalized Position	0.5 A	1 A	1.5 A	2 A	2.5 A	3 A	3.5 A
0.0	0.248	0.386	0.473	0.511	0.576	0.596	0.604
0.1	0.221	0.367	0.452	0.498	0.546	0.574	0.594
0.2	0.195	0.345	0.432	0.478	0.521	0.559	0.577
0.3	0.189	0.324	0.405	0.458	0.505	0.529	0.555
0.4	0.148	0.278	0.378	0.441	0.491	0.519	0.544
0.5	0.123	0.251	0.337	0.406	0.40	0.498	0.531
0.6	0.102	0.218	0.309	0.376	0.438	0.476	0.512
0.7	0.091	0.179	0.249	0.317	0.368	0.406	0.455
0.8	0.088	0.141	0.226	0.295	0.345	0.391	0.425
0.9	0.061	0.113	0.187	0.245	0.315	0.358	0.395

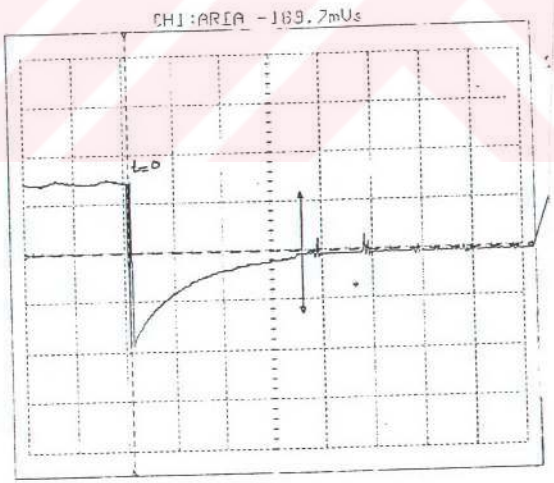


Figure 2.33 Typical integral signal

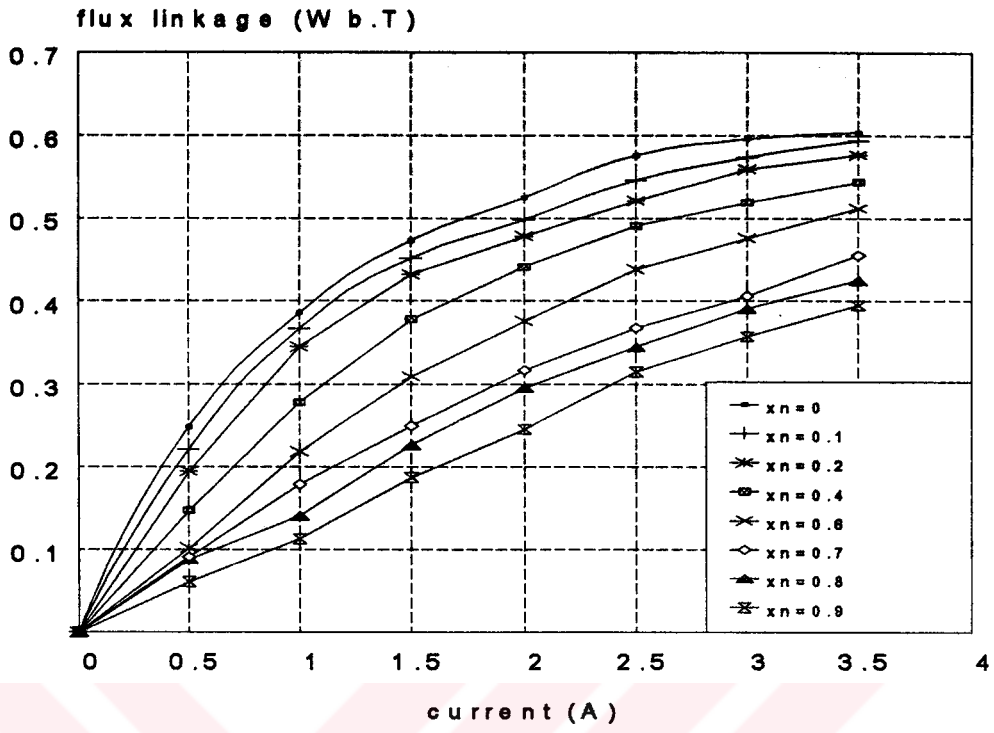


Figure 2.34 Measured flux linkage values for SR2

2.6.4 Comparison of Measured and Calculated Torque and Flux Linkage for SR2

A) Torque Prediction

Measured and computed static torque curves for SR2 are plotted in Fig.2.35 against normalized position for three different excitation levels, and also tabulated in Table.2.6.

The results display an extremely good agreement between measurements and computations. The error is largest for the low excitation level (1A). This is probably because the measured torque is less than 5% of the rating of the torque transducer. For the highest excitation levels (3A) ANSYS prediction gives about an 8% higher

value for certain positions (0.6 and 0.7) than measured torque. This is an indication that the airgap size in the manufactured motor may be more than specified. To find the effect of airgap size, the field solition procedure is repeated at 3A excitation level for airgap length 0.3 mm.The results arr given in Table 2.7 and plotted in Fig.2.36.

Table 2.6 Measured and calculated torque of SR2 for different excitation levels

	measured torque(Nm)			predicted torque (ANSYS)		
normalized position	I= 1A	I= 2A	I= 3A	I= 1A	I = 2A	I = 3A
0.1	0.36	0.52	0.61	0.272	0.492	0.606
0.2	0.40	0.72	0.92	0.313	0.695	0.890
0.3	0.41	0.88	1.20	0.343	0.898	1.201
0.4	0.43	1.06	1.50	0.360	1.040	1.524
0.5	0.49	1.16	1.75	0.362	1.106	1.800
0.6	0.45	1.14	1.88	0.371	1.134	1.980
0.7	0.45	1.12	1.85	0.370	1.153	2.000
0.8	0.37	0.91	1.71	0.320	1.032	1.800
0.9	0.20	0.36	0.58	0.070	0.261	0.583

Table 2.7 Measured and predicted (at g=0.3 mm) torque of SR2

normalized position	measured torque (Nm)	predicted torque (g=0.3mm)
0.1	0.61	0.47
0.2	0.92	0.71
0.3	1.2	0.91
0.4	1.5	1.28
0.5	1.75	1.632
0.6	1.88	1.80
0.7	1.85	1.88
0.8	1.71	1.76
0.9	0.58	0.54

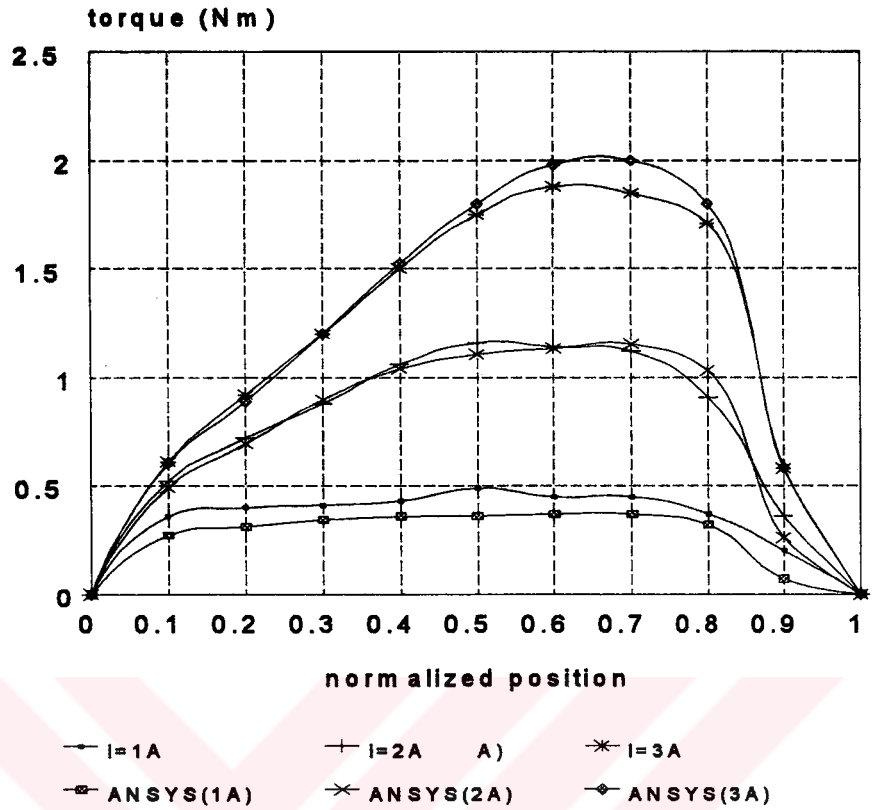


Figure 2.35 Measured and calculated torque of SR2 for different excitation levels

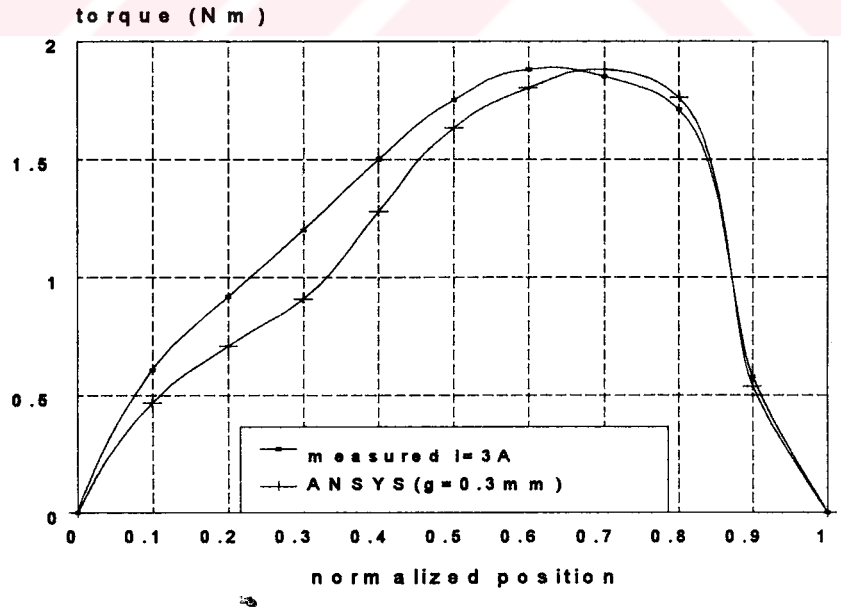


Figure 2.36 Measured and predicted (at g=0.3 mm) torque of SR2

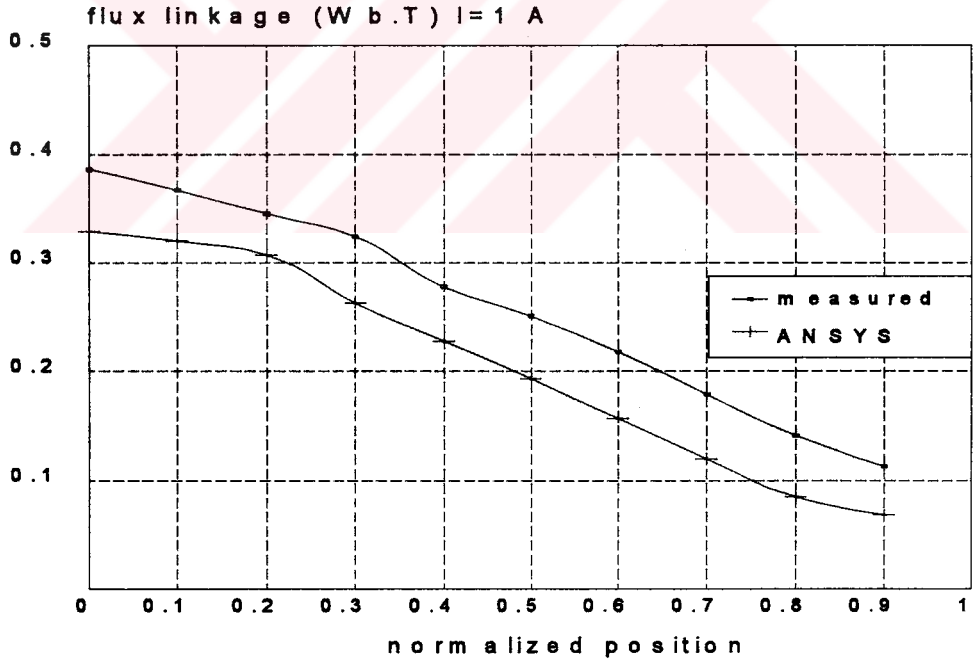
Comparison of the field solution for SR2 at g = 0.255 mm and g = 0.3 mm indicates a considerable detoriation in accuracy although the peak torque is better predicted. This suggests that the reason for prediction of higher torque for the designed airgap (0.255 mm) is not likely to be due to a much larger airgap in the test model but perhaps due to the difficulty of representing behaviour of the magnetic material under extreme saturation conditions. Extreme saturation levels occur at the corners of teeth around positions which correspond to peak torque due to very small overlap and channeling of tooth flux to this region.

B) Flux Linkage Prediction

Measured and computed flux linkages for SR2 are plotted against position for three phase current values (1A, 2A, 3A) in Fig.2.37 and tabulated in Table 2.8. Investigation of this figure show that the shape of the curve is well predicted for all current levels. However, as expected a constant discrepancy exists between measurements and computations. This discrepancy is attributable to the end winding flux linkage which could not be taken into account in two dimensional field solutions. The fact that the magnitude of the discrepancy increases with current level supports this conclusion. It is possible to compute and account for end winding flux leakage if desired. Moreover, leakage fluxes are also calculated by the method explained in 2.4.2 and small values around 10^{-4} are found and then neglected.

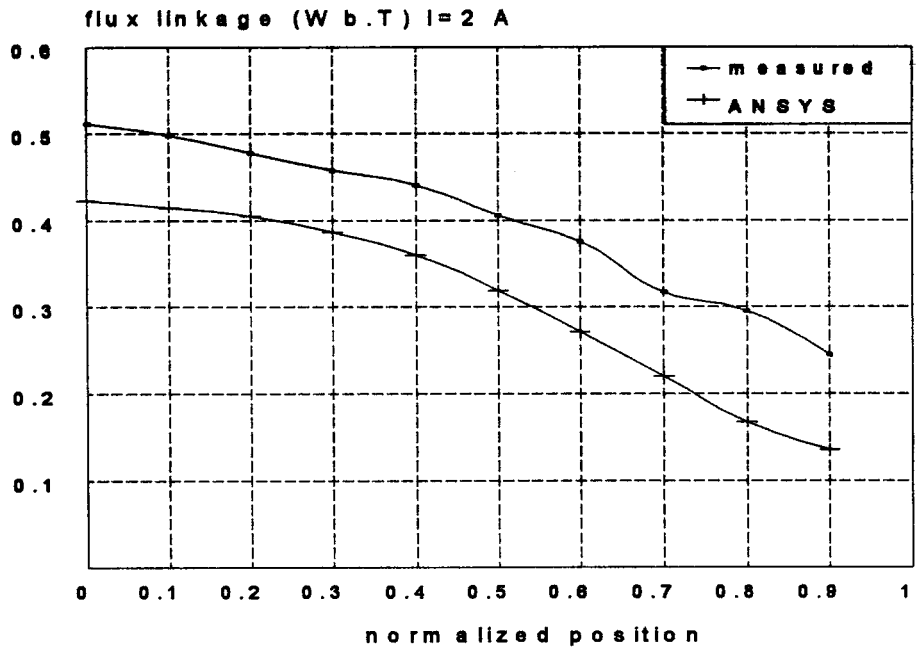
Table 2.8 Measured and Predicted Flux Linkage for SR2 (Wb.T)

	measured flux linkage-Wb.T			Predicted (ANSYS)		
normalized position	I = 1A	I = 2A	I = 3A	I = 1A	I = 2A	I = 3A
0.0	0.386	0.511	0.596	0.329	0.423	0.459
0.1	0.367	0.498	0.574	0.320	0.415	0.451
0.2	0.345	0.478	0.559	0.307	0.405	0.445
0.3	0.324	0.458	0.529	0.263	0.387	0.434
0.4	0.278	0.441	0.519	0.228	0.360	0.420
0.5	0.251	0.406	0.498	0.193	0.319	0.397
0.6	0.218	0.376	0.476	0.157	0.271	0.361
0.7	0.179	0.317	0.406	0.120	0.220	0.306
0.8	0.141	0.295	0.391	0.085	0.167	0.242
0.9	0.113	0.245	0.358	0.068	0.135	0.202

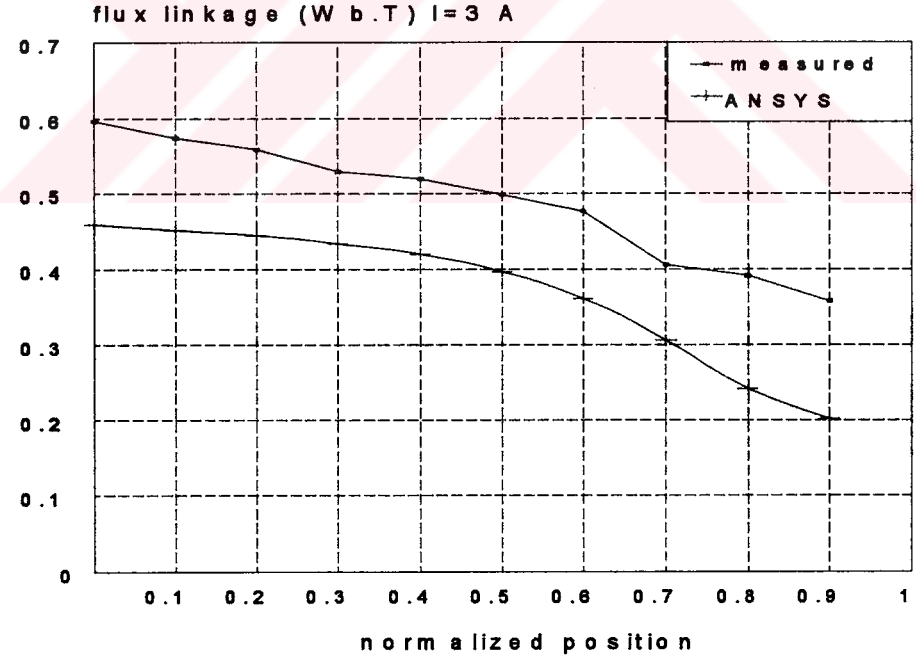


(a) I=1 A

Figure 2.37.a,b,c Measured and computed flux linkages for SR2 for different excitations



(b) I=2 A



(c) I= 3 A

Fig. 2.37 Cont'd

2.7. Conclusion

The study in this part of this thesis was aimed to familiarize with the use of professional magnetic field solution sorftware and gain the capability to use such software for design purposes of SR motors.

In the first part of this work, 'macro's are developed for the calculation of flux linkage, leakage flux and torque from the field solution. The effects of mesh distribution on solution accuracy and torque and flux linkage calculations are investigated. Rules are laid out for obtaining accurate results. Field solution software is also used for the prediction of torque-position and flux linkage- position curves of two different SR motors.

Measurements of the torque and flux linkage characteristics of SR motors are conducted. When measured data is compared with computed torque curves the agreement is found to be very good. Comparison of measured and computed flux linkage- position curves displayed that a constant discrepancy exists between measurements and predictions due to the end winding leakage which can not be taken into account in two dimensional field solutions. However, it must be noted that often the difference in the area under the curves is required in performance computations and in that case the constant end winding leakage plays no role. If necessary, the end winding leakage can be calculated using analytical and numerical techniques.

The study in this section clearly displayed that magnetic field solution technique and the routines developed for torque and flux linkage calculation are reliable and can be used to verify the performance of a srm.

CHAPTER 3

NEURAL NETWORK BASED FORCE CALCULATION OF ASYMMETRICALLY SLOTTED STRUCTURES

3.1 Introduction

For the designer, accurate prediction of torque-position curves is vitally important since it gives the information about the motor's single step and dynamic response and also maximum static and dynamic torques. On the other hand, as discussed in the previous chapters, because of the structure, torque prediction of srm is extremely tedious and needs several finite element field solutions which takes huge computation time even on a fast workstation. Since the aim is to find the optimum motor structure with minimum torque ripple, plenty of torque-position curves for different structures are required. But, using finite element analysis in an iterative optimization process is almost impossible due to the time constraint.

In view of the limitations of the finite element analysis, another method which can be used for torque curve prediction of an srm is sought. Force and permeance data which is numerically computed [16,17], per unit length for a doubly salient structure with identical stator and rotor teeth, contains information about the nonlinear nature of the problem, and is used as a data base. However, the data is computed for equal slotting for both rotor and stator sides. For the optimization process data for an unequal slotting is also required. It is shown[19], that the permeance and

force for an unequal slotting can be obtained from the data available for equal slotting. The procedure will be described in section 3.4.

Inherently, the data on hand includes only some discrete points in the design space but an accurate optimization process requires the continuous search space. In other words, a continuous function that maps the design variables on force is required. Due to its highly nonlinear nature of the problem, it is misleading to derive a mathematical function that fits the finite number of data points at hand. What remains is various numerical methods or artificial neural networks. Due to the reasons outlined later in this chapter, the latter is preferred. Throughout in this chapter, force data set production using artificial neural networks for both symmetrical and asymmetrically slotted teeth pairs is explained.

3.2 Normalization

Since produced torque of an srm highly depends on the airgap parameters of the motor such as stator and rotor tooth widths, airgap length, and rotor pole pitch, optimization should be based on these variables. Airgap parameters of an srm are shown in Fig.3.1 where single stator pole and two rotor poles are shown.

In Fig.3.1 λ_r denotes rotor pole pitch, t_r and t_s denote rotor and stator pole widths respectively, y is overlap and x is the displacement between the center lines of stator and rotor poles.

For the sake of generality all dimensions should be normalized as a fraction of rotor pole pitch. The normalization makes all procedures less complex and comparisons more meaningful.

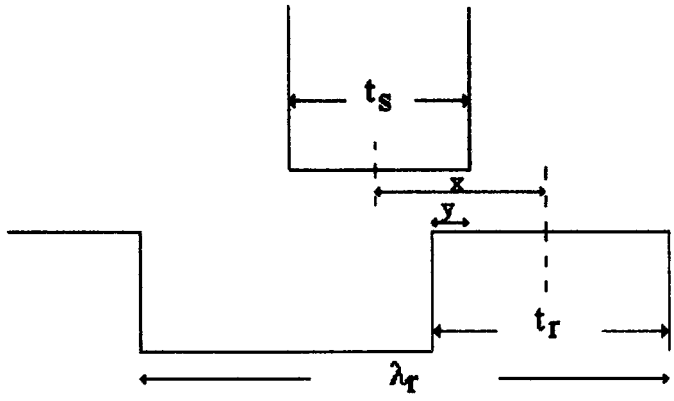


Figure 3.1 Airgap region of an srm

Since positional relationship between stator and rotor poles repeats with a period of λ_r, the position knowledge should somehow be expressed as a function of rotor pole pitch, λr. Any position can be converted to normalized position from;

$$x = x_n \cdot \frac{\lambda_r}{2} \tag{3.1}$$

where 'x' (see Fig.3.1) is the displacement and 'x_n' is normalized displacement. It is also convenient to express the pole widths as a fraction of λ_r. Normalized values of stator and rotor pole widths, t_{sn} and t_{rn} are defined as follows;

$$t_{sn} = \frac{t_s}{\lambda_r} \tag{3.2}$$

$$t_{rn} = \frac{t_r}{\lambda_r} \tag{3.3}$$

Finally, the tooth pitch is normalized with respect to g.

$$\text{pitch to gap ratio} = \frac{\lambda_r}{g} \tag{3.4}$$

3.3 Unit Doubly Salient Structure

Unit doubly salient structure (udss), whose rotor pole pitch and core length are 1m long, is the scale model of the original geometry. The original geometry and corresponding udss are shown in Fig. 3.2.

Having a general structure with unity rotor pole pitch and core length, udss is a very convenient tool as far as the analysis is concerned. Since all dimensions are converted, performance comparisons of the motors with different dimensions could be possible. All dimensions, which are in transverse plane, are obtained as scaled by $1/\lambda_r$ and in axial dimensions by $1/L_c$. So, for the coordinates following relationship may be written for the geometries shown in Fig.3.2;

$$x_o = \lambda_r x_u \tag{3.5.a}$$

$$y_o = \lambda_r y_u \tag{3.5.b}$$

$$z_o = L_c z_u \tag{3.5.c}$$

where the subscript "o" refers to the original geometry and "u" refers to the udss.

In order to carry the nonlinear nature of the actual geometry to the scaled model, the magnetic quantities B and H should be scaled by a scale factor of unity. So, mmf per pole pair and flux crossing the gap over a rotor pole pitch may be scaled as follows;

$$F_o = \lambda_r F_u \tag{3.6}$$

$$\phi_o = \lambda_r L_c \phi_u \tag{3.7}$$

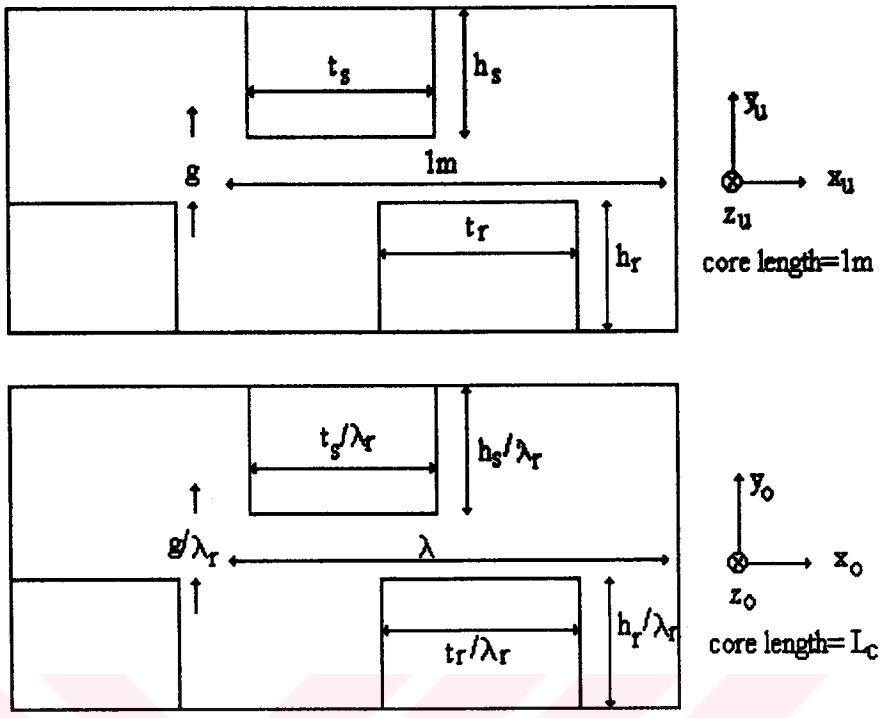


Figure 3.2. A doubly salient geometry and unit doubly salient structure

3.4 The Data

For optimization, numerically computed force and permeance data will be used as a base. The data was computed by Ertan[16,17], for per unit axial length for a doubly salient geometry with 'identical' stator and rotor teeth as a function of the following variables:

λ/g: pitch to gap ratio

t/λ: normalized tooth width

B_t: average flux density

x_n: normalized position

Throughout the computations , a geometry with three identical stator and rotor teeth pairs is considered and force and permeance is calculated for the interior teeth pair. The geometry for which data have been computed has a tooth pitch, equal to 0.0172m. That means, for 'udss' equivalent mmf and force values have to be multiplied by the constant; 1/0.0172. Typical B_t-mmf and force-mmf curves obtained are shown in Fig. 3.4. and force values are tabulated in Table 3.1

Ertan[16,17] and Besenek[7] obtained these curves for 6 different λ/g ratios (40, 70,100,150, 200,250) and three different t/λ ratios (0.3, 0.4, 0.5). For the calculations slot depth is chosen as 40g to eliminate the slot effects. Ertan showed in his work that this choice of slot depth effectively represents an infinite slot.

Normalized permeance of a given magnetic structure having identical teeth pairs with tooth width t, tooth pitch λ, airgap length g, for a given B_t and x_n is a function of normalized variables t/λ and λ/g only and defined as follows:

$$P_n = B_t \frac{(t/\lambda)}{F(\lambda/g)\mu_o} \tag{3.8}$$

The corresponding flux, ϕ and mmf, F can be found as ;

$$\phi = B_t\, t \tag{3.9}$$

(since, core length L is assumed to be 1m long)

$$F = \frac{\phi}{P} \tag{3.10}$$

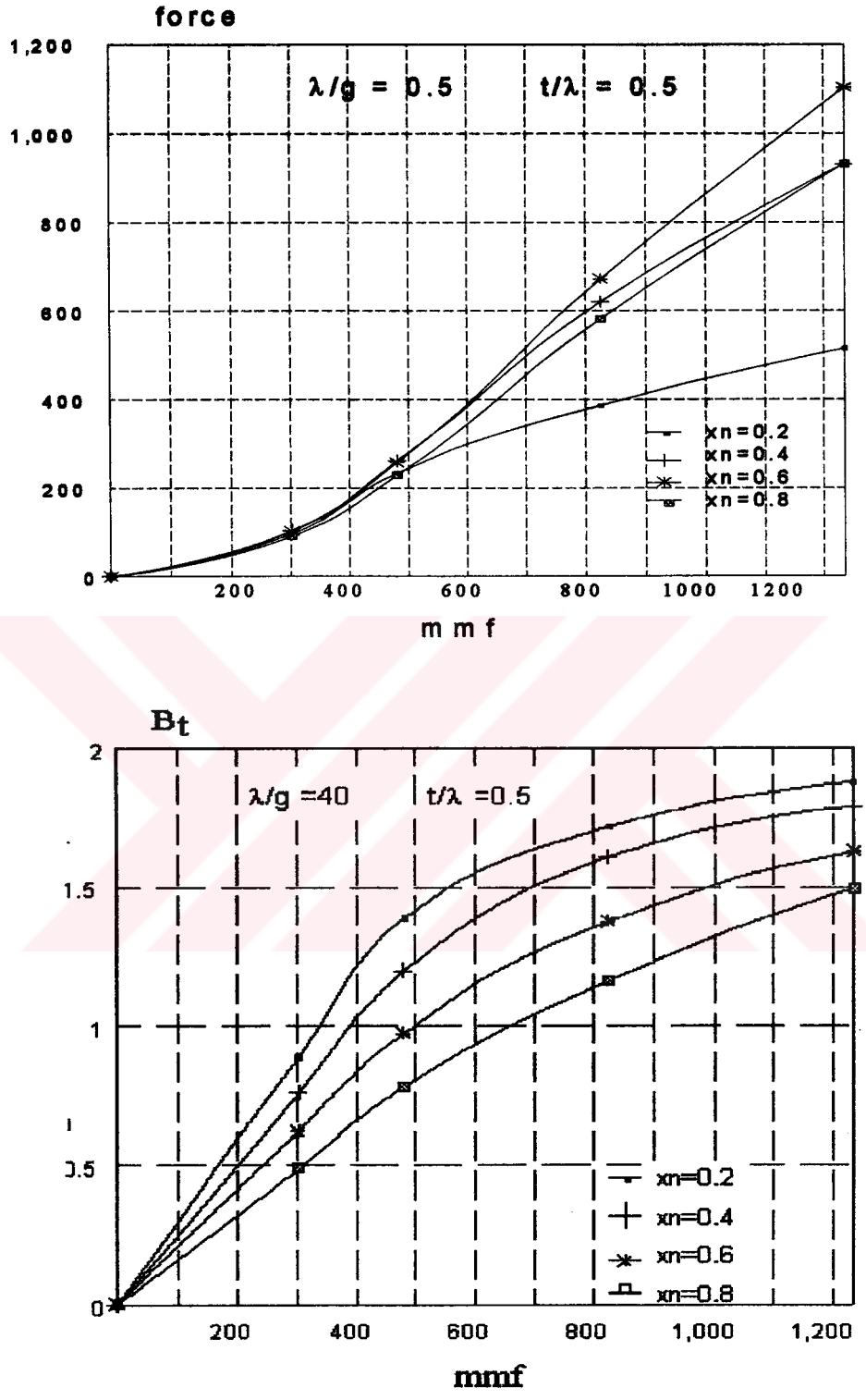


Figure 3.3 Sample force-mmf and B_t-mmf curves computed by Ertan[16,17] where tooth pitch λ_r is 0.0172 m and core length is 1m long.

Table 3.1 Normalized Force-mmf (S(Nt)-F(A)) data computed by Ertan (L_c=1m)

		********** λ/g **********											
t/λ	x_n	40		70		100		150		200		250	
****	*** *	F	S	F	S	F	S	F	S	F	S	F	S
0.3	0.2	302	99.1	175	65.6	125	51	90	40.3	70	33.3	55	25.8
		431	245	220	104	200	129	150	110	110	31	90	67.8
		825	389	470	333	350	308	250	243	160	199	160	173
		1235	490	707	458	525	499	380	414	270	323	230	284
0.3	0.4	302	110	175	69.6	125	53.1	90	41.6	70	33.3	55	26.1
		431	277	220	110	200	135	150	113	110	82.5	90	68.9
		825	723	470	453	350	364	250	270	160	209	160	180
		1235	1008	707	808	525	647	380	482	270	343	230	296
0.3	0.6	302	108	175	67	125	51.5	90	40.6	70	32.9	55	25.4
		431	275	220	165	200	131	150	113	110	80.9	90	68.3
		825	705	470	481	350	375	250	283	160	216	160	186
		1235	1210	707	837	525	619	380	461	270	327	230	280
0.3	0.8	302	20.6	175	7.9	125	4.2	90	2.1	70	1	55	0.6
		431	52.2	220	18	200	10.7	150	5.3	110	2.6	90	1.6
		825	154	470	56.6	350	32.9	250	16.2	160	8	160	5.1
		1235	334	707	128	525	74	380	37.4	270	16	230	10.6
0.4	0.2	302	99.1	175	65.7	125	51	90	41	70	33.3	55	25.8
		431	246	220	104	200	123	150	110	110	80.5	90	67.6
		825	396	470	320	350	284	250	234	160	192	160	170
		1235	533	707	432	525	454	380	400	270	311	230	278
0.4	0.4	302	107	175	63.6	125	52.6	90	41.6	70	33.7	55	26.1
		431	270	220	103	200	134	150	114	110	82.1	90	68.5
		825	632	470	443	350	358	250	266	160	205	160	176
		1235	944	707	465	525	638	380	479	270	337	230	291
0.4	0.6	302	110	175	71	125	52.8	90	41.6	70	33.3	55	26.1
		431	276	220	172	200	134	150	113	110	82.5	90	69
		825	719	470	449	350	361	250	270	160	209	160	180
		1235	1215	707	803	525	637	380	431	270	343	230	296
0.4	0.8	302	100	175	66.2	125	50.3	90	40	70	32.5	55	25.2
		431	252	220	162	200	128	150	111	110	80.3	90	67.4
		825	696	470	459	350	366	250	279	160	214	160	185
		1235	1137	707	793	525	598	380	453	270	323	230	277

Table 3.1 Cont'd

t/λ	x_n	40		70		100		150		200		250	
****	***	F	S	F	S	F	S	F	S	F	S	F	S
0.5	0.2	302	95.6	175	64.4	125	50.6	90	40.7	70	33.2	55	25.8
		431	234	220	202	200	128	150	109	110	80.2	90	67.4
		825	387	470	302	350	283	250	230	160	190	160	168
		1235	515	707	454	525	468	380	392	270	306	230	276
0.5	0.4	302	103	175	67.5	125	52.2	90	41.4	70	33.6	55	26
		431	261	220	107	200	133	150	113	110	81.7	90	68.2
		825	621	470	425	350	350	250	263	160	200	160	173
		1235	931	707	725	525	618	380	471	270	330	230	286
0.5	0.6	302	102	175	69	125	51.7	90	41.3	70	33.6	55	26
		431	258	220	167	200	131	150	113	110	81.7	90	63.3
		825	672	470	432	350	351	250	263	160	203	160	174
		1235	1104	707	767	525	622	380	471	270	334	230	289
0.5	0.8	302	90.8	175	62.7	125	49.2	90	39.6	70	32.8	55	25.6
		431	229	220	153	200	125	150	108	110	80	90	67.4
		825	583	470	400	350	333	250	255	160	201	160	175
		1235	932	707	690	525	576	380	443	270	327	230	286

3.5 Discussion on the Effect of the Asymmetrical Slotting

As seen from the normalization formulas in Sec.3.2, all design parameters are normalized with respect to rotor pole pitch. But, for all practical srm designs, number of stator and rotor teeth must be different for repeatable stepping. For different number of rotor and stator teeth, it is obvious that there is always overlap between the poles. In that case however, several problems arises;

1. Does the adjacent teeth have any effect on the performance? If so, how can this be taken into account?

2. If the data computed for symmetrical slotting is to be used for torque calculations, can it be used and is any modification needed in using this data?

This issues will be considered below;

A carefull consideration will show that a srm with even number of rotor poles and even number of stator poles excited in diametrically oposite pairs will have negligable mutual coupling between phases. If iron is assumed to be infinitely permeable, it can easily be shown that the mutual coupling between phases is indeed zero. This is true, because with equal ampere-turns in each coil, the mmf drop across the gap between one stator pole and the rotor is exactly equal to the mmf drop across the gap between the rotor and the diametrically opposite stator pole. This means that the rotor is at zero magnetic potential with respect to the stator. Therefore no leakage flux will flow between rotor and the poles belonging to the other phases.

This may also be proved by the simple analysis of the magnetic circuit of the srm as shown in Fig.3.4. In the circuit diagram, F_1 and F_2 are ampere-turns of diametrically excited coils and they are equal. R_1 and R_2 are the sum of gap, stator and rotor reluctances. R_{o1} and R_{o2} are the sum of overlapping stator teeth, rotor and gap reluctances and are equal as well. It may easily be proved from the circuit that, when back core mmf is assumed to be zero, in other words iron permeability is very high in this region, leakage flux ϕ_2 is exactly equal to zero.

$$R_1 = R_2 = R_g + R_s + R_r \tag{3.11}$$

where R_g, R_s, and R_r are gap, stator teeth, and rotor teeth reluctances respectively,

$$R_{o1} = R_{o2} = R_{og} + R_{os} + R_{or} \tag{3.12}$$

where R_{og}, R_{os}, and R_{or} are gap, neighbour stator teeth, and rotor teeth reluctances respectively. From the circuit, loop equations may be written as;

$$F_1 = R_1\phi_1 + R_2\phi_2 \quad (3.13.a)$$

$$F_2 = R_1\phi_1 + R_2\phi_2 \quad (3.13.b)$$

$$\phi_1 = F_1/R_1 \quad (3.13.c)$$

By substituting Eq.3.13.c in Eq.3.13.a, it is found that ϕ_2 is equal to zero. The discussion given here is also verified using numerical field solution technique and discussed once again in Sec.5.2.

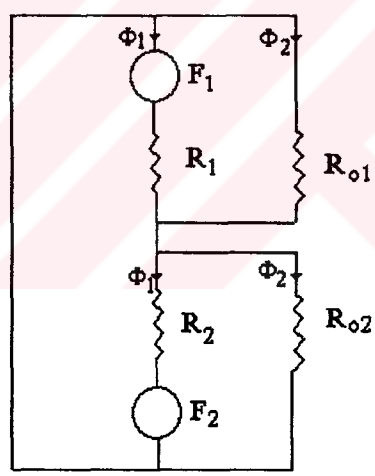


Figure 3.4 Magnetic circuit diagram of an srm when one phase is excited

The evaluation above indicates that unless the back iron mmf drop is high i.e., a significant proportion of excitation mmf drops in this regiondue to wrong design or extremely high excitation, the adjacent teeth have no role to play even if rotor teeth under the excited pole overlaps also the next or previous stator teeth.

The data available is computed by assuming that all of the adjacent teeth are under excitation. Therefore as discussed in Ref.19 and Sec.3.8 a reduction in force produced occurs when a rotor tooth is under the effect of more than one stator teeth.

When it is desired to use this data for calculation of the force for a pair of teeth in a srm, obviously the embedded effect of adjacent teeth somehow be removed for finding a correct force value. A rough correction approach is described in Ref.19 and also mentioned in Sec.3.9.2. Obviously, if an adjacent pair closer than 25g is present at the instant force is calculated, there is a reduction effect in force produced as compared to the situation and this must be compansated using Fig.3.12.

While using the data in this work, all of the dimensions are normalized with respect to λ_r. If the rotor teeth is found to approach the adjacent teeth less than 25g than λ_r is redefined as in Eq.3.25. In this manner the effect of adjacent teeth embedded in the data is removed.

3.6 Obtaining Flux vs MMF and Force vs MMF Characteristics Using Artificial Neural Networks

In order to obtain force-mmf and flux-mmf characteristics of a given geometry of switched reluctance motor at a certain operating condition using the data mentioned before, artificial neural networks are preferred to other curve fitting techniques such as cubic splines or polynomials. The neural network will be trained by the data to give force and flux of a given structure as a function of the airgap parameters, and given excitation (mmf). A brief explanation about artificial neural networks is contained in the following subsections. The advantages of using neural networks may be summarized as follows:

a) Obtaining fast results; Once the network is trained with the whole data the network gives fast response as a software module.

b) Obtaining a precise mapping; If the neural network is defined with the correct number of neurons in the hidden layers, results will be very accurate.

c) Since network is working as a function within the software, programming effort is minimized.

d) For multi-input systems, other techinques such as splines causes complexity in programing, since all two dimensional surfaces have to be defined seperately. But in the usage of neural networks, there is no limitation on the dimensions of input and output vectors

3.7 Artificial Neural Networks

> "An artificial neural network is an information or signal processing system composed of a large number of a simple processing elements, called artificial neurons or simply nodes, which are interconnected by direct links called connections and which cooperate to perform parallel distributed processing in order to solve a desired computational task."[10]

Artificial neural networks may be considered as a simplified model of a human brain because of their ability to adopt to different environment by changing their connection strenghts or structure.

3.7.1 Basic Neuron Model

The basic artificial neuron can be modelled as a nonlinear device with multi-input and output (Fig.3.4). Weighted interconnections 'w_{ij}' are also called synaptic weigths or strenghts. The cell body represents a nonlinear limiting or threshold function $\Psi(u_j)$.

This simplest model of artificial neuron sums the n weighted inputs and passes the result through a nonlinearity process according to the equation

$$y_j = \psi\left(\sum_{i=1}^{n} w_{ij} x_i + \Theta_j\right) \tag{3.14}$$

where Ψ is a limiting or threshold function, called an activation function, Θ_j is the external threshold, also called an offset or bias, w_{ji} are the synaptic weights or strenghts, x_i are the inputs (i=1,2,,,n), n is the number of inputs and y_j represents the output. A threshold value Θ_j may be introduced by adding another input x_0, equal to +1, to the system and corresponding weight w_{0j} , so the Eq.(3.14) may be written as

$$y_j = \psi\left(\sum_{i=0}^{n} w_{ij} x_i\right) \tag{3.15}$$

where $w_{0j}=\Theta_j$, $x_o=1$.

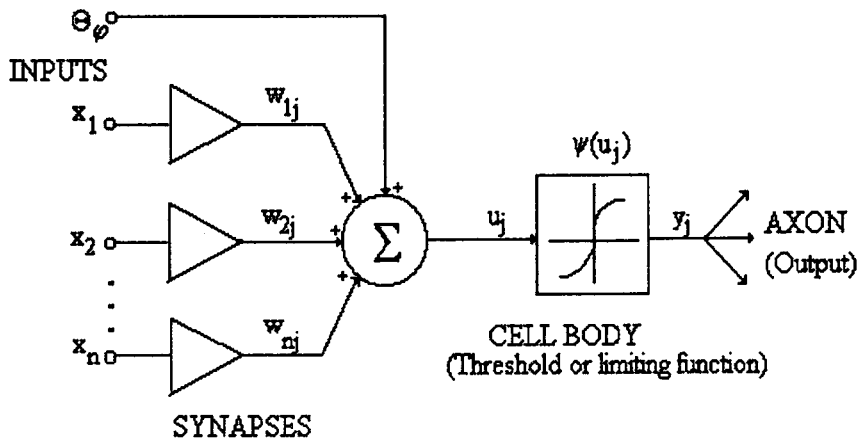


Figure 3.5 Basic neuron model

In the basic neuron model, the output signal is usually determined by a monotonically increasing sigmoid (S-shaped) function of a weighted sum of the input signals. Such a sigmoid function is mathematically defined for example as

$$y_j = \tanh \gamma u_j = \frac{1 - e^{-2\gamma u_j}}{1 + e^{-2\gamma u_j}} \qquad (3.16)$$

for a symmetrical representation or

$$y_j = \frac{1}{1 + e^{-\gamma u_j}} \qquad (3.17)$$

for an unsymmetrical unipolar representation where γ is a positive constant or variable which controls the slope of sigmoidal function. In comparison with the other nonlinearity functions sigmoidal activation function is more convinient and realistic and used in this work. These sigmoidal functions are shown in Fig.3.6.a and Fig.3.6.b respectively.

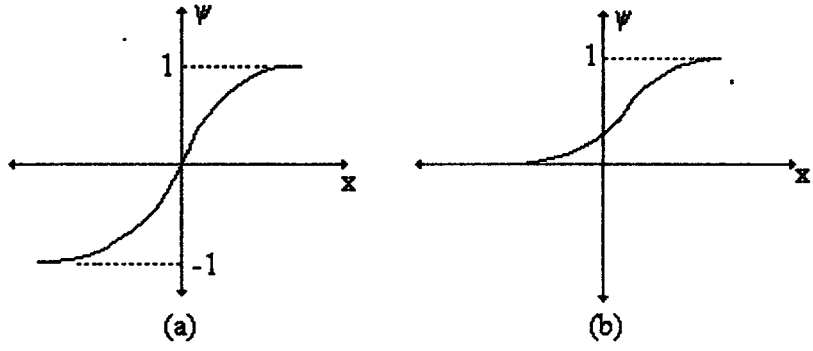


Figure 3.6 Symmetrical (a) and unsymmetrical (b) sigmoidal activation functions

3.7.2 Architecture of Three-Layer Perceptron

In this study three-layer feedworward network is used as described in the following sections. Standard multilayer networks are a class of feedforward neural networks with neurons in the layers. All neurons in the layers are connected to all neurons in the adjacent layers through uni-directional links. These links are represented by synaptic weights. The snapthic weights act as signal multipliers on the corresponding links (interconnections).

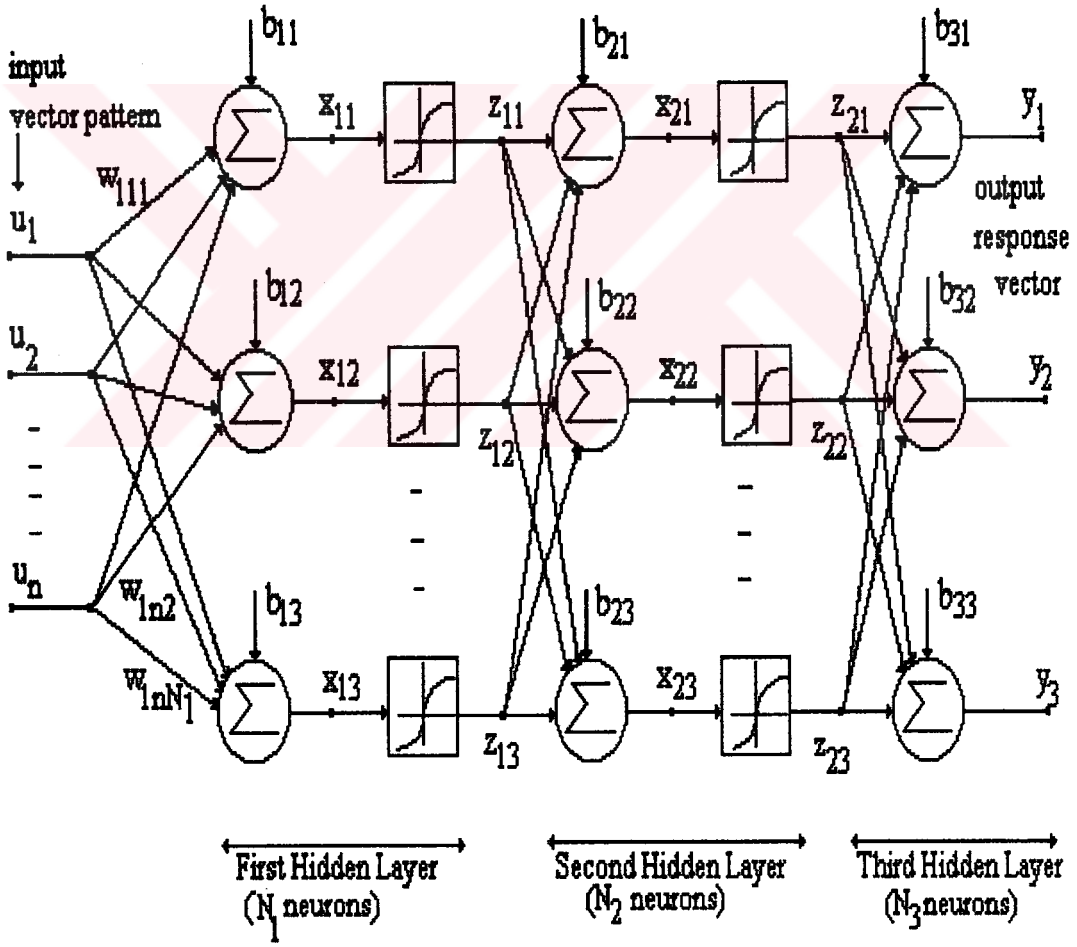


Figure 3.7 Tree layer feedforward neural network

In Fig.3.7 a three layer perceptron shown. The neurons are grouped in sequentally connected layers; each layer is numbered 1,2,3. The neurons of layer one are called the first hidden layer and the last layer is the output layer where the response of the network comes through. The neuron layers between the input and output patterns are called 'hidden layers'. Theoretically, there is no limitations on the number of hidden layer, but in practice there will be one or two hidden layers. Each neuron is connected to all neurons of the two adjacent layers and to no other neurons. Note that connections within a layer or from higher to lower layers are not permitted. The arrows indicate the flow of information. Generally, the multilayer perceptron has a different number of neurons and different synaptic weights for different layers. The neurons usually take activity in the normalized range from +1 to -1 (in this study this range is preferred).

Once the synaptic weights are determined through optimization their values are kept constant in using the network predictions. These values determine the network behaviour and its capability to correctly process (map) the input data. In order to obtain the required network behaviour the values of the synaptic weights must be properly computed. Such a computation is called 'learning' or 'training' process. During the training process information is also propagated back through the network and it is used to update the synaphtic weights succesfully first in the output layer, second in the hidden layer, and last in the first hidden layer. The training process and 'batch learning algorithm' will be described in the following subsection.

3.7.3 Learning Algorithms

Learning algorithms may be classified into two groups as on-line and batch learning. In the batch learning algorithm first all learning samples are accumulated, than weights are updated according to the error functional for the whole set. But in on-line approach determination of weights is accomplished by updating for each

individual learning sample iteratively. On the other hand, batch learning approach takes longer time than on-line procedure. In this study batch back propagation algorithm is preferred. The reasons for the choice of batch learning algorithm may be summarized as follows:

a) All learning examples are already available and no temporal adoptation is required.

b) A high precision mapping may be obtained

c) "Batch learning provides a better estimate of the gradient components and avoids a mutual interference of the weight changes caused by different patterns"[10].

3.7.4 Batch Learning Algorithm with Unconstrained Optimization

Batch learning is essentially a mean square error minimization process. As cited before, a feedforward neural network works as a nonlinear function that maps a given input vector on an output vector with predetermined dimensions. The training pattern consist of a set of input vectors and corresponding output vectors. Training of the network means choosing synaptic weights such that the array of output vector that the neural network computes for the array of training input vectors fits the array of training output vector as much as possible. Theoretically, the best measure that represents the degree of fit is the mean square error measure defined as

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(y_{ij} - yd_{ij}\right)^2 \qquad (3.18)$$

where y is the computed output, yd is the desired output, n is the number of the training samples, and m is the dimension of the output vector.

The feedforward neural network may be considered as a chain of mathemathical transformations where at each layer of the network the input vector is transformed first linearly and then nonlinearly. The output of each layer is the input of the consequtive layer. So it is possible to express the error at each level as a function of the synaptic weights and threshold values corresponding to that level, analitically. By this property, the error at the output level may be propagated backwards to compute the errors implied in the preceeding levels. That is the reason why this algorithm is called as the back propogation algorithm. Thanks to this ability of analytical representation, it is possible to calculate the effect of individual changes in snaptic weights and thresholds on the performance measure, mean square error. In other words, it is possible to express the partial derivatives of the mean sqare error with respect to the synaptic weights and the treshold values analytically. Once these gradient vectors are obtained what remains is the employment of an optimization algorithm to minimize the mean square error with respect to the synaptic weights and treshold values.

The method choosen is the steepest descent algorithm. Essentially, the algorithm, beginning from an initial point, scans the error surface such that it iteratively moves along the direction where the slope of the error surface is steepest. In the algorithm one dimensional search is carried out to find the optimum amount of advance in the direction of the gradient. First, given the initial point ($w_{nij}(0)$), the gradient vector (∇E) at that point, two predetermined search parameters α_0 ,α_1 ($K_0/|\nabla E|$and $K_1/|\nabla E|$ where K_0 and K_1 are arbitrary parameters choosen heuristically) and weights are updated by;

$$w_{nij}(k+1) = w_{nij}(k) + \alpha\left(\frac{\partial E}{\partial w_{nij}(k)} + \beta(w_{nij}(k) - w_{nij}(k-1))\right) \qquad (3.19)$$

(where β is momentum constant defined by the user to control the rate of convergence) by taking $\alpha=\alpha_0$ and $\alpha=\alpha_1$ seperately, and the corresponding error is

calculated for both. The error is assumed to be a parabolical function of α within the interval $[0,\alpha_1]$ which passes through the error points calculated at $\alpha=0$, $\alpha=\alpha_0$ and $\alpha=\alpha_1$, and a parabolic fit is carried out accordingly. The optimum search parameter α^* is choosen as the value that minimizes the fitted parabole and the weights are updated according to Eq.3.19. This procedure is iterated until convergence is maintained. Throughout the iterations the results are monitored and to ease convergence the algorithm is intervened by changing the momentum constant β when necessary. The batch learning program, which is contained in Appendix A, is written on C programming language and the program steps are described as follows:

ALGORITHM STEPS

1. Define arbitrary weights w_{nij} where

   n: layer number

   i: neuron number of the corresponding layer

   j: neuron number of the consequtive layer

2. Calculate states:

$$\overline{X_1} = \begin{bmatrix} x_{11} \\ x_{12} \\ . \\ . \\ x_{1N_1} \end{bmatrix} = \begin{bmatrix} w_{111} w_{121} \cdots\cdots w_{1n1} \\ w_{112} \cdots\cdots\cdots\cdots \\ . \\ . \\ w_{11N_1} \cdots\cdots\cdots w_{1nN_1} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ u_n \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ . \\ . \\ b_{1N_1} \end{bmatrix} \qquad (3.20)$$

where vector $\bar{u}$ is input pattern and, $\bar{b}$ is the treshold vector of first layer.

$$\overline{Z_1} = \begin{bmatrix} z_{11} \\ z_{12} \\ . \\ . \\ z_{1N_1} \end{bmatrix} = \psi\left(\overline{X_1}\right) = \tanh \gamma \overline{X_1} \qquad (3.21)$$

In the same manner calculate vectors, $\overline{X_2}, \overline{Z_2}$ and output response vector $\bar{y}$ for every input patterns,

3. Calculate error function (Eq.3.18),

4. Calculate gradient vector of error function. (all gradient components are calculated analitically using 'chain rule'),

$$\nabla \overline{E} = \left[\frac{\partial E}{\partial w_{111}} \frac{\partial E}{\partial w_{112}} \cdots \frac{\partial E}{\partial w_{211}} \cdots \frac{\partial E}{\partial w_{311}} \cdots \frac{\partial E}{\partial b_{111}} \cdots \right]^T \qquad (3.22)$$

5. Calculate search parameter α^* as described before,

6. Calculate new synaptic weight using Eq.3.19,

7. If there is problem related with the convergence, update momentum constant. One of the facility of the program is the random momentum constant (β) generator, if random momentum constant updating is preferred, this step is omitted.

8. Check convergence and go to step 2.

3.8 Networks Trained for the Computation of Force vs MMF and Flux vs MMF Curves of Symmetrically Slottted Structures

From the above mentioned data obtained by Ertan[16,17], neural networks are trained to obtain force vs. mmf and flux vs. mmf curves for symmetrically slotted structures.

3.8.1 The Network (NN1) for Computation of Force vs MMF Curves

The first network computes force for a given geometry and given mmf value. Inputs of the network are λ/g, t/λ, x_n, and mmf (F) , and the output of the network is force (S). Training data computed by Ertan is defined for a geometry where tooth pitch λ is equal to 0.0172 m. Force (output) and mmf (input) values may be easily converted to udss by multiplying with 1/0.0172. Before training, all input and output data is normalized into the range [0,1]. This have to be pointed out that all inputs and outputs of the network have to be normalized for guaranteing convergence. This is done by dividing λ/g to 250, mmf and force values to 1500 (maximum in Ertan`s data). t/λ and, x_n are not needed to be normalized since they are within the desired range. The training set is contained in Appendix A and the specifications of this network are summarized as;

Inputs: λ/g, t/λ, x_n, mmf(F)
Output: Force (S)
Number of neurons in the first hidden layer: 10
Number of neurons in the seccond hidden layer: 10
Number of data points in the training set : 500

Unfortunately, there is no theoretically robust method concerning the choice of number of neurons in each hidden layer for a given problem. There exists an inherent trade-off between accuracy and generalization which are functions of the number of neurons in the hidden layers. That is, the smaller is the number of hidden layer neurons, the higher is the generalization and vice versa. Additionally, larger number of hidden layer neurons implies larger computational time until convergence is reached. These are the reasons why a large number of hidden layer neurons is not always the best alternative. So, the best is making the choice heuristically by trial and error method. Beginning with a low number of hidden layer neurons, the network is trained continously monitoring the convergence rate and if this rate is very low, the training process is terminated. Another training session with a larger number of hidden layer neurons is initiated from the scratch still monitoring the rate of convergence and this process is repeated until an appropriate convergence rate is observed. After this point the number of hidden layer neurons is not updated again and the network is left to converge until the overall mean squared error falls below 10^{-2}. In order to observe the neural network performance, force-mmf curves similar to Fig.3.8 are plotted for several combinations of λ/g, t/λ, x_n values. As seen on the representative example Fig.3.8, force-mmf relation is plotted simultaneously for the training set values (computed) and neural network output, on the same graph.

During the training, due to insufficiency of data, for some intervals (with very high or very low values of mmf) poor fits were observed and to overcome this problem new data points obtained (interpolated or extrapolated) from force-mmf graphs (for the training set values) and from the studies of Yagan[38], were added to the training set.

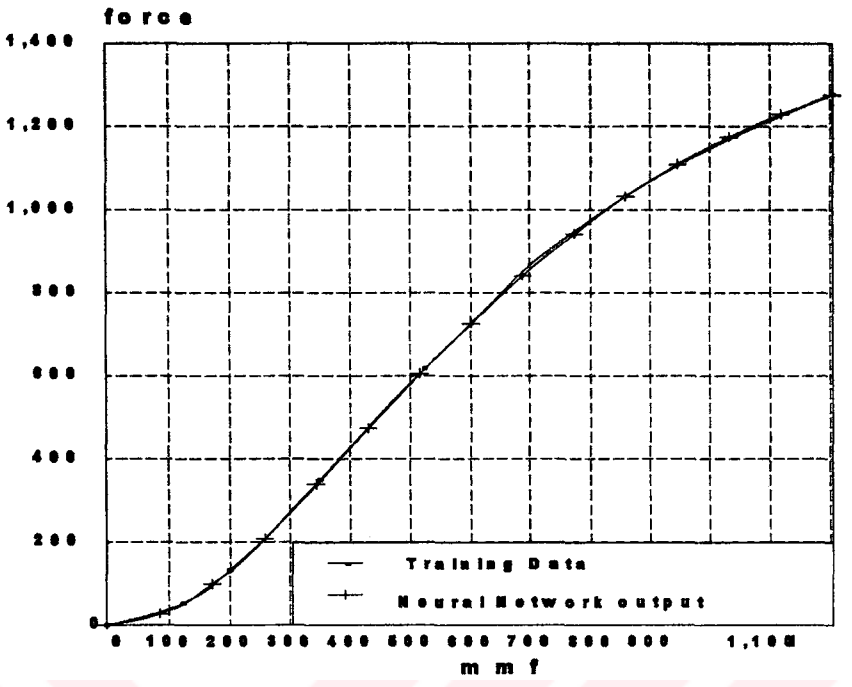


Figure 3.8 Force-mmf curves calculated using neural network in comparison with the base data (λ/g=100, t/λ=0.5, x_n=0.4, λ_r=0.0172 and L_c=1m) (A representative example)

3.8.2 The Network (NN2) for Computation of Flux vs MMF Curves

The second network is trained to find the relation of mmf values with magnetic flux density B_t. Its inputs are λ/g, t/λ, x_n, and B_t and output is mmf. It is trained for obtaining force for unequal teeth pairs, as described in the following sections. The network specifications are:

Inputs: λ/g, t/λ, x_n, B_t

Output: mmf (F)

Number of neurons in the first hidden layer: 10

Number of neurons in the seccond hidden layer: 10

Number of data points in the training set :880 (Appendix A)

New data points are also generated (by interpolation and extrapolation) and added to the original set to overcome the problem of poor fitting. This network is also divided into two network modules to guarantee convergence. Both networks were trained until the total cost is equal to 10^{-2}. First module is trained for the input values of λ/g equal to 40-70-100, and second module is trained for the λ/g values 100-150-200-250. Representative sample curves may be seen in Fig.3.9.

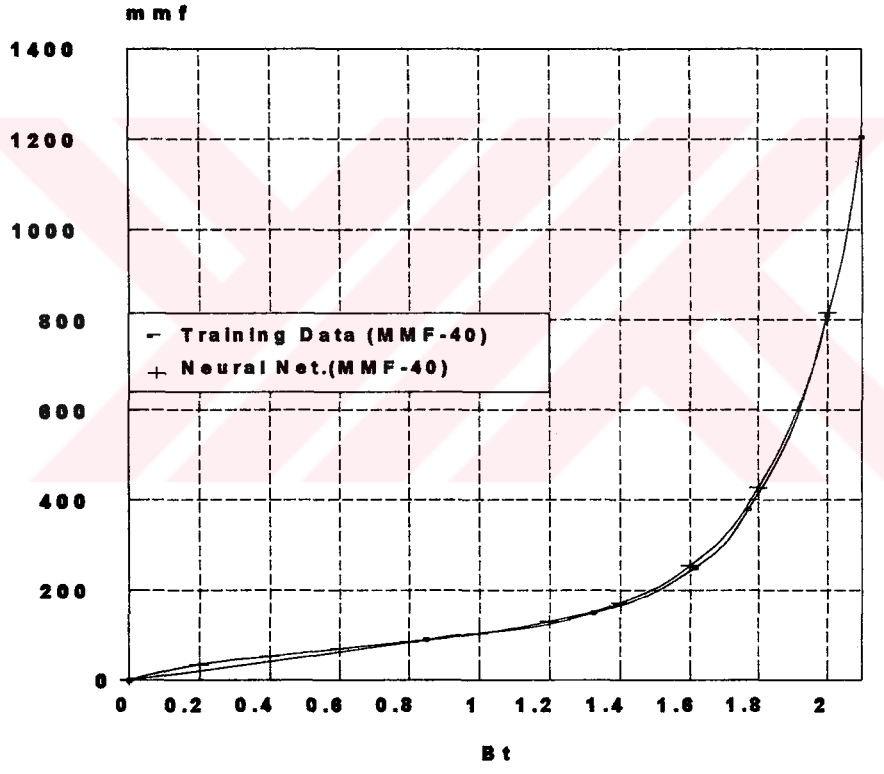


Figure 3.9 Force-mmf curves calculated using neural network in comparison with the base data (λ/g=150, t/λ=0.4, x_n=0.2 and λ_r=0.0172 and L_c=1m) (A representative example)

3.9 Normalized Force Calculation for Asymmetrically Slotted Structures

3.9.1 Theory

For obtaining force values of asymmetrical slotted structures, a method developed by Ertan[19] is used. The method may be explained by the help of an asymmetrical teeth pair shown in Fig.3.10.

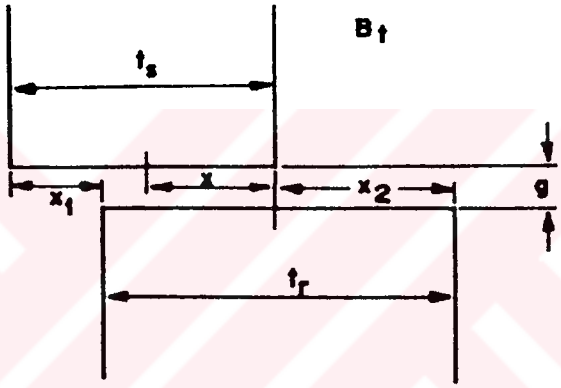


Figure 3.10 An asymmetrical teeth pair

Normalized force and permeance may be calculated by using two different geometries having identical teeth pairs shown in Fig.3.11. According to the method, symmetric geometries having equal stator and rotor teeth widths t_s and t_r are considered. For the flux of both geometries to be equal, equation below should hold.

$$\text{teeth pair flux} = \phi = B_{ts} . A_s = B_{tr} . A_r \qquad (3.23)$$

where A_s and A_r are stator and rotor teeth crossectional areas and B_{ts} and B_{tr} are average flux density values of two geometries. It is clear that , if average flux densitiy of geometry-a is B_t, average flux density of geometry-b would be $B_t . t_s/t_r$. For this reason MMF values to produce these flux densities are also different. From these geometries force and permeance are calculated.

$$S = (S_a + S_b) / 2 \tag{3.24}$$

$$P_x = 2. P_{na} . P_{nb} / (P_{na} + P_{nb}) \tag{3.25}$$

Using the permeance value of asymmetric structure, mmf value that can produce teeth pair flux ϕ , may be calculated as:

$$F = \phi / P_x \tag{3.26}$$

From this formula, resultant mmf (F) can be determined in terms of symmetrical geometry mmf values as :

$$F = (F_a + F_b) / 2 \tag{3.27}$$

For the approach to be valid, distances d_1 and d_2 should be at least 25g. The procedure mentioned below will be applied if these distances are lower .

As can be seen clearly from Fig.3.11, the geometry-a has stator and rotor tooth width of t_s , where geometry-b has t_r and normalized displacement is $x_n = 2.x / \lambda$. The approximation below should be utilized, if distance to the next pair of the moving rotor teeth is lower than 25g in the actual geometry.

$$\lambda_a = \text{Max} (\lambda , t_s + x_1 + 25g) \tag{3.28}$$

$$\lambda_b = \text{Max}\ (\ \lambda\ ,\ t_r + x_2\ \ +25g\ ) \qquad (3.29)$$

Therefore, in the calculations, λ/g and t/λ values for geometries (a) and (b) will be different if the distances d_1 and d_2 are lower than 25g. So the effects of neighbouring teeth will be considered as an important issue.

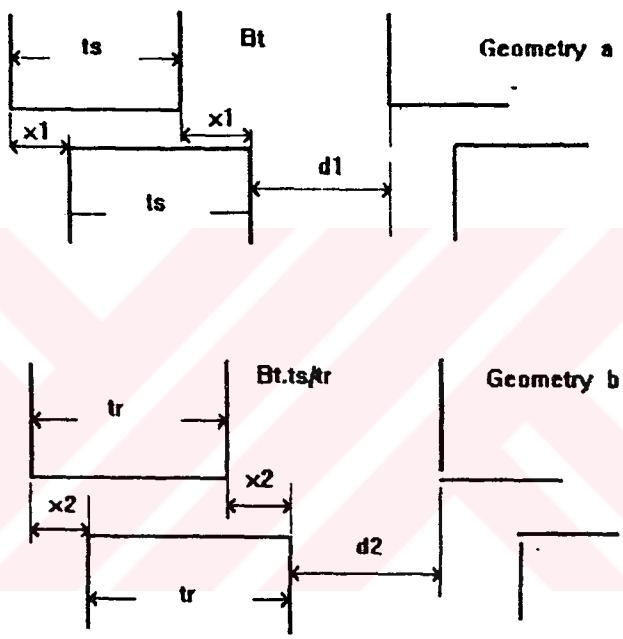


Figure 3.11 Two geometries with identical teeth pairs

3.9.2 Force Correction Factor

Altough it is not needed here, if necessary to reduce error in calculated forces from the data available a correction factor could be used. In other words if it was necessary to use the data while the distance to the next teeth was less than 25g a reduction factor due to this, could be read from Fig.3.12. Clearly if in the real

geometry such a reduction is not present, the force found from the data must be increased this much.

The percentage of reduction which is independent of position, was calculated by Ertan[19] for various flux density values and summarized in Fig.3.12. In the form of two curves one for flux density greater than 1.1 Tesla and the other for flux density lower than 0.5 T. For the flux density values in between the two extremes linear interpolation may be used. Table 3.2 shows the data for the curves.

Table 3.2 Force correction factors

d/g	reduction Bts>1.1T	reduction Bts<0.5T
25.7	0	0
13.6	6.4	2.7
10	10.6	4.4

These data points are fitted to polynomials using MATLAB for B_{ts}>1.1T and B_{ts}<0.5 T, and values within these range are found using linear interpolation.

The polynomials calculated are as follows.

For B_{ts}>1.1T

$$P1(x) = -0.0021x^3+0.1414x^2-3.6153x+34.6808 \quad (3.30)$$

For B_{ts}<0.5T

$$P_2(x) =0.0157x^2-0.8381x+11.1889 \quad (3.31)$$

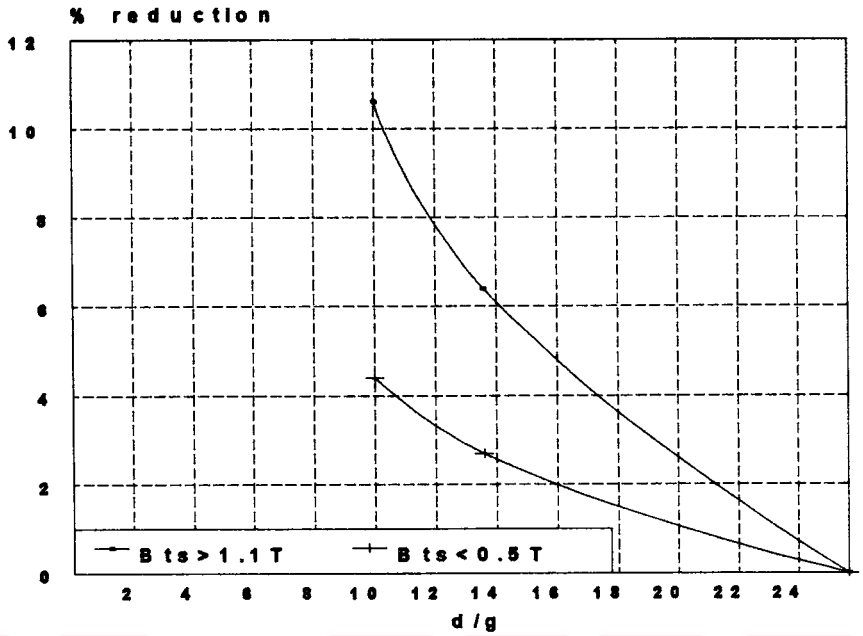


Figure 3.12 Force correction polynomials

3.9.3 Algorithm for Force Calculation of Asymmetrically Slotted Structures

The algorithm for data set production for asymmetrically slotted structures is written on C programming language and is contained in Appendix A. Basic framework of the algorithm is as follows:

STEP 1) For all the combinations of t_s/λ, t_r/λ, x_n, and B_{ts} do step 2 to step 8

*t_s/λ - 0.3, 0.4, 0.5

*t_r/λ - 0.3, 0.4, 0.5

*λ/g - 40, 70, 100, 150, 200, 250

*B_t - 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2

STEP 2) For λ=0.0172 (Calculation is based on Ertan's data) calculate the following parameters:

* $t_s = (t_s/\lambda).\lambda$
* $t_r = (t_r/\lambda).\lambda$
* $g = \lambda/(\lambda/g)$
* $x = (x_n.\lambda)/2$
* $d=\lambda - (t_s/2 + t_r/2 + x)$

STEP 3) For GEOMETRY-A calculate:

* λ_a=MAX(λ , t_s+x+25.g)
* $\lambda/g=\lambda_a/g$
* $t_s/\lambda=t_s/\lambda_a$
* $x_n=2.x/\lambda_a$
* using NN2 (neural network explained in section 3.7.2) find mmf for geometry-a (F_a)

  F_a=NN2(λ/g, t_s/λ, x_n,B_{ts})
* using NN1 (neural network explained in section 3.7.1) find corresponding force.

  S_a=NN1(λ/g, t_s/λ, x_n, F_a)

STEP 4) For GEOMETRY-B calculate:

* $B_{tr} = B_{ts}.t_s/t_r$
* λ_b = MAX(λ , t_r+x+25.g)
* $\lambda/g=\lambda_b/g$
* $t_r/\lambda=t_r/\lambda_b$
* $x_n=2.x/\lambda_b$
* using NN2 (neural network explained in section 3.7.2) find mmf for

geometry-b (F_b)

F_b=NN2(λ/g, t/λ, x_n,B_{tr})

* using NN1 (neural network explained in section 3.7.1) find corresponding force.

S_b=NN1(λ/g, t_r/λ, x_n, F_b)

STEP 5) Calculate mmf and force values for asymmetrical teeth pair:

* $F = (F_a+F_b)/2$
* $S = (S_a+S_b)/2$

STEP 6) Calculate correction factor (CF) if d/g<25:

This procedure is omitted here. However, for the sake of generality its description is given.

* if Bts>1.1 T use polynomial 1 (P1)

  CF= P1(d/g)

* if Bts<0.5 use polynomial 2 (P2)

  CF= P2 (d/g)

* if 0.5<Bts<1.1

  CF1 = P1(d/g)

  CF2 = P2 (d/g)

  CF= CF2+(CF1-CF2)/(0.6/(Bts-0.5))

STEP 7) If d/g <25

* S = S(100±CF)/100

STEP 8) Write the results to an output file

3.9.4 Training Force Data of Asymmetrically and Symmetrically Slotted Structures

From the above mentioned program, force data points for asymmetrically slotted structures are obtained. Sample force data curves obtained for asymmetrically slotted structures are shown in Fig.3.13. These data points and force data for symmetrically slotted structures had to be trained together to be used in optimization. By using the weights found from this work, a program is used to obtain static torque characteristics of any given structure, and this program is contained in Appendix B. The specifications of this network are summarized as follows:

Inputs: λ/g, t_s/λ, t_r/λ, x_n, mmf(F), Output: Force (S)

Number of neurons in the first hidden layer: 20

Number of neurons in the seccond hidden layer: 10

Number of data points in the training set : 2037

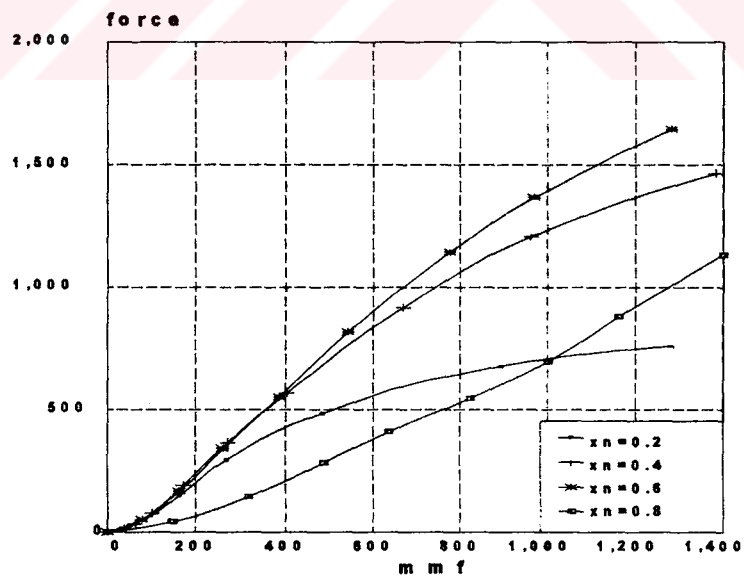


Figure 3.13 Force curves of asymmetrically slotted structures obtained from the algorithm in section 3.8.3 (λ/g=200, t_s/λ=0.4, t_r/λ=0.3, λ=0.0172m and L=1m)

3.10 Verification of the Results on Two Different Test Motors (SR1, SR2)

Two test motors where torque position curves were measured as explained in Chapter 2, are used to verify the accuracy of the obtained curves. The results are given below:

A) SR1 MOTOR:

(First the quantities t_s/λ_r, t_r/λ_r, λ_r/g have to be determined)

Outer diameter of the rotor: 70 mm

Number of rotor poles: 6

Stator tooth width (t_s): 8.5 mm

Rotor tooth width(t_r) : 10 mm

Airgap length (g): 0.2 mm

Number of turns (N): 300

Core length (L_c): 91.5 mm

Current : 4 A

λ_r = (2*Π*35) / 6 = 36.652 mm = 0.0366 m

t_s/λ_r = 8.5 /36.652 = 0.232

t_s/λ_r = 8.5 /36.652 = 0.273

λ_r/g = 36.652/0.2 = 183.26

mmf = I*N = 4*300 = 1200 AT

mmf converted to UDSS is mmf/λ_r =1200/0.0366 = 32786 AT

Since the output of the neural network is force, it have to be converted to torque which is proportional to force

$$T = 2.S.\, d_i\, .\lambda_r.L_c \qquad (3.32)$$

(where d_i is the radius of the rotor)

The results for 4A current excitation are shown in Table 3.3 and Fig.3.14. For this motor, t_s/λ_r and t_r/λ_r values are 0.232 and 0.273 respectively and these values are out of the neural network training range (0.3 , 0.4, 0.5). In addition, BH characteristic of this motor is different from the BH characteristic used for the production of the force data. The error at normalized position 0.6 may be explained with these reasons.

Table 3.3 Measured[7] and calculated torque (Nm) for SR1

x_n	Measured	Calculated
0.1	3.3	4.74
0.2	8	8.11
0.3	10.6	9.79
0.4	11.2	10.52
0.5	9.1	11.19
0.6	3.0	9.37

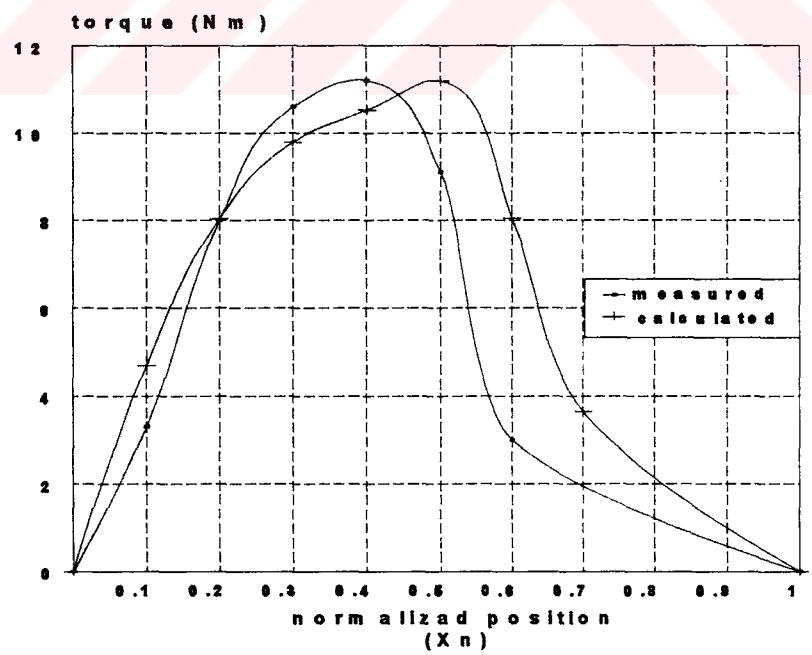


Figure 3.14 Measured[7] and calculated torque-position curves of SR1

B) SR2 MOTOR:

Outer diameter of the rotor: 38.6 mm

Number of rotor poles: 6

Stator tooth width (t_s): 8.2 mm

Rotor tooth width(t_r) : 8.2 mm

Airgap length (g): 0.255 mm

Number of turns (N): 322

Core length (L_c): 40 mm

Current :1,2,3 A

Using the same method necessary quantities may be calculated as follows:

λ_r = (2*Π*19.3) / 6 = 20.21 mm = 0.02021 m

t_s/λ_r = 8.2 /20.21 = 0.406

t_s/λ_r = 8.2 /20.21 = 0.406

λ_r/g = 20.21/0.255 = 79.25 (The results are shown in Table 3.4 and in Fig.3.15)

Table 3.4 Measured and calculated torque(Nm) values for SR2

X_n	I=3A measured	I=3A calculated	I=2A measured	I=2A calculated	I=1A measured	I=1A calculated
0.1	0.61	0.48	0.52	0.4	0.36	0.2
0.2	0.92	0.936	0.72	0.714	0.4	0.29
0.3	1.2	1.31	0.88	0.91	0.41	0.35
0.4	1.5	1.56	1.06	1.04	0.43	0.38
0.5	1.75	1.73	1.16	1.12	0.49	0.39
0.6	1.88	1.77	1.14	1.13	0.45	0.38
0.7	1.85	1.68	1.12	1.14	0.45	0.379
0.8	1.71	1.38	0.91	0.84	0.37	0.27
0.9	0.58	0.23	0.36	0.15	0.2	0.1

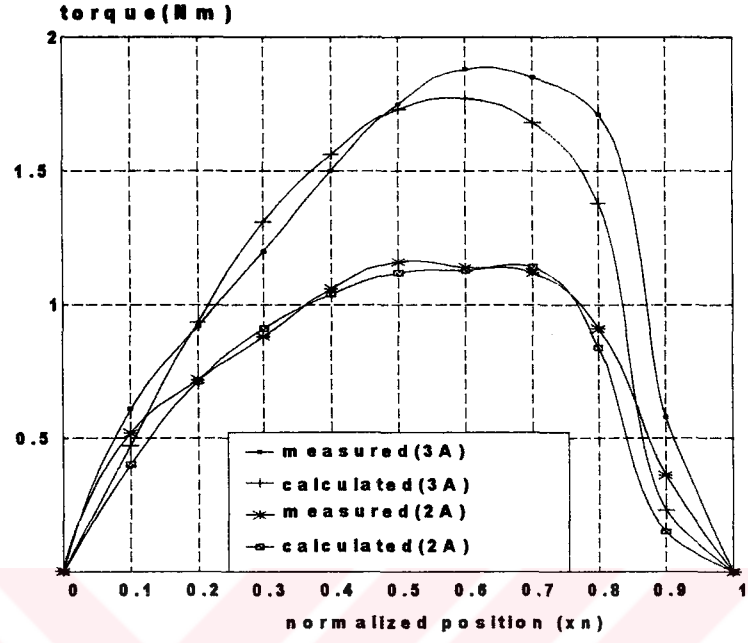


Figure 3.15 Measured and calculated torque-position curves for SR2

From the curves, it can be observed that the shape and magnitude of the torque curve is well predicted. This result is very satisfactory in view of the facts that the B-H curve of the material used for the test motor is slightly different than the one used for obtaining the data, and uncertanities exist in the airgap size due to the difficulty of measurement (as explained in Chapter 2). On the other hand, force calculations of these two motors, back core mmf drop is assumed to be zero. From the finite element analisis of the motors, it is found that back-core mmf drop is negligibly small and this assumption is supported.

CHAPTER 4

FORMULATION OF THE OPTIMIZATION PROBLEM

4.1 Introduction

As its name implies, the purpose of this thesis is to find optimum parameters for switched reluctance motors that minimize torque ripple. Essentially, the thesis is an optimization study and all the effort up to now is aimed to present the original problem in such a way that it is possible to formulate it in a standard constraint optimization problem.

Throughout this chapter, the processes pertaining to formulation of the problem in the form of a constraint minimization problem and the solution technique will be explained in detail. In the last part of the chapter, optimum parameters found for different excitation levels are also presented.

4.2 Constrained Optimization Problem in General

In its general form, a typical constrained optimization problem can be formulated as;

Minimize $f(\bar{x})$

Subject to $p_i(\bar{x}) = a_i$ for $i=1,.....,m_1 < n$

$$q_i(\bar{x}) \leq b_i \qquad \text{for } i=1,.....,m_2$$

$$\bar{x} \in X \subset R$$

$$c_k \leq x_k \leq d_k \qquad \text{(Bounded variables) } k=1,.....,n$$

where $f(\bar{x})$ is the function to be minimized or the objective function. Left and right hand side of inequality constraints are represented by functions $q_i(\bar{x})$, and constant parameters b_i respectively. Similarly, left and right hand side of equality constraints are represented by functions $p_i(\bar{x})$ and constant parameters a_i. Constants m_1, m_2 and n are number of equality constraints, number of inequality constraints and dimension of the vector $\bar{x}$ respectively. The constraints ensure that the system satisfies a set of specified requirements. $\bar{x}$ is a vector of independent optimization variables in terms of which the objective function and constraint functions can be computed. In other words, the constraints determine a subset of the domain (feasible region) defined for the independent optimization variables $(\bar{x})$. The eventual aim of a constraint minimization problem is to choose the optimal vector $\bar{x}^*$ from the feasible region enforced by the constraints that minimizes the function $f(\bar{x})$. For the solution of the problem represented in this form several methods exist (Lagrangian methods, exact penalty method, etc.). In this study, Augmented Lagrangian optimization technique is preferred

4.3 Augmented Lagrangian Method

For an unconstrained minimization problem there are several solution methods. But since in a constraint minimization problem the objective function is forced to reach its minimum within a region determined by the constraints these solution methods can not be directly applied. The main idea of constraint minimization is to convert

the problem with constraints into an unconstrained form where the constraints are implicitely imposed. The two well known methods to serve this aim is the Lagrangian method and the penalty method.

Here in this study, Augmented Lagrangian method which is the combination of the Lagrangian and Penalty Methods is used. These methods may be explained as follows;

In the Lagrangian method, the terms that the difference of the left hand side and the right hand side of equality and unequality constraints are added to the objective function after being multiplied with new variables called Lagrangian multipliers. This new function is called the "Lagrangian function" (Eq.4.1) and minimization of this function with some unconstrained minimization method gives the solution for the original constrained minimization problem.

$$L(\bar{x},\bar{\alpha},\bar{\beta}) = f(\bar{x}) + \sum_{i=1}^{m1}\alpha_i(a_i - p_i) + \sum_{j=1}^{m2}\beta_j(b_j - q_j) \qquad (4.1)$$

Penalty method on the other hand may be described as follows;

> The basic idea in penalty methods is to eliminate some or all of the constraints and add to the objective function a penalty term which prescribes a high cost to infeasible points. Associated with these methods is a parameter w, which determines the severity of the penalty and as a consequence the extend to which the resulting unconstrained problem approximates the original constrained problem. As w takes higher values, the approximation becomes increasingly accurate[8].

In view of these considerations the Augmented Lagrangian function with Lagrangian and Penalty terms may be defined as;

$$L_a(\bar{x},\bar{\alpha},\bar{\beta}) = f(\bar{x}) + \bar{\alpha}^T(\bar{a}-\bar{p}) + \bar{\beta}^T(\bar{b}-\bar{q}) - w_1p_1 - w_2p_2 - w_3p_3 \qquad (4.2)$$

where each penalty weight $w_i > 0$ and, α^T and β^T are the vectors of Lagrangian multiplier for equality and unequality constraints respectively and;

$$p_1 = \sum_{i=1}^{ml}(a_i - p_i)^2 \qquad (4.3)$$

$$p_2 = \sum_{j\in C_a}(b_j - q_j)^2, \qquad C_a = \{\ j:\ \beta_j > 0\} \qquad (4.4)$$

$$p_3 = \sum_{j\in C_a}(b_j - q_j)^2, \qquad C_b = \{\ j:\ \beta_j = 0 \text{ and } q_j \geq b_j\} \qquad (4.5)$$

where $p_i = p_i(x)$ (i=1,2....,m_1) and $q_j = q_j(x)$ (j=1,2....,m_2)

Gradient of the augmented Lagrangian function is :

$$\nabla L_a = \nabla f(x) + \sum_{i=1}^{ml} \alpha_i^+ \nabla p_i + \sum_{j\in C_a} \beta_j^+ \nabla q_j + \sum_{j\in C_b} 2w_3(b_j - q_j)\nabla q_j \qquad (4.6)$$

where ∇p_i and ∇q_j are gradient terms of equality and inequality constraints respectively and these terms are found analitically and replaced in to the problem.

$$\alpha_i^+ = \alpha_i - 2w_1(a_i - p_i) \qquad (4.7)$$

if $j \in C_a$, then;

$$\beta_j^+ = \begin{cases} 0, \text{ if } \beta_j - 2w_2[b_j - q_j] \leq 0 \\ \beta_j - 2w_2[b_j - q_j], \text{ otherwise} \end{cases} \qquad (4.8)$$

if $j \in C_b$, then;

$$\beta_j^+ = \begin{cases} 0, \text{ if } [b_j - q_j] \geq 0 \\ 2w_3[b_j - q_j], \text{ otherwise} \end{cases} \qquad (4.9)$$

w_i values during this update process are those values used in the previous step. After new β values are generated, the sets C_a and C_b are also updated.

Given this standard formulation of Augmented Lagrangian Method the associated torque ripple minimization problem variables and parameters are presented in the next section.

4.4 Optimization Parameters

Correct design of airgap geometry is very effective on torque production of the motor. For this reason, this work is based on optimizing the airgap parameters of the motor. Because of the reason explained in section 3.2, the effect of adjacent teeth is assumed to be negligible.

Independent optimization variables in this problem are listed below:

λ/g - tooth pitch over airgap length,
t_s/λ - stator tooth width over tooth pitch,
t_r/λ - rotor tooth width over tooth pitch,

$$x = (\lambda/g \;\; t_s/\lambda \;\; t_s/\lambda). \qquad (4.10)$$

These are the variables to be optimized. Moreover, parameters used for the construction of objective function are:

x_n - normalized rotor position,

mmf - Applied mmf to the teeth region.

Inorder to reduce the computational difficulties which may be associated with the method used, normalized parameters rather than the actual parameters should be used throughout the optimization process. This is because if the orders of the optimization parameters are very different from each other the convergence may be slowed down. Therefore, in our case, all optimization parameters are normalized within the range [0-1]. This is obtained by dividing λ/g values 250 (maximum value defined), and mmf values to 1500 (since for λ=0.0172, maximum value of mmf is 1500, for udss structure maximum mmf is 1500/0.0172). The force on the other hand is observed to span a range between 0 to 1500 N for the data. For this reason force data is normalized with respect to 1500 N. Since other parameters are in the range [0-1] they are not needed to be normalized.

4.5 Constraints

Constraints of the problem are :

$$40 \leq \lambda/g \leq 250 \qquad (4.11.a)$$

in normalized form (normalized with respect to λ=250) this constrained may be written as;

$$0.16 \leq \lambda/g \leq 1 \qquad (4.11.b)$$

$$0.3 \leq t_s/\lambda \leq 0.5 \qquad (4.12)$$

$$0.3 \leq t_r/\lambda \leq 0.5 \qquad (4.13)$$

The reason for these constraints is the fact that the neural network that functions as the objective function is trained for λ/g, t_s/λ, t_r/λ values within these intervals. The networks were trained within these regions, because above or below these ranges designing motor is physically unpracticle, e.g. very norrow or large airgap, very thin or thick tooth widths.

- For minimizing out position inductance;

$$t_s/\lambda + t_r/\lambda \leq 1$$

The first three constraints implies that, the last one is automatically satisfied ; hence it is redundant and it is not included to the problem.

- For increasing the winding space,

$$t_s/\lambda - t_r/\lambda \leq 0 \qquad (4.13)$$

For all practical designs, rotor pole width will be greater than or equal to stator pole width since srm has only stator windings.

- Every parameter have to be greater than zero (nonnegativity constraints)

- For the motor be capable of self starting[33],

$$t_s/\lambda \geq 1/q \qquad (4.14)$$

where q is the number of phases.

For this problem q is assumed to be 4 and this condition is also automatically satisfied because of Eq.(4.12) and is redundant.

Then the constraints of the optimization problem that we deal with may be written in normalized form as;

i) $0.16 \leq \lambda/g \leq 1$

ii) $0.3 \leq t_s/\lambda \leq 0.5$

iii) $0.3 \leq t_r/\lambda \leq 0.5$

iv) $t_s/\lambda - t_r/\lambda \leq 0$

v) $t_s/\lambda \geq 0, t_r/\lambda \geq 0, \lambda/g \geq 0$

4.6 Objective Function: Formulation of Normalized Torque Ripple

The most crucial point in the torque ripple optimization process, as mentioned several times in the previous chapters, is the fact that the objective function which is the torque ripple function can not be expressed as a mathematical function of the objective variables. This is the reason why a neural network that maps the objective variables into the torque ripple was trained with a set of computed data points. So the neural network is the main tool in the objective function formulation process.

The neural network trained is a function (subroutine) within the optimization program written on C, which returns force (S) when called with parameters λ/g, t_s/λ, t_r/λ, mmf, x_n.

In the discussion above the way λ/g, mmf (F), force (S) are normalized within the neural network was described. In searching for the torque ripple the optimization procedure again requires these variables in normalized form. Therefore, no further conversion of variables is needed. In summary, the source data is for a structure with λ=0.0172. The neural net extracts the required information from this data set and it is then converted to either to udss data or data for a specific motor.

Here, it is important to reconsider the fact that this study does not involve one single optimization problem but five similar minimization problems for five different mmf levels (30, 40, 50, 60, 70 kAT).

Given the fact that the neural network trained returns force (S), torque is theoretically proportional to force and torque ripple is a phenomenon related with the fluctuation of force among the different positions of the motor, the following heuristic is proposed to represent the torque ripple.

For each vector $[\ \lambda/g\ \ t_s/\lambda\ \ t_r/\lambda\ \ mmf\ ]^T$ of interest the neural network is called to return force for each normalized rotor positions (11 times):

$$x_n = 0,\ 0.1,\ 0.2,\ .....,1$$

As seen from Fig.4.1 in a range of 30° ($0 \le x_n \le 1$ in normalized form) the points on the force locus corresponding to each x_n is calculated and this constitutes an array:

$$S_y(i) = \begin{cases} \max\left[S\left(\lambda/g, t_s/\lambda, t_r/\lambda, mmf, x_i\right), S\left(\lambda/g, t_s/\lambda, t_r/\lambda, mmf, x_i + 0.5\right)\right] \\ \quad \text{if } x_i \le 0.5 \\ \max\left[S\left(\lambda/g, t_s/\lambda, t_r/\lambda, mmf, x_i\right), S\left(\lambda/g, t_s/\lambda, t_r/\lambda, mmf, x_i - 0.5\right)\right] \\ \quad \text{if } x_i > 0.5 \end{cases}$$

$$\text{for } i = 1, ...,11 \qquad (4.15)$$

where $S(\lambda/g,\ t_s/\lambda,\ t_r/\lambda,\ mmf,\ x_i)$ is the force value returned by the neural network for

$$\bar{y} = (\ \lambda/g\ \ t_s/\lambda\ \ t_r/\lambda\ )^T \qquad (4.16)$$

Next, the peak value of the force is found

$$S_y^{max} = \max\{S_y(1), S_y(2),..., S_y(11)\} \tag{4.17}$$

For a fixed mmf value objective function to be minimized is defined as

$$f(\bar{y}) = \frac{1}{2} \cdot \sum_{i=1}^{11} \left(\frac{S_y^{max} - S_y(i)}{S_y^{max}} \right)^2 \tag{4.18}$$

By dividing the difference term to S_{max} , normalized form of ripple is obtained and comparison between different mmf levels becomes meaningful.

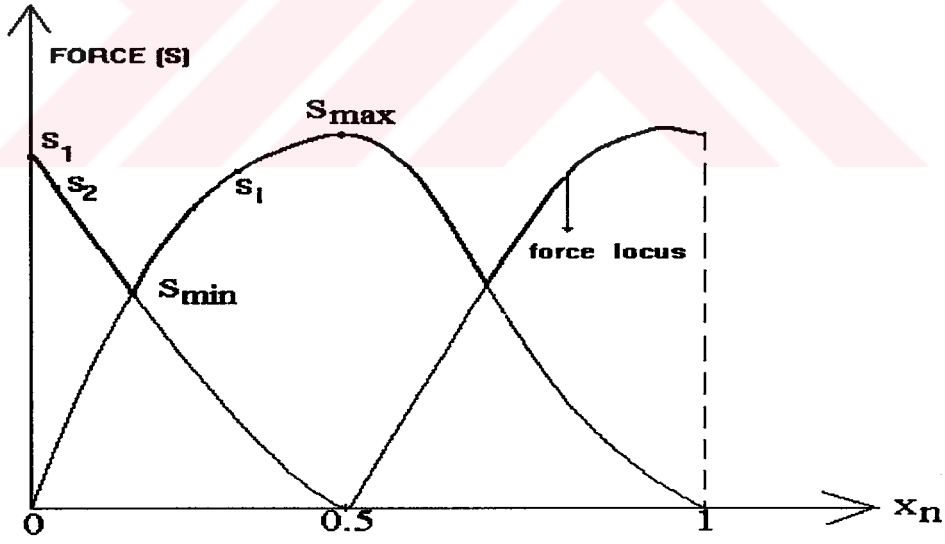


Figure 4.1 Torque ripple curve

4.7 Minimization of the Augmented Lagrangian Function: Davidon Fletcher Powell (DFP) Method

In the previous section, the constrained minimization problem was converted to a unconstrained minimization problem by the Augmented Lagrangian method. The Augmented Lagrangian function implicitly involves the constraints and in this form it can be minimized by several unconstrained minimization methods.

In this study Davidon Fletcher Powell method which is known as one of the Quasi-Newton methods is used. In order to understand this method, first the Newton's method should be understood.

Beginning from the simplest case, suppose that the function f of a single variable x is to be minimized, and suppose that at a point x_k where a measurement is made it is possible to evaluate the three numbers $f(x_k)$, $f'(x_k)$, $f''(x_k)$. Then by Taylor expansion around x_k it is possible to construct a quadratic function q which at x_k agrees with f up to second derivatives, that is

$$q(x) = f(x_k) + f'(x_k)(x - x_k) + 0.5\, f''(x_k)(x - x_k)^2 \qquad (4.19)$$

We may then calculate an estimate of x_{k+1} of the minimum point of f by finding the point where the derivative of q vanishes. Thus setting

$$q'(x_{k+1}) = f'(x_k) + f''(x_k)(x_{k+1} - x_k) = 0 \qquad (4.20)$$

we find

$$x_{k+1} = x_k - (f'(x_k)/ f''(x_k)) \qquad (4.21)$$

This process which is illustrated in Fig.4.2, can be repeated at x_{k+1}

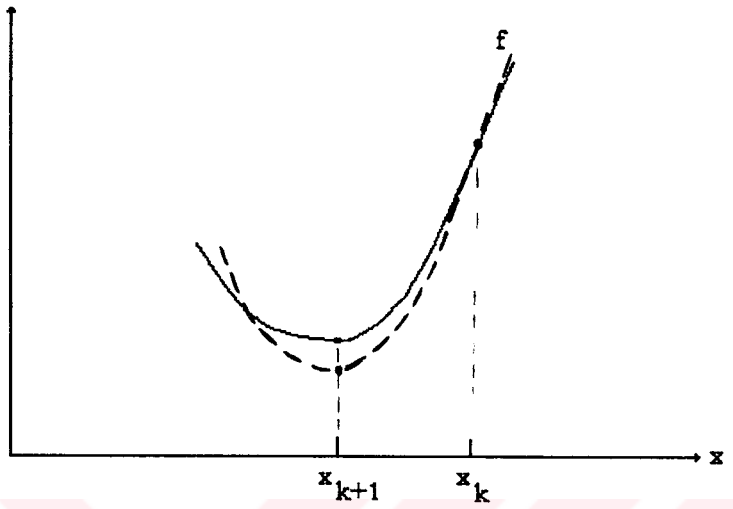


Figure 4.2 Newton's method for minimization

The idea behind Newton's method is that, the function f being minimized is approximated locally by a quadratic function, and this approximate function is minimized exactly. For a function f of several variables (f: $R^n \rightarrow R$), by the same logic we can approximate f by the truncated Taylor series near $\bar{x}_k$

$$f(\bar{x}) \cong f(\bar{x}_k) + \nabla f(\bar{x}_k)(\bar{x} - \bar{x}_k) + \frac{1}{2}(\bar{x} - \bar{x}_k)^T H(\bar{x}_k)(\bar{x} - \bar{x}_k) \qquad (4.22)$$

The function is minimized when;

$$\bar{x}_{k+1} = \bar{x}_k - \left[H(\bar{x}_k)\right]^{-1} \nabla f(\bar{x}_k)^T \qquad (4.23)$$

Eq.4.23 is the pure form of Newton's method. In view of the second order sufficiency conditions for a minimum point, we assume that at a relative minimum point $\bar{x}^*$, The Hessian matrix $H(\bar{x}^*)$ is positive definite. We can then argue that if f has continuous second partial derivatives, $H(\bar{x})$ is positive definite near $\bar{x}^*$, and hence the method is well defined near the solution.

At points remote from the solution, the algorithm must be modified in order to guarantee convergence. The modification is that a search parameter α is introduced so that the method takes the form

$$\bar{x}_{k+1} = \bar{x}_k - \alpha_k \left[H(\bar{x}_k)\right]^{-1} \nabla f(\bar{x}_k)^T \qquad (4.24)$$

where α_k is selected to minimize f. Near the solution we expect, on the basis of how Newton's method was derived, that $\alpha_k \cong 1$. Introducing the parameter for general points, however, guards against the possibility that the objective might increase with α_k=1, due to nonquadratic terms in the objective function.

In most cases, such as this study evaluation and inversion of the Hessian matrix is impractical or costly. The idea underlying quasi-Newton methods is try to construct the inverse Hessian or an approximation of it, using information gathered as the descent process progresses. The current approximation is then used at each stage to define the next descent direction as in the Newton's method. Ideally, the approximations converges to the inverse of the Hessian at the solution point and the overall method behaves somewhat like Newton's method.

One of the most sophisticated methods for constructing the inverse Hessian is the Davidon Flecher Powell Method. This method is one of the most effective search method because of the following advantages:

1. It requires only the value and the gradient of the function to be minimized.
2. Search directions are always guaranteed to be in the directions of descent.
3. Starting from any positive definite matrix, the approximated positive definite matrices converge to the actual inverse Hessian at the solution point.

The theory underlying the method can be found in any advanced optimization text book[25]. Here only the algorithm will be given.

STEP 0:

Start with any symmetric positive matrix S_0 (preferably the identity matrix) and any point $\bar{x}_0$. S_k denotes the approximation for the inverse Hessian at the kth iteration. Then starting with k=0,

STEP 1:

Set $d_k = - S_k.g_k$

where $g_k = \nabla f(\bar{x}_k)^T$, d_k is the direction of descent.

STEP 2:

Minimize $f(\bar{x}_k + \alpha d_k)$ with respect to $\alpha \geq 0$ to obtain $\bar{x}_{k+1}$, $p_k = \alpha_k d_k$ and g_{k+1}

(Search parameter α is evaluated (minimized) by the method explained in section.3.6.4)

STEP 3:

Set $q_k = g_{k+1} - g_k$ and

$$S_{k+1} = S_k + \frac{p_k p_k^T}{p_k^T q_k} - \frac{S_k q_k q_k^T S_k}{q_k^T S_k q_k} \qquad (4.25)$$

Update k and return to STEP 1.

This algorithm keeps up iterating until $\left\|\bar{x}_{k+1} - \bar{x}_k\right\| \le \varepsilon$, where ε is a number within the range $[10^{-2}, 10^{-6}]$chosen by the user.

As it may seen from the method the gradient vector (of the objective function f) needs to be evaluated at each iteration. But in this study, since the objective function (Augmented Lagrangian L_a) is not a pure mathematical function but an indirect function of several neural network outputs, in order to evaluate the gradient a numerical differentiation method is required. This method is the subject of the next section.

4.8 Numerical Differentiation of the Objective Function

The gradient of the Augmented Lagrangian function is given by Eq.(4.2). It may be seen that this gradient involves the gradient term of the original objective function (torque ripple function). The torque ripple, for a given input vector, can only be evaluated algorithmically from several neural network outputs. So the differentiation should also be carried out computationally by typical numerical differentiation methods.

In its simple form since

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{4.26}$$

the straightforward method is to choose a sequence $\{h_k\}$, so that $h_k \to 0$ and compute the limit of the sequence

$$D_k = \frac{f(x+h_k) - f(x)}{h_k} \qquad \text{for } k=1,2,.....,n,.... \tag{4.27}$$

Beginning from D_1, D_k ($k = 2,3,....$) is computed iteratively, also computing the difference D_i-D_{i-1} at each successive iteration. The process is terminated when this difference falls below a very small number ε (defined by the user) for a number of consecutive iterations. At this termination point (say, iteration j), D_j is chosen as the final approximated value of $f'(x)$.

If the function f(x) can be evaluated at values that lie to the left and right of x than a two point formula will involve abscissas that are chosen symmetrically on both sides of x. The simplest central-difference formula is;

$$f'(x) \cong \frac{f(x+h) - f(x-h)}{2h} \tag{4.28}$$

In this study, a more sophisticated version of this equation is used;

$$f'(x) \cong \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} \tag{4.29}$$

The derivation of Eq.4.29 is available in Reference 27.

The above formula is given for a function of one single variable but since the torque ripple function is a function of several variables the algorithm defined by Eq.(4.29) is applied to each individual component of the input vector simultaneously to obtain the gradient.

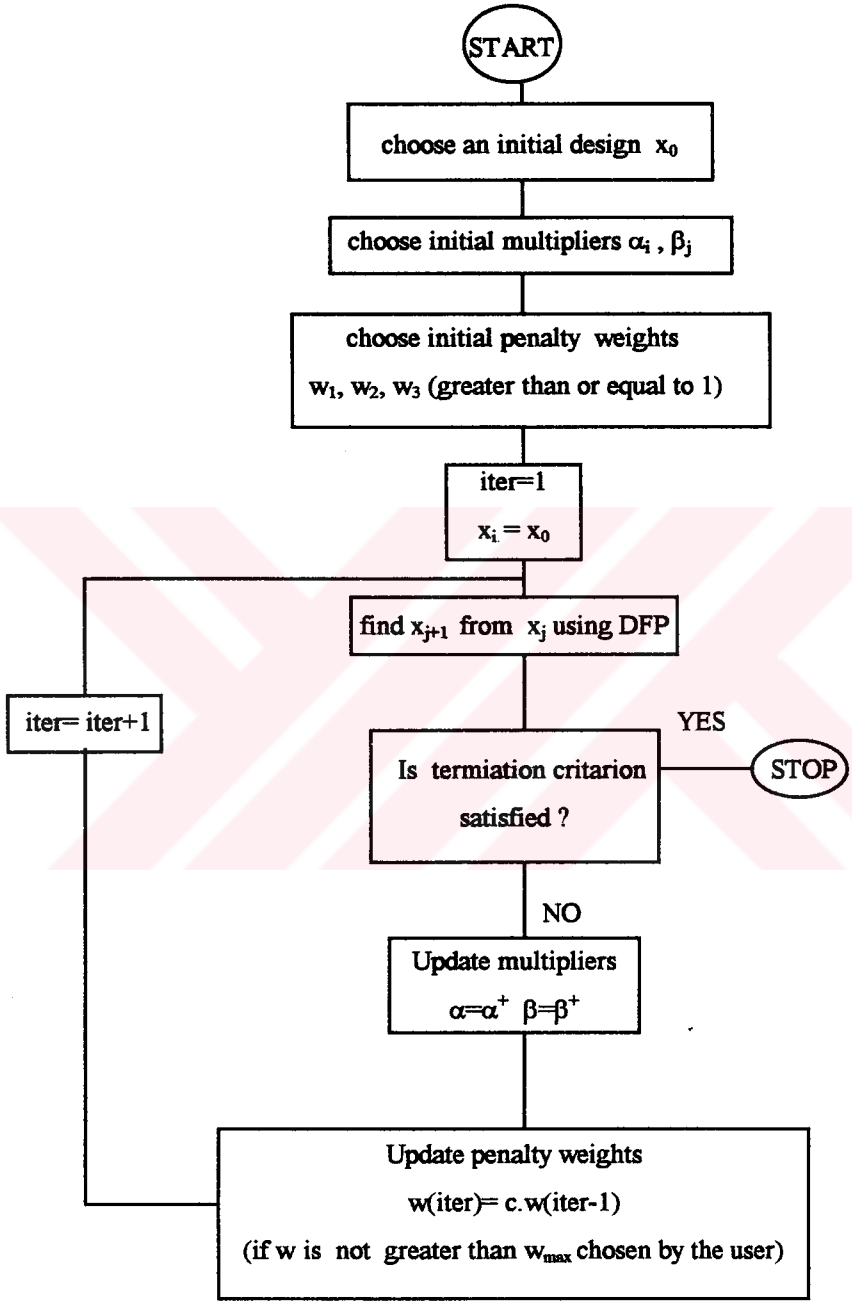


Figure 4.3 Flowchart of the optimization procedure

4.9 Flowchart of the Algorithm

The unconstrained optimization procedure described in the previous section may be summarized by the flowchart shown in Fig.4.3. This program is also written on C programming language and contained in Appendix B.

4.10 Application of the Optimization Algorithm

The solution of the unconstrained optimization problem can be made urbitrarily close to the optimal solution of the original problem by choosing the penalty factor, w, sufficiently large. However the behaviour of the method depends on the choice of the penalty factor and the starting point. Especially for starting points far from the final points, choice of large penalty factors will give way to computational difficulties[33]. Therefore, it is suitable to start by a moderate penalty factor and to find a solution for the resulting problem. Then the penalty factor is increased by a certain amount and starting from the solution of the previous step, another solution is found. The penalty factor at each step is increased according to the relationship;

$$w(i) = c.w(i-1), \tag{4.30}$$

where $w(i) < w_{max}$, and i is the number of iteration. The optimal choice of penalty increment factor, c, depends on the problem and even on starting point in a quite complex manner. In general for low values of c($c<2$) convergence needs a large number of iterations, whereas for high values of c($c>7$) convergence problems arise. For our problem, initial values of constants w, and c are chosen as 2, and 2 respectively, and maximum value of w (w_{max}) is limitted to32.

Lagrangian Multipliers for equality and inequality constraints, α_i (i=1,..,m, where m is the number of equality constraints) and β_j (j=1,..,n, where n is the number of inequality constraints) are both chosen initially as 1 and are updated in each iteration according to the equations 4.8 and 4.9.

For all optimization procedures, optimization criteria is used for terminating the process. Throughout the optimization procedure convergence factor, ε, is chosen as 10^{-6} for accuracy.

4.11 Optimization Results

The torque ripple minimization procedure is carried on from 54 different initial points which are the combinations of the following values of the variables

λ/g : 40, 70, 100, 150, 200, 250

t_s/λ : 0.3, 0.4, 0.5

t_r/λ : 0.3, 0.4, 0.5

In order to determine the effect of the variables on torque ripple a series of solutions is obtained with objective function in Eq.(4.18), while keeping the MMF as a parameter. The results of this investigation are summarized in Table 4.1. This table indicates that torque ripple is minimized for large λ/g (>200) values and λ/g increases with increasing saturation. At the minimum ripple point t_s/λ is around 0.4 and t_r/λ is around 0.5.

A mathematical search for minimum unfortunately conceals many facts from the designer. It is of interest to discover how sensitive the torque ripple is to variation of each variable. For this reason, a further investigation is carried out where for example MMF, t_s/λ, t_r/λ combinations are fixed and λ/g is allowed to vary. In this

case, a different concept, 'percentage torque ripple' is introduced. Percentage torque ripple (see Fig.4.1) may be calculated as:

$$\% \text{ torque ripple} = \frac{S_{max} - S_{min}}{S_{max}} \cdot 100 \qquad (4.31)$$

Table 4.1 Optimum variables for different MMF levels for udss

MMF (kA)	λ/g	t_s/λ	t_r/λ
30	200.0	0.42	0.50
40	199.1	0.44	0.48
50	201.2	0.42	0.50
60	250.0	0.4	0.50
70	249.2	0.37	0.50

A typical result is given in Fig.4.4, for different combinations of t_s/λ, and t_r/λ, at 50 kA excitation level. For example, Fig.44.a, where t_s/λ, and t_r/λ are equal to 0.4 and 0.5 respectively, clearly displays how torque ripple falls from %40 to nearly %5 with increasing λ/g.

In general, as the mmf applied to udss is changed between 40 kAT to 70 kAT, the best results are found for t_s/λ, and t_r/λ are equal to 0.4 and 0.5 respectively. The ripple at this condition is about %6. The point at which the minimum ripple occurs shift towards higher λ/g values as the mmf is increased as indicated in Table 4.2. The worst ripple occurs for norrow teeth (t_s/λ, and t_r/λ are equal to 0.3) and is in

the order of %35. For other combinations, the ripple is about %20. When both sides have a (t_s/λ, and t_r/λ) ratio of 0.5, the ripple is again low. However, large t_s/λ values leave little space for windings and is not practically acceptable.

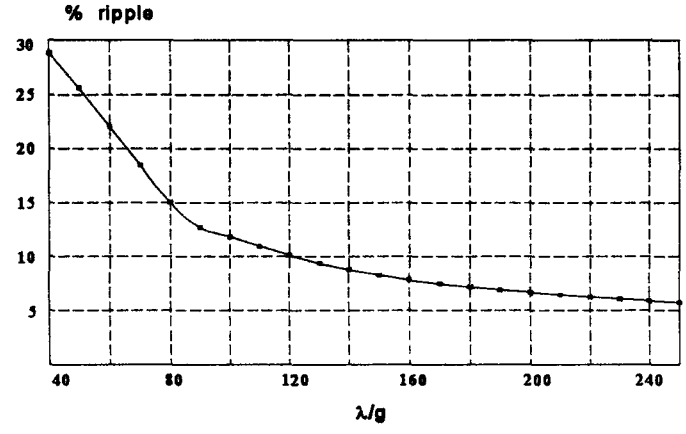
Fig.4.5 displays another set of results where λ/g is kept as a parameter and t_r/λ is allowed to be the only variable of optimization, and the excitation is 50 kA. This figures shows how sensitive the torque ripple is to t_r/λ. For example, in Fig.4.5.b variation of t_r/λ between 0.5 to 0.48, while t_s/λ=0.4, doubles the ripple and increases from 5% to 10%. For different excitation levels, graphs which shows the effect of λ/g and t_r/λ on torque ripple are contained in Appendix B. Investigation of these figures show that the variation of torque ripple has its similar trend for different excitation levels.

Generally in design optimization problems, optimum intervals are preferred to optimum points. Hence a table presenting minimum ripple regions for the design parameters is prepared for different values of mmf, t_s/λ, t_r/λ. Table 4.2 summarizes the findings and gives λ/g values and regions corresponding to minimum torque ripple. From the table it may be concluded that, minimum ripple values obtained where t_s/λ and t_r/λ values are around 0.4 and 0.5 respectively.

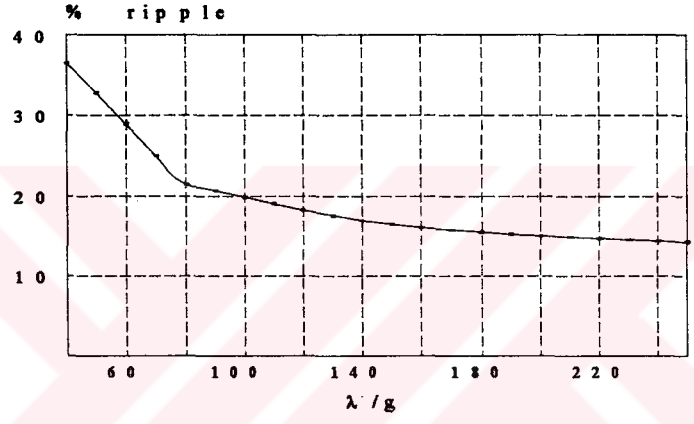
It may be also concluded that for low excitation levels ripple is slightly higher than what is found for higher excitation levels (greater than 50 kAT for udss) and optimum λ/g values lie in the range λ/g greater than 100, and values λ_r/g less than 200 for 30 kAT and nearer 250 for excitation values approaching 40 kAT. At higher excitation levels (greater than 50 kAT), ripple tends to decrease with increasing excitation but is not very noticable for t/λ combinations other than 0.3,0.3. In this excitation range the optimum torque ripple value is available for λ/g values between 150-250.

Table 4.2 Sensitivity to λ/g: Minimum ripple regions

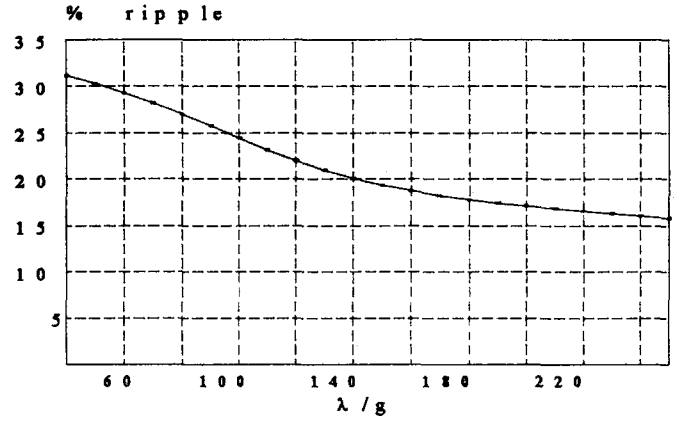
MMF (KAT)	t_w/λ	t_r/λ	MINIMUM RIPPLE VALUE (%)	λ/g value corresponding to minimum ripple point	λ/g regions corresponding to minimum ripple (up to + %10 of minimum ripple)
30	0.3	0.3	36.49	140	80-250
30	0.4	0.4	19.73	140	100-200
30	0.5	0.5	9.17	140	110-180
30	0.3	0.4	16.82	120	90-140
30	0.3	0.5	18.73	140	100-200
30	0.4	0.5	5.88	160	130-200
40	0.3	0.3	39.5	250	90-250
40	0.4	0.4	23.85	250	100-250
40	0.5	0.5	15.21	160	100-250
40	0.3	0.4	16.81	120	90-140
40	0.3	0.5	20.96	250	120-250
40	0.4	0.5	8.88	250	130-250
50	0.3	0.3	39.44	250	120-250
50	0.4	0.4	21.06	250	150-250
50	0.5	0.5	10.78	250	190-250
50	0.3	0.4	14.23	250	170-250
50	0.3	0.5	15.84	250	190-250
50	0.4	0.5	5.76	250	200-250
60	0.3	0.3	35.37	250	180-250
60	0.4	0.4	21.69	250	160-250
60	0.5	0.5	9.62	250	200-250
60	0.3	0.4	16.11	250	170-250
60	0.3	0.5	14.51	250	190-250
60	0.4	0.5	6.81	250	200-250
70	0.3	0.3	22.05	250	120-250
70	0.4	0.4	15.65	250	220-250
70	0.5	0.5	9.55	250	190-250
70	0.3	0.4	17.58	250	150-250
70	0.3	0.5	22.21	250	200-250
70	0.4	0.5	6.82	250	200-250



(a) $t_s/\lambda=0.4$, $t_r/\lambda=0.5$

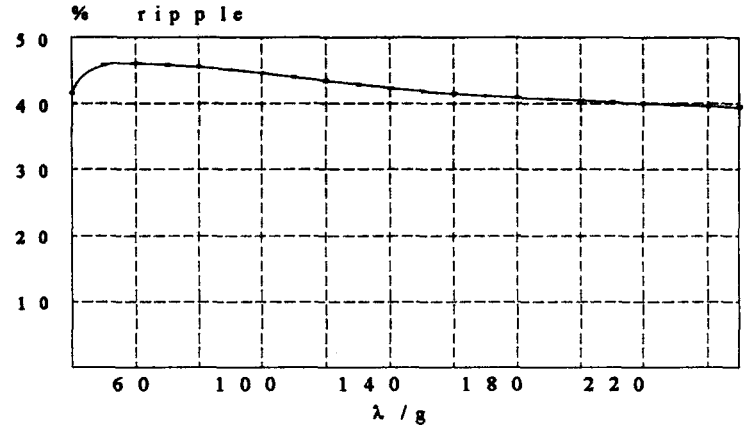


(b) $t_s/\lambda=0.3$, $t_r/\lambda=0.4$

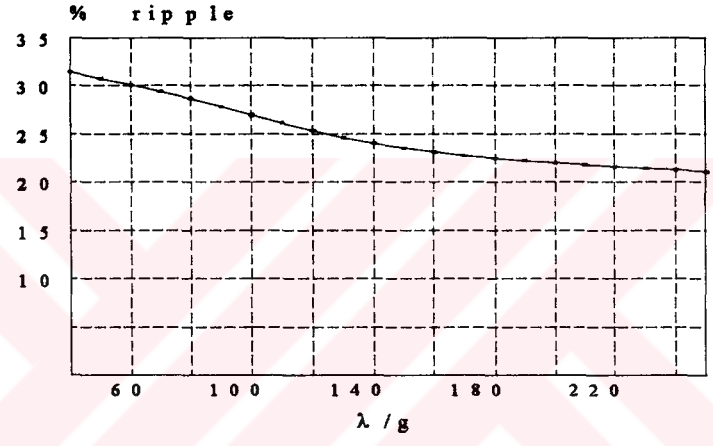


(c) $t_s/\lambda=0.3$, $t_r/\lambda=0.5$

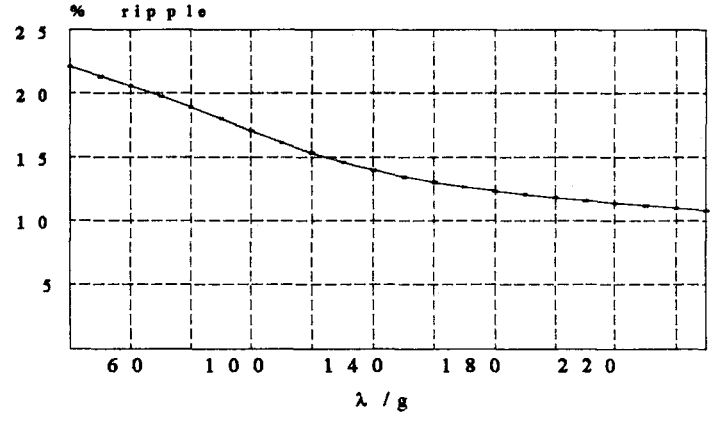
Figure 4.4 Effect of λ/g on torque ripple for different t_s/λ, t_r/λ values at 50 kA



(d) $t_s/\lambda=0.3$, $t_r/\lambda=0.3$

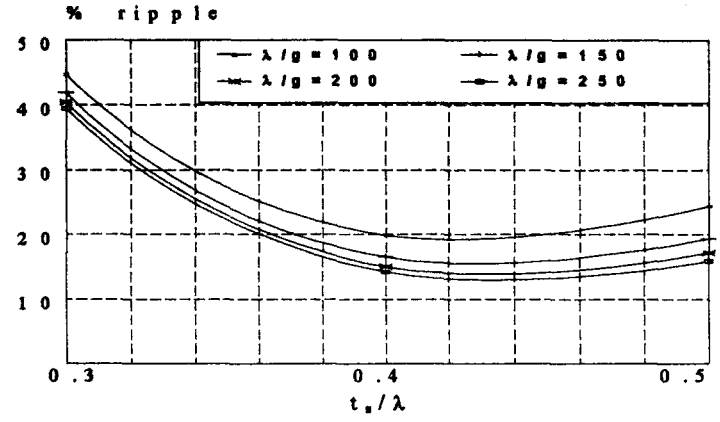


(e) $t_s/\lambda=0.4$, $t_r/\lambda=0.4$

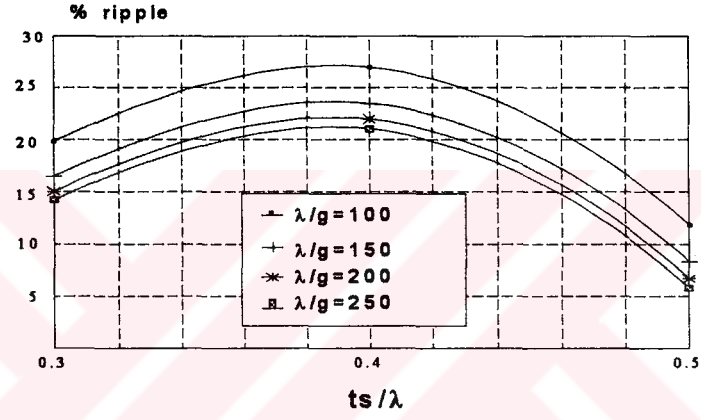


(e) $t_s/\lambda=0.5$, $t_r/\lambda=0.5$

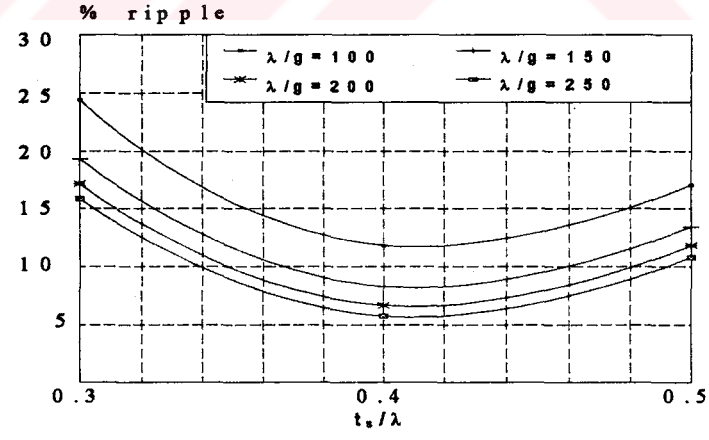
Figure 4.4 Cont'd



(a) t_r/λ=0.3



(b) t_r/λ=0.4



(c) t_r/λ=0.5

Figure 4.5 Effect of t_s/λ on torque ripple for different λ/g values at 50 kA mmf

From this research, the following generalizations can be made:

1. Torque ripple is lower when λ/g is greater than 150.
2. Torque ripple is increased when rotor and stator teeth widths are reduced, for example when t_r/λ or t_s/λ is around 0.3 the amount of torque ripple is greater.
3. Minimum ripple values are obtained where t_s/λ and t_r/λ values are equal to 0.4 and 0.5 respectively.
4. With increasing mmf, t_s/λ and t_r/λ values where minimum ripple points are found to be similar.
5. With increasing mmf, sensitivity to λ/g is increased. For example from Table 4.2 it is seen that, at 40 kA excitation, λ/g range corresponding to minimum ripple is approximately [100-250], when it is [200-250] at 70 kA excitation mmf.

4.12 Discussions

This chapter presents the results of an investigation that pursues the effect of magnetic circuit parameters on torque ripple of a doubly-salient switched reluctance motor which operates with a constant level of excitation at low operating speeds. The effect of ripple is most pronounced at these speeds since shaft speed changes are also more noticable. Although, due to the simplifying assumptions the torque ripple values may show some variation in practice, the general trend is clearly observable from the investigation. The findings indicate that by correctly choosing λ/g values greater than 150 and setting t_s/λ to values around 0.4, and t_r/λ to values around 0.5, the torque ripple can be minimized at low speeds.

However, although these evaluations are valid within the assumptions made, an important practicle aspect of designing a srm is left out. The winding space

requirement may affect the choice of t_s/λ and t_r/λ and a more complete investigation this factor must also be taken into account. However, the findings here are summarized in Table 4.2 to indicate ranges of parameters where low torque ripple may be obtained and the general trend for reducing torque ripple. A designer can use the information provided to achieve low torque ripple designs.

CHAPTER 5

COMPARISON OF THE OPTIMIZATION RESULTS WITH FINITE ELEMENT FIELD CALCULATIONS

5.1 Introduction

In this chapter, static torque curves of optimum geometries (the geometries defined by the results of the optimization study) are predicted using finite element analysis to test the validity of the findings from the data and the optimization.

A discussion on the adoptation of the data is also contained in this chapter. Effect of overlapping on force for doubly salient structures is discussed and methods are proposed for eliminating this effect. Another important issue contained in this chapter is the normalization of the parameters of the practical motors with unequal number of stator and rotor teeth. Normalization with respect to the stator and rotor tooth pitch is discussed and results from different normalizations for test motor SR2 are compared

5.2 A Discussion on the Adaptation of the Data to the Present Problem

One important issue in this study is the effect of adjacent tooth pairs on the force produced by asymmetrically slotted doubly salient structures. The usage of a neural network, with the available data, made it possible to make force calculations for various geometries within a wide range as discussed in Chapter 3. Hovewer, it is

important to consider the fact that the available data had been produced by Ertan[16,17] for structures with equal stator and rotor tooth pithces and tooth widths.

In Section 3.9, the method developed by Ertan[19] for force calculation of asymmetrically slotted structures from symmetrically slotted tooth pairs is explained. In order to understand the method first the method used to obtain the original data by Ertan should be explained. The main point to understand is the fact that during the force data calculation by field solutions by Ertan[19], not only the stator tooth for which the force was intended to be calculated, but all adjacent stator teeth were excited as well. Hovewer, in this study the aim is to obtain force-position characteristics resulting from one-phase-on excitation. For force calculation of asymmetrically slotted structures, the aim is to eliminate the effect of the adjacent excited stator tooth. This is accomplished by modelling the structure such that the distance between two adjacent tooth pairs is larger than its actual value.

In Fig.5.1, adjacent tooth pairs are presented. In this figure, "o" represents amount of overlap and "x" represents the distance between the rotor tooth and the adjacent stator tooth. In his study, by using various field solutions, Ertan proved that for values of "x" larger than 25g, the effect of adjacent tooth pair is negligible.

Considering this fact, in force calculations of asymmetrically slotted structures, if rotor tooth pitch, λ_r (t_s+d+x), is less than t_s+d+25g, the adjacent teeth has the effect to decrease force. As discussed in Sec.3.4, in a switched reluctance motor when a pair of symmetrical pairs are excited adjacent play no role in force production. Hence, in this study tooth pitch is assigned the value t_s+d+25g and reminding the fact that all parameters are normalized with respect to this new value, and force calculations are carried out according to this new normalization. By this method the effect of adjacent tooth is eliminated. This assumption means that the force produced by a pair is mainly dictated by the amount of overlap.

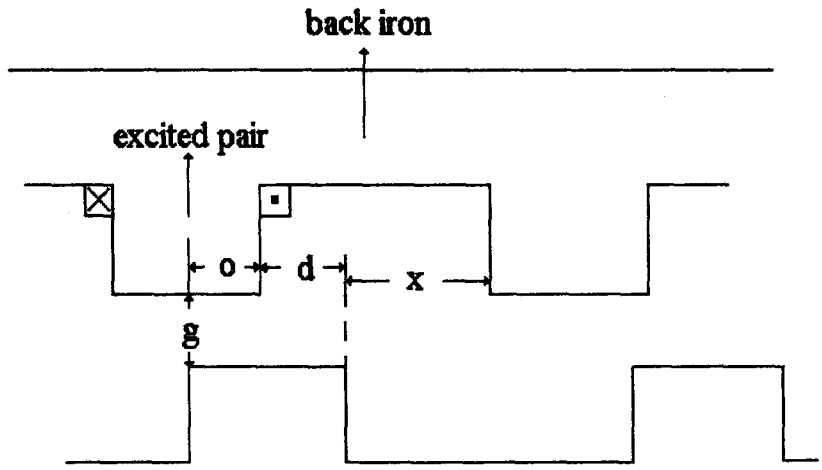


Figure 5.1 A model of doubly salient structure

In order to verify this approach, considering a geometry (as seen in Fig.5.2) where overlapping effect is considerably high, and assuming the stator tooth width of the test motor SR2 is widened such that t_s/λ_r is equal to 0.5, force characteristics are obtained using ANSYS. In this verification study the parameters t_s/λ_r, t_r/λ_r, λ_r/g are 0.5, 0.4 and 79.25 respectively, and it is obvious for this geometry that overlapping effect is high. In Table 5.1, ANSYS and neural network results are compared. In this table both the amount of overlapping (o/g), and the distance between the rotor tooth and the adjacent stator tooth is given for various positions in normalized forms.

In Fig.5.3, the results in Table 4.1 are plotted. The good agreement between the neural network results and ANSYS solutions is readily observable. The discreapency at x_n is equal to 0.7 and 0.8 is probably due to meshing problems and needs to be verified ones again. However, these results support the assumption that the overlap with the adjacent pole does not introduced a serious error to predictions.

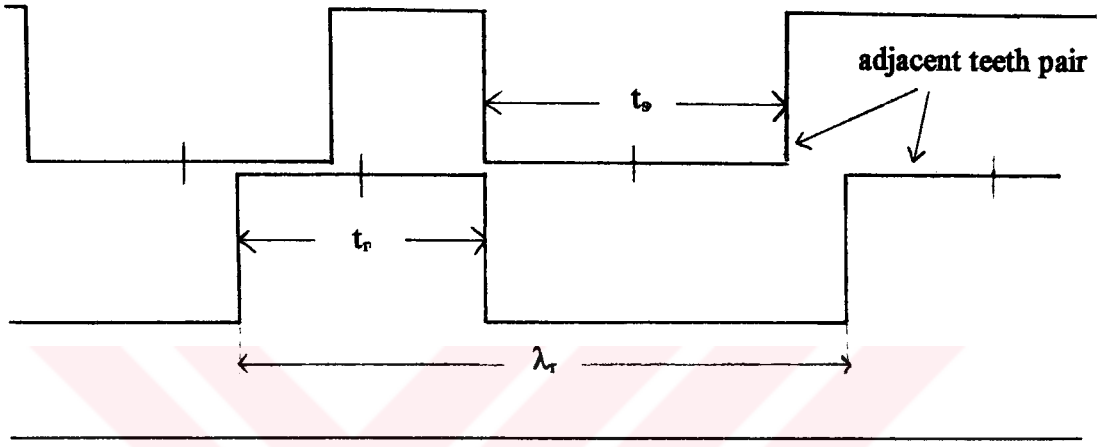


Figure 5.2 The relative position of teeth for t_s/λ_r=0.5, t_r/λ_r=0.4 for position x_n= 0.6

Table 5.1 Comparison of force data calculated using ANSYS and neural network to test the effect of adjacent teeth pair (t_s/λ_r=0.5, t_r/λ_r=0.4, λ_r/g=79.25).

rotor position (degree)	normalize rotor position w.r.t λ_r (x_n)	overlap o/g	x/g	Torque calculated using ANSYS	Predicted torque from neural net.
3	0.1	31.7	19.81	0.25	0.46
6	0.2	27.7	15.85	0.68	0.92
9	0.3	23.8	11.9	1.06	1.28
12	0.4	19.8	7.93	1.37	1.54
15	0.5	15.85	3.96	1.59	1.72
18	0.6	11.9	0	1.75	1.84
21	0.7	7.93	-3.96	1.87	1.80
24	0.8	3.96	-7.93	1.96	1.70

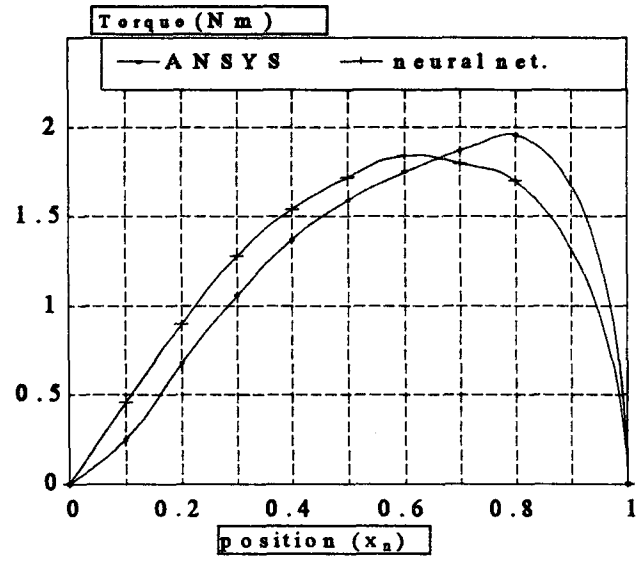


Figure 5.3 Comparison of force data calculated using ANSYS and neural network to test the effect of adjacent teeth pair (t_s/λ_r=0.5, t_r/λ_r=0.4, λ_r/g=79.25).

Another important problem is that the stator and rotor tooth pitches of the structure, which the original force data is calculated for, are equal. But, practically this is impossible, since the number of stator and rotor teeth have to be different for operation. For example, for the test motor SR2, number of stator teeth (N_s) is equal to 8 and number of rotor teeth (N_r) is equal to 6, and it is obvious that the stator and rotor tooth pitches are different ($\lambda_s=6\lambda_r/8$). In this case, the problem is which one of the tooth pitches (λ_s or λ_r) is used for parameter normalization. But for ideal case, it means there is no overlapping between phases (x>25g), there is no difference between normalization with respect to stator and rotor tooth pitches.

For verifying this fact, for the test motor SR2 all parameters are normalized with respect to stator tooth pitch and the results of two types of normalizations are compared with the measured force data. In Table 5.2 the results of different type of normalizations are shown. From the table, the results are similar when the effect of

overlapping is small. But for normalized positions which greater than 0.6, the error of second type of normalization is increased. In this table, normalized positions of the rotor with respect to λ_s and λ_r are also seen for the same amount of rotation, and here x_{ns} is normalized rotor position with respect to λ_s, and x_{nr} is the normalized position with respect to λ_r. Since the ratio of the number of stator and rotor teeth is 8/6, the relationship between x_{ns} and x_{nr} may be derived easily as $x_{ns} = 8.x_{nr}/6$.

Table 5.2 Predicted force from neural net. data of SR2 for normalizations with respect to λ_s and λ_r

rotor position (degree)	normalized position with respect to λ_r (x_{ns})	normalized position with respect to λ_s (x_{nr})	x/g	measured torque of SR2 at 3A excitation	torque calculated by normalizing parameters w.r.t λ_r	torque calculated by normalizing parameters w.r.t λ_s
3	0.1	0.13	23.80	0.61	0.47	0.41
6	0.2	0.26	19.81	0.92	0.93	0.88
9	0.3	0.40	15.85	1.20	1.31	1.23
12	0.4	0.53	11.90	1.50	1.56	1.47
15	0.5	0.67	7.93	1.75	1.73	1.62
18	0.6	0.80	3.96	1.88	1.77	1.57
21	0.7	0.93	0	1.85	1.68	0.92
24	0.8	1.06	-3.96	1.71	1.38	0

5.3 Comparisons of Torque Curves Computed from Field Solutions and Neural Network Simulations

B-H data chosen in finite element based torque calculations of the motor designs found for the five different mmf levels, is the same with the B-H data in Ertan's computations. In order to reduce the slot effects, sloth depth was chosen (by Ertan) as 40g for the calculation of the data. Ertan, in his work, showed that this choice of slot depth effectively represents an infinite slot. For this reason, slot depth is

slot depth effectively represents an infinite slot. For this reason, slot depth is choosen as 40g also in finite element modelling. Since the back iron mmf drop is neglected in this study, in order to reduce the drop, the back iron is modeled such that it is very thick ($3t_s$). Conductors are modeled such that they are very small and are sufficiently far away from the airgap region. Models are carefully meshed as explained in Chapter 2.

For optimum geometries at different mmf levels, torque curves are obtained from field solutions (ANSYS) and also from neural network algorithm in order to make comparisons between them. The results are presented in Table 5.3 and Figures (5.4 to 5.8).

Percentage torque ripple associated with the curves are calculated from Eq.4.30 and tabulated in Table.5.4 for each approach (finite element and neural network). This table shows that the difference in torque ripple levels calculated from the two different approaches are not large. The error is in an acceptable range since the both approaches inherently have their error allowances. The resultant torque curves are similar in shape. The reason of the error may be related with the differrences in modelling with finite element analysis and neural network model, and solution errors due to meshing in numerical field solutions.

Table 5.3 Torque values computed using ANSYS and neural network simulations for optimum designs at different mmf levels.

MMF (kA)	Optimum Geometry	Normalized Position (x_n)	Torque (Nm) ANSYS	Torque(Nm) Neural Net.
30	λ/g=200 t_s/λ=0.418 t_r/λ=0.5	0.1	22.8	25.5
		0.2	52.7	54.8
		0.3	72.3	70.7
		0.4	82.1	78.4
		0.5	84	81.7
		0.6	84.2	82.1
		0.7	84.8	81.4
		0.8	84	78.9
		0.9	63	23.5
40	λ/g=199.1 t_s/λ=0.436 t_r/λ=0.48	0.1	29.9	43.9
		0.2	58.1	74.6
		0.3	78.1	91.5
		0.4	90.8	99.5
		0.5	97.4	102.6
		0.6	100	1031
		0.7	99.6	103.9
		0.8	93.4	105.2
		0.9	25.6	41.9
50	λ/g=201.2 t_s/λ=0.419 t_r/λ=0.495	0.1	30.4	51.9
		0.2	69.2	88.7
		0.3	102.2	112.1
		0.4	124.9	123.4
		0.5	133.7	128.1
		0.6	135.7	129.1
		0.7	136.3	129.3
		0.8	135.9	130.9
		0.9	110.9	54.7

Table 5.3 Cont'd

60	$\lambda/g=250$ $t_s/\lambda=0.4$ $t_r/\lambda=0.5$	0.1	36.1	68.1
		0.2	86.4	106.6
		0.3	120.4	134.1
		0.4	146.6	148.7
		0.5	158.5	154.9
		0.6	162.6	156.8
		0.7	165.4	156.9
		0.8	166.2	157.7
		0.9	150.7	68.3
70	$\lambda/g=249.2$ $t_s/\lambda=0.374$ $t_r/\lambda=0.5$	0.1	37.7	81.8
		0.2	93.7	118.2
		0.3	140.2	148.6
		0.4	167.8	165.7
		0.5	177.4	173.3
		0.6	183.8	176.1
		0.7	185.2	177
		0.8	186.5	173.7
		0.9	158.1	87.2

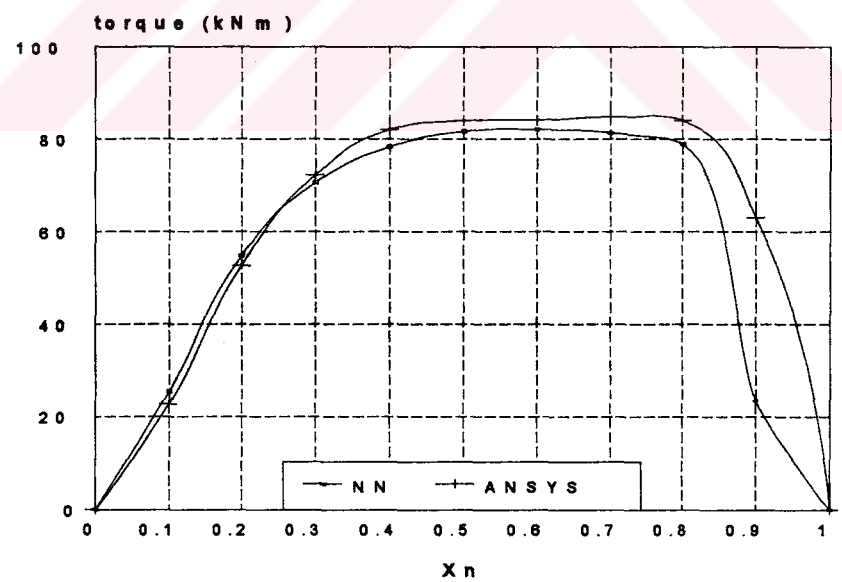


Figure 5.4 Torque curves computed using ANSYS and from neural network simulations(optimum design for MMF=30kA, $\lambda/g=200$, $t_s\lambda=0.418$, $t_r\lambda=0.5$)

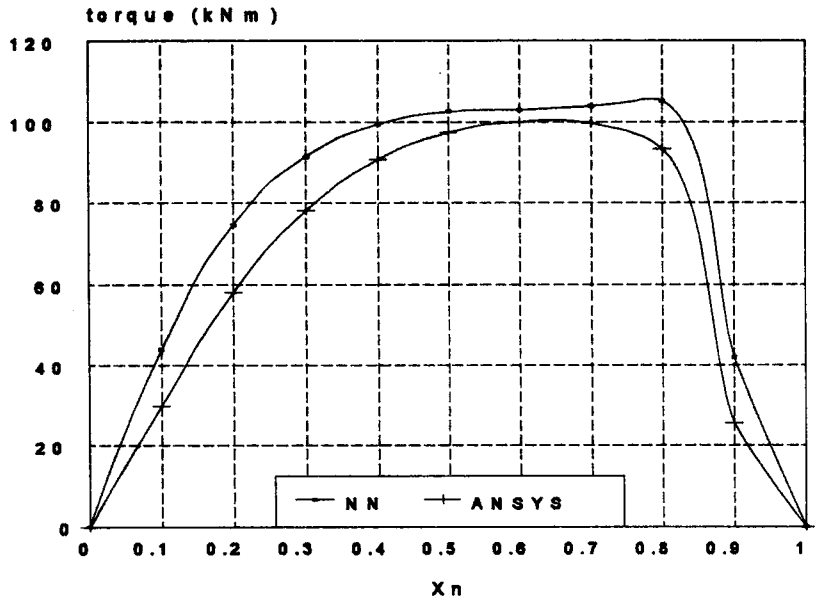


Figure 5.5 Torque curves computed using ANSYS and from neural network simulations (optimum design for MMF=40kA, λ/g=199.1, $t_s\lambda$=0.436, $t_r\lambda$=0.48)

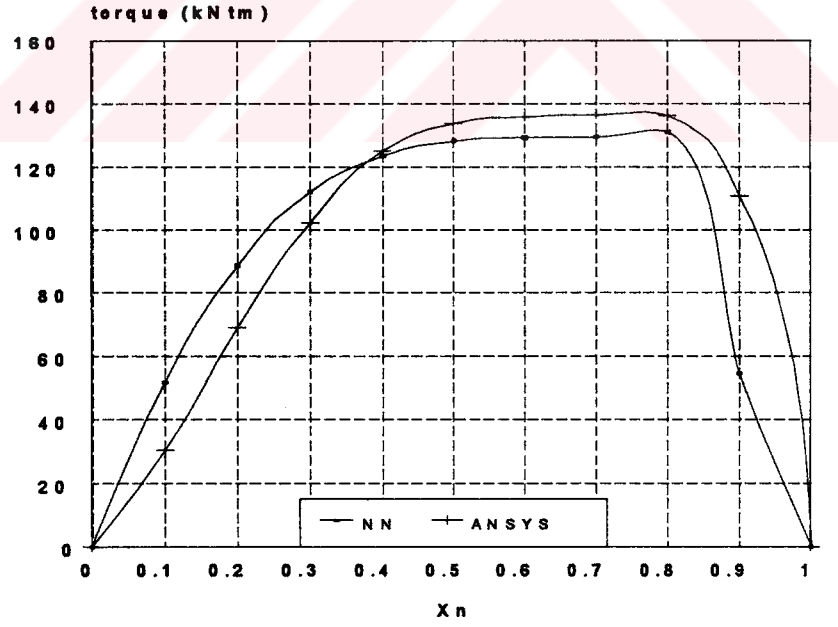


Figure 5.6 Torque curves computed using ANSYS and from neural network simulations (optimum design for MMF=50kA, λ/g=201.2, t_s/λ=0.419, t_r/λ=0.495)

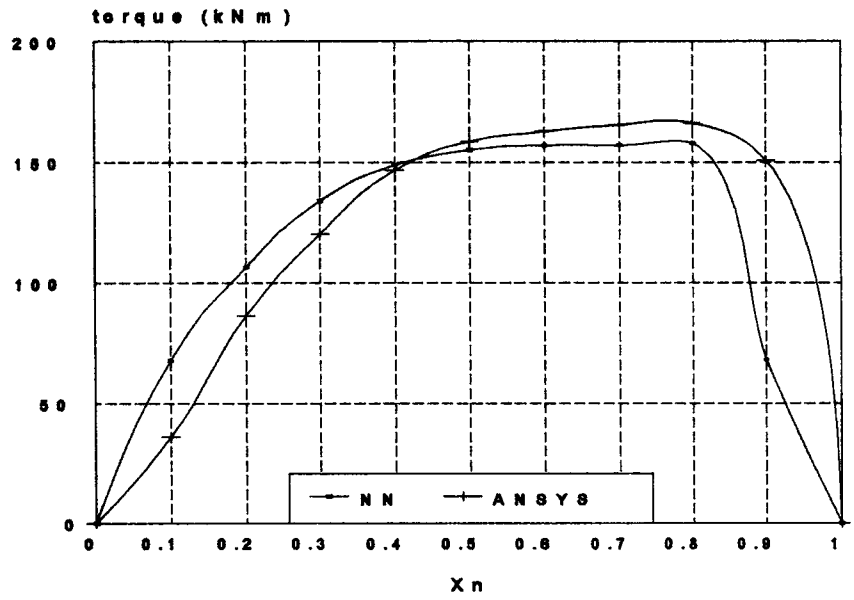


Figure 5.7 Torque curves computed using ANSYS and from neural network simulations (optimum design for MMF=60kA, λ/g=250, t_s/λ=0.4, t_r/λ=0.5)

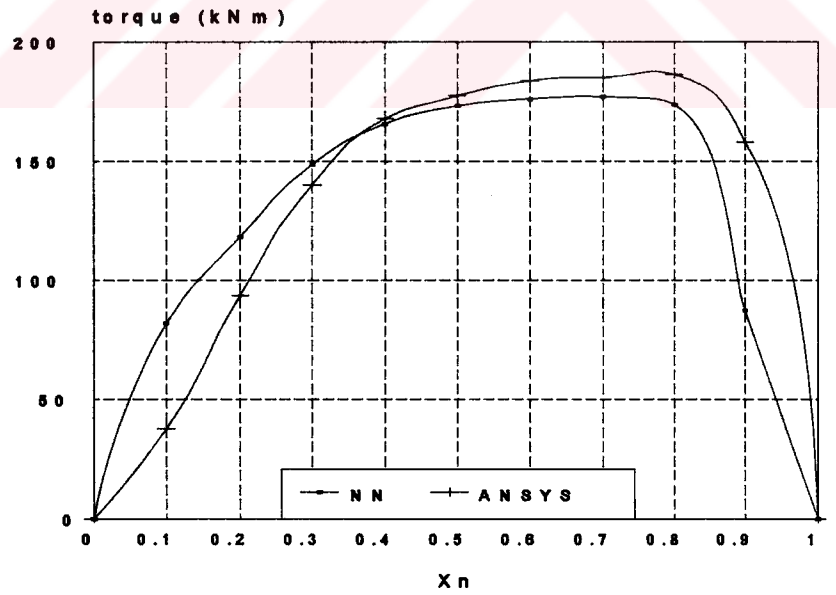


Figure 5.8 Torque curves computed using ANSYS and from neural network simulations (optimum design for MMF=70kA, λ/g=249.2, t_s/λ=0.374, t_r/λ=0.5)

Table 5.4 Torque ripple levels of optimum geometries calculated from neural network and ANSYS

MMF of optimum geometry (kA)	% Torque ripple (ANSYS calculation)	% Torque ripple (neural network calculation)
30	8	8.5
40	15.2	8.2
50	11.3	8.9
60	10.4	10.2
70	10.2	9.6

CHAPTER 6

CONCLUSIONS

6.1 General

In the first part of this work, a professional magnetic field solution software (ANSYS) is assessed for accuracy in predicting flux linkage, leakage flux and torque of a switched reluctance motor. For this purpose macros are developed. The effects of mesh distribution on both solution accuracy and torque and flux linkage calculations are investigated. Rules are laid out for obtaining accurate results. Measurements of the torque and flux linkage characteristics of SR motors are conducted. When measured data is compared with computed torque curves agreement is found to be very good. This study clearly displayed that magnetic field solution technique and the routines developed for torque and flux linkage calculation are reliable and can be used to verify the performance of an srm. However, it is found that, a typical problem with practical levels of saturation (approximately 8000 nodes) takes about 25 minutes on a HP-700 workstation.

Since the aim is to find the optimum motor structure with minimum torque ripple, a large number of solutions would be required. As a result of this it is easy to conclude that, using finite element analysis in an iterative optimization process is almost impossible due to time constraint. For this reason, force and permeance data which is numerically computed by Ertan[16,17], is used as a data base. This data is computed per unit length of a doubly salient structure with identical stator and rotor teeth, containing information about the nonlinear nature of the problem.

The problem then, is extracting the required information from this data set without loss of accuracy. Another important consideration is the time taken to reach the desired data point. This is very important, since the aim here is to optimize a three variable problem and access to this data will be required hundreds of times in a typical optimization procedure. For this reason, neural networks are found to be suitable for this problem. Neural networks not only have accurate prediction ability, but are suitable for an optimization algorithm, since once the network is trained finding a data point involves calculation of a simple analytical equation. If neural networks and other interpolation techniques, such as multi-input splines are compared, it can be observed that for a system which has many inputs (in our problem number of inputs is 5), neural networks are more advantageous. Because, for multi-input systems other techniques such as splines causes complexity in programming and they are also time consuming, since all two dimensional surfaces have to be defined separately. But in the usage of neural networks, there is no limitation on the dimensions of input and output vectors.

One important aspect about neural networks is the choice of the number of hidden layer neurons. The choice should be made such that the neural network recognizes the pattern underlying the training data with the lowest possible number of hidden layer neurons which is called as 'generalization'. With greater number of neurons than the number sufficient to generalize, the neural network 'memorizes' the training data which means that although the neural network seems to be trained perfectly, it makes large oscillations while being used as a device for interpolations between the data points in the training set. So the strategy for choosing the number of neurons was, to begin with a low number of hidden layer neurons and monitoring the interpolation performance on graphs, and than increasing the number of neurons gradually until the desired accuracy for accessing the data achieved.

As discussed in Chapter 3, three networks are trained. These are networks trained to represent force-mmf (500 data points in training set)and Bt (flux density)-mmf (880

data points) curves of symmetrically slotted geometries, and a third network trained to represent force vs. mmf variation of asymmetrically slotted structures (2037 data points).The first two networks are in fact only needed to construct the third network. The force vs. mmf network for asymmetrically slotted geometries is crucial since it simplifies the prediction process as described in Chapter 3. In this work, overall mean square error of 10^{-2} is taken to be sufficient for accuracy. On the other hand, each one of these networks is found to take about a month on a HP-700 workstation including corrections for portions of the data which were found to be poorly trained. This experience indicates that training a network is a tedious work if high accuracy is desired.

Using the approach[20] described in Chapter 3 and the force-mmf network for asymmetrically slotting, it became possible to predict the torque-position curves of any asymmetrically slotted geometry. It must be noted however that in the approach adopted here the excitation region is excluded from the calculation since the data also assumes that this region does not affect the force produced. For this reason, using this data essentially means neglecting the effect of the excitation winding, the space it occupies, also losses, temperature rise etc. This however is taken to be an acceptable assumption at the first stage of the problem.

Since our problem is a constraint optimization, the problem has to be converted into an unconstrained form where the constraints are automatically imposed. The two well-known methods to serve this aim is the Lagrangian method and the penalty method. In this work, the Augmented Lagrangian method, which is the combination of Lagrange and Penalty methods, is preferred since this combined method has the advantages of enlarged region of convergence and the requirement of fewer iterations for convergence. In this combined form, some problems related with penalty weights, such as divergence from the global minimum, is avoided. For minimization of the Augmented Lagrangian function Davidon Fletcher Powel method which is one of the quasi-Newton methods, is preferred. The reason for

using this sophisticated method is that; only the first derivatives of the function to be minimized with respect to each independent variable is required and the method is guarantees search direction to be in the direction of the descent. Then the design problem is formulated as torque ripple minimization problem with three variables and inequality constraints. Starting from 54 initial points, optimum variables are found . For convergence, approximately 300 iterations are required, which takes approximately about an hour.

Then, to get an insight to the problem for different mmf levels, optimum parameters that minimize torque ripple are found. The results are presented so that rather than specific optimum points intervals in which torque ripple is within 10% of the minimum. It is believed that this form of the data will prove useful for the designer. The results of this study are presented in Table 4.2 and figures Fig.4.4-5 and in Appendix B.In summary, for different mmf levels, the best results are found for t_s/λ_r and t_r/λ_r values of 0.4 and 0.5 respectively and the worst ripple for narrow teeth (t/λ_r=0.3) for both sides. The point at which minimum ripple occurs shift towards higher λ_r/g values as the mmf is increased within the interval 100-250.

Finally, to assure that the results of the optimization are meaningful, the static torque characteristics of optimum geometries found for different mmf levels are computed also using finite element analysis. The comparison given in Table 5.4 shows that the results of the optimization match the computations.

In this work, the optimum ripple conditions obtained rely on two basic assumptions;

1. The saturation in the back core is negligible,
2. The winding space is not a constraint on the mmf that can be applied to teeth region.

In practice both of these conditions play an important role in designing a practical device. Therefore it is necessary to incorporate these considerations into the design procedure.

However, the approach presented here clearly indicates how torque ripple can be minimized and what can be done to minimize it. The results given here are obviously not exact values, both because of the computation errors and assumptions involved, but indicate the general trend with changing parameters. So instead of using the parameters obtained from optimization directly as they appear, a better approach would be to examine the tables which show the ripple for various combinations of design parameters and assess several designs in the optimum range. The designers experience in deciding on various parameters and deciding on current densities is obviously important.

6.2 Future Work

In this thesis, the search for optimum geometry is limited to conditions which corresponds to low speed operation. Indeed this condition is the most important from the point of view of torque ripple since inertia effect is most observable. However, considering the medium speed range, also including back emf effects for torque ripple minimization is also worthy of investigation.

It may be also worthwhile to develop a torque ripple optimization approach to investigate how a more realistic set of o parameters can be found including the effect of winding space.

REFERENCES

1. Accernly, P.P., "Stepping Motors: A Guide to Modern Theory and Practice", Peter Peregrinus Ltd., London,1992.

2. Amin, B., "Analysis of the Torque Ripple and Design Considerations for Improving the Rate of Torque Ondulations in Variable Reluctance Motors", Pisa, Italy, 1988, pp:19-22

3. ANSYS Commands Manual, Swanson Analysis Systems, Revision 5.0, Houston, 1992.

4. ANSYS Procedures, Swanson Analysis Systems, Revision 5.0, Houston, 1992.

5. ANSYS Theoretical Manual, Swanson Analysis Systems, Revision 5.0, Houston, 1992.

6. ANSYS User's Manual, Swanson Analysis Systems, Revision 5.0, Houston, 1992.

7. Beşenek, N., "Numerically Calculated Force and Permeance Data for Doubly Salient Geometries", Ms. Thesis, METU, 1988.

8. Bertsekas, D.P., "Constrained Optimization and Lagrange Multiplier Methods", London, 1982.

9. Blenkinshop, P.T.,"A Novel Self Commutating Singly Excited Motor", Ph.D. Thesis, University of Leeds, 1976.

10. Cichocki, A., Unbehauen, R., "Neural Networks for Optimization and Signal Processing", John Wiley and Sons, England, 1993.

11. Cook, R.D., Malkus, D., Plesha, M. "Concepts and Applications of Finite Element Analysis", USA, 1989

12. Corda J., Stephenson, J.M., "Analytical Estimation of the Maximum and Minimum Inductances of Doubly-Salient Motor", Proc. of the International Conference on Stepping Motors and Systems, University of Leeds UK. pp 50-59 September 1979.

13. Corda, J., Stephenson, J.M., "Computation of Torque and Current in Doubly Salient Reluctance Motors from Nonlinear Magnetization Data", Proc.IEE, 1979, Vol.126, no.5, pp.393-396.

14. Corda, J., " Effects of the Form of Magnetic Circuits on Torque Pulsations of Switched Reluctance Motor", ICEM, Manchester, UK, 1990, pp:88-93.

15. Diriker, A., "Optimum Parameters of Asymmetrically Slotted Structures", M.S. Thesis, METU, 1987.

16. Ertan,H.B., Huges, A., Lawrenson, P.J., "Efficient Numerical Method for Predicting the Torque/Displacement Curve of Saturated VR Stepping Motors.",Proc. IEE, vol.127, No:4, pp 246-252, July 1980.

17. Ertan,H.B., "Analytical Prediction of Torque and Inductance Characteristics of Identically Slotted Doubly-Salient Reluctance Motors.",Proc. IEE, Vol.133, No.4 pp. 230-236, 1986.

18. Ertan, H.B., "Prediction of Torque and Permeance Displacement Characteristics of Asymmetrical Tooth Pairs", Proc.ICEM, Munich, Germany, pp:97-100, September 1986.

19. Ertan, H.B., "Asimetrik Çıkık Kutuplu Yapılarda Kuvvetin Hesabı", Doçentlik Tezi, 1982.

20. Hung, "Efficient Torque Ripple Minimization of Switched Reluctance Motors", July 1993, Montreal, IMACS-TC1, pp:2104-2109.

21. Kuo, C., Benjamin, "Theory and Applications of Step Motors", Boston, 1974

22. Lawrenson, P.,"Switched Reluctance Motor Drives", IEE Electronics and Power, February 1983, pp.144-147.

23. Lawrenson, P., Stephenson, J.M., Blenkinshop, P.T., Corda, J.,"Variable-speed Switched Reluctance Motors", Proc. IEE, Electrical Power Applications, 1980, Vol 127, no.4, pp.253-265.

24. Le Chanedac, Multon , Hassine, "Current Feeding of Switched Reluctance Motor, Optimization of Current Waveform to Minimize the Torque Ripple", IMACS-TC1, Montreal, July 1993, pp:267-272.

25. Luenberger, "Linear and Nonlinear Programming", Addison-Wesley Publishing Company, Washinghton, 1972.

26. Marinescu, M.,Marinescu, N., " Numerical Computation of Torques in Permanent Magnet Motors by Maxwell Stresses and Energy Method".IEEE, Transections on Magnetics, Vol. 24, No.1, January 1988, pp:463-466.

27. Mathews, J.H., "Numerical Methods", Prentice Hall International, UK, 1987.

28. Mellitt, B., Rashid, M.,H. "Voltage-Time Integral Method for Measuring Machine Inductance" Proc. IEE, Vol.121, No.9, September 1974, pp:1016-1017.

29. Moallem, M., Ong, C. M., "Predicting the Torque of a Switched Reluctance Machine from its Finite Element Field Solution" IEEE Transections on Energy Conversion, Vol. 5, No.4, December 1990, pp:733-739

30. O'Donavan, P.J., Roche, R.C., Kavanagh, M.G., Egan, J., Murphy, M.D., "Neural Network Based Torque Ripple Minimization in a Switched Reluctance Motor", IECON, Bologna, Italy, pp.1226-1231, 1994.

31. Ray, W.F., Davis, R.M., Lawrenson, P.J., Stephenson, J.M., Fulton, N.N., Blake, R.J., "Switched Reluctance Motor Drives forRail Traction: A Second View", Proc. IEE, Electrical Power Applications, 1984, Vol.131, no.5, pp.220-225.

32. Schramn, Williams,Green, "Optimum Commutation Current Profile on Torque Linearization of Switched Reluctance Motors", ICEM, Manchester, UK, 1992, pp:484-488.

33. Tohumcu, M., "Optimum Design of Switched Reluctance Motors", A Doctor of Philosophy Thesis, METU, 1985.

34. Tormey,D.P., Torrey, D.A.,"The Design of a Low Current Variable Reluctance Motor Drive with Constrained Torque Ripple", Int. Conf. on Electrical Machine, Massachusetts Institude of Technology,USA, pp. 788-793, August 1990.

35. Unnewehr, L.E., Koch,W.H., "An Axial Air-gap Reluctance Motor for Variable Speed Applications", IEEE Trans., 1974, pp.367-376.

36. Wallace, D.G., Taylor, "Three Phase Switched Reluctance Motor Design to Reduce Torque Ripple", Int. Conf. on Electrical Machine, Massachusetts Institute of Technology, USA, pp:783-787, August 1990.

37. Wallace, D.G., Taylor, "Low-Torque-Ripple Switched Reluctance Motors for Direct-Drive Robotic", IEEE Transections on Robotics, Vol:7, No:6, December 1991, pp:733-741.

38. Yağan, Ö., "A new Approach to the Design Optimization of Switched Reluctance Motor", M.S. Thesis, METU, 1990.

APPENDIX A

PROGRAMS AND DATA SETS RELATED WITH NEURAL NETWORK BASED FORCE CALCULATION OF ASYMMETRICALLY SLOTTED STRUCTURES

A.1. Artificial Neural Network Algorithm:

The following algorithm, which was written on C programming language, is used to train a neural network that recognizes the pattern underlying the data available. The algorithm was explained in Sec 3.5. In order to run the program succesfully, data file and related weight file have to be contained in the same directory.

NEURAL.C

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define boy 25

main()
{
/**definition of parameters**/
FILE *fp;  FILE *fout;  FILE *fyd;  FILE *fy;
double w[4][boy][boy],big,t,bt[boy][boy];
int gb,iter,a,b,c,i,j,jj,nn,l,kk,ii,s,ss,tt,h,say,N1,N2,iv,in,on;
int n,k,p,itn,dimu,dimy,dimx1,dimx2,dd,dim,fare,sayi,wsay,tel;
double cost,top,der,ex1,der1,out,cons,ex,ut,yt,sum;
double gb3[boy],gw3[boy][boy],gb2[boy],gw2[boy][boy],gb1[boy],gw1[boy][boy];
double GALM,GALM2,ALM,AL,A,B,Bk,ALMT,GAL2,dr,NORM,CST,pert,cold;
double *u,*y,*yd,*x1,*x2,*z1,*z2,*JJ,*GJ,*X,*d,*old,*old1;

/****  READ INPUT/OUTPUT DATA FILE  *****/
  if((fp=fopen("mmf-force.dat","r")) ==NULL) {
        printf("can not open the file\n");    exit(0);  }
/*mmf-force.dat training set file (or any training set) have to be contained in the same directory
with the program. This file have to contain inputs and outputs in the same raw.*/
```

```
printf("\nWhat is the number of input/output vectors? \n "); scanf("%d",&iv);
printf("What is the number of inputs ? \n");  scanf("%d",&in);
printf("What is the number of outputs ? \n"); scanf("%d",&on);
printf("What is the number of neurons in the first layer?\n"); scanf("%d",&N1);
printf("What is the number of neurons in the second layer?\n"); scanf("%d",&N2);
printf("Number of iterations=\n"); scanf("%d",&itn);
printf("random weights(0) or read from file(1)\n"); scanf("%d",&tel);
/*****************************************************************/
/*dynamic memory allocation part*/
dimu=iv*in; dimy=iv*on; dimx1=iv*N1; dimx2=iv*N2;
dim=in*N1+N1*N2+N2*on+N1+N2+on;

if((u=(double*) malloc(dimu*sizeof(double)))==NULL ) exit(1);
if((y=(double*) malloc(dimy*sizeof(double)))==NULL ) exit(1);
if((yd=(double*) malloc(dimy*sizeof(double)))==NULL ) exit(1);
if((x1=(double*) malloc(dimx1*sizeof(double)))==NULL ) exit(1);
if((z1=(double*) malloc(dimx1*sizeof(double)))==NULL ) exit(1);
if((x2=(double*) malloc(dimx2*sizeof(double)))==NULL ) exit(1);
if((z2=(double*) malloc(dimx2*sizeof(double)))==NULL ) exit(1);
if((JJ=(double*) malloc(dim*sizeof(double)))==NULL ) exit(1);
if((GJ=(double*) malloc(dim*sizeof(double)))==NULL ) exit(1);
if((X=(double*) malloc(dim*sizeof(double)))==NULL ) exit(1);
if((d=(double*) malloc(dim*sizeof(double)))==NULL ) exit(1);
if((old=(double*) malloc(dim*sizeof(double)))==NULL ) exit(1);
if((old1=(double*) malloc(dim*sizeof(double)))==NULL ) exit(1);

/*reading training set from the file mmf-force.dat*/
nn=1;
   for(i=1;i<1+iv;i++)
  {  jj=1;  for(kk=nn;kk<nn+in;kk++)  {fscanf(fp,"%lg",&ut);  a=in*(i-1)+jj; u[a]=ut;  jj=jj+1;  }
          l=1;
          for(kk=(nn+in);kk<nn+in+on;kk++)
          { fscanf(fp,"%lg",&yt);  b=on*(i-1)+l;  y[b]=yt; l=l+1; }    nn=nn+in+on;   }

/***  defining arbitrary weights  ***/
/* if the parameter 'tel' is 0 weights are not read from any file but defined arbirarily*/
big=0;  dr=100;  a=1;
for(b=1;b<1+in;b++) {for(c=1;c<1+N1;c++) {w[a][b][c]=rand();
 if(big< w[a][b][c]) big= w[a][b][c] ; }}
a=2;  for(b=1;b<1+N1;b++) {for(c=1;c<1+N2;c++) {w[a][b][c]=rand();
if(big< w[a][b][c]) big= w[a][b][c] ;}}        a=3;
for(b=1;b<1+N2;b++) { for(c=1;c<1+on;c++) {w[a][b][c]=rand(); if(big< w[a][b][c])
big= w[a][b][c] ; } }
a=1;  for(b=1;b<1+in;b++) { for(c=1;c<1+N1;c++) { w[a][b][c]=w[a][b][c]/(big*dr);  }}
a=2;  for(b=1;b<1+N1;b++){  for(c=1;c<1+N2;c++)   {w[a][b][c]=w[a][b][c]/(big*dr);}}
a=3;  for(b=1;b<1+N2;b++) { for(c=1;c<1+on;c++) {  w[a][b][c]=w[a][b][c]/(big*dr); }}
t=0; a=1;  for(c=1;c<1+N1;c++){ bt[a][c]=rand();  if(t< bt[a][c])  t= bt[a][c] ; }
a=2;  for(c=1;c<1+N2;c++) {  bt[a][c]=rand();  if(t< bt[a][c])  t= bt[a][c] ; }
a=3;  for(c=1;c<1+on;c++) {  bt[a][c]=rand();  if(t< bt[a][c]) t= bt[a][c] ; }
a=1; for(c=1;c<1+N1;c++)          {  bt[a][c]=bt[a][c]/(t*dr);}
a=2;  for(c=1;c<1+N2;c++) {bt[a][c]=bt[a][c]/(t*dr);}
a=3; for(c=1;c<1+on;c++) {bt[a][c]=bt[a][c]/(t*dr);}

/******* parameter initialization *******/
```

```
gb=in*N1+N1*N2+N2*on+N1+N2+on; say=0; Bk=0; cold=0;
for(a=1;a<gb+1;a++) { old[a]=0; old1[a]=0; }
wsay=0; AL=1; sayi=0;
if(tel>=1) /*if weights are read from file which is previously obtained*/
{
if((fyd=fopen("weight.dat","r"))==NULL) { printf("can not open the file\n"); exit(0); }
fscanf(fyd,"%d %d %d %d\n",&in ,&N1,&N2, &on);
for(i=1;i<in+1;i++) { for(k=1;k<N1+1;k++) { fscanf(fyd,"%lg \n",&w[1][i][k]);}}
for(k=1;k<N1+1;k++) { for(p=1;p<N2+1;p++) { fscanf(fyd,"%lg \n",&w[2][k][p]); }}
for(p=1;p<N2+1;p++) { for(j=1;j<on+1;j++)          { fscanf(fyd,"%lg \n",&w[3][p][j]); }}
for(k=1;k<N1+1;k++) { fscanf(fyd," %lg \n",&bt[1][k]); }
for(p=1;p<N2+1;p++) { fscanf(fyd,"%lg \n",&bt[2][p]); }
for(j=1;j<on+1;j++) { fscanf(fyd,"%lg \n",&bt[3][j]);          }
}

/******************************** ITERATIONS ************************************/
for(iter=1;iter<itn;iter++)
{ /* for temporary weights output of the network is calculated and error is obtained */
/***          OUTPUT VECTOR CALCULATION          *****/
cons=10;
for(ii=1;ii<iv+1;ii++)
   {
 /***    CALCULATION OF THE STATE VECTORS X1 AND Z1    ***/
        for(k=1;k<N1+1;k++)
        { sum=0;   for(n=1;n<in+1;n++)
                { a=in*(ii-1)+n;    sum=w[1][n][k]*u[a]+sum; }
                c=N1*(ii-1)+k;  x1[c]=sum+bt[1][k];
/*** SIGMOID OPERATION ***/          z1[c]=tanh(cons*x1[c]); }
/***    CALCULATION OF THE STATE VECTORS X2 AND Z2    ***/
for(p=1;p<N2+1;p++)
        {sum=0;   for(k=1;k<N1+1;k++) { c=N1*(ii-1)+k;          sum=w[2][k][p]*z1[c]+sum;
}
                b=N2*(ii-1)+p;   x2[b]=sum+bt[2][p];
        /*** SIGMOID OPERATION ***/   z2[b]=tanh(cons*x2[b]); }
/***      CALCULATION OF THE OUTPUT VECTORS          ***/
for(j=1;j<on+1;j++)
        {sum=0;  for(p=1;p<N2+1;p++) { b=N2*(ii-1)+p; sum=w[3][p][j]*z2[b]+sum; }
                dd=on*(ii-1)+j;  yd[dd]=sum+bt[3][j]; }}
                 /***     COST FUNCTION     ***/
cold=cost;  sum=0;
for(i=1;i<iv+1;i++)
        { for(j=1;j<on+1;j++) { dd=on*(i-1)+j;
                cost=(y[dd]-yd[dd])*(y[dd]-yd[dd])+sum;
                sum=cost; }} cost=cost/2;
if(((cold-cost)<0.001)&&(iter>2) ) pert=1; /* if cost doesn`tchange more,
   printf("\nCOST = %lg  iter= %d \n",cost,iter);
/*****writing the weights to an output file after every 10 iteration****/
wsay++;
if (wsay>10)    {
for(ii=1;ii<iv+1;ii++) { for(j=1;j<on+1;j++) {dd=on*(ii-1)+j;
printf("\n out= %lg  out= %lg",yd[dd] ,y[dd]);}}
if((fy=fopen("wght_temp.dat","w"))==NULL) {
        printf("can not open the file\n");
        exit(0);  }
```

```
fprintf(fy,"%d %d %d %d\n",in ,N1, N2, on);
for(i=1;i<in+1;i++) { for(k=1;k<N1+1;k++) {fprintf(fy,"%lg \n",w[1][i][k]); }}
for(k=1;k<N1+1;k++) { for(p=1;p<N2+1;p++) { fprintf(fy,"%lg \n",w[2][k][p]);          }}
for(p=1;p<N2+1;p++){ for(j=1;j<on+1;j++) { fprintf(fy,"%lg \n",w[3][p][j]); }}
for(k=1;k<N1+1;k++) { fprintf(fy," %lg \n",bt[1][k]);          }
for(p=1;p<N2+1;p++) { fprintf(fy,"%lg \n",bt[2][p]);          }
for(j=1;j<on+1;j++) { fprintf(fy,"%lg \n",bt[3][j]); }
wsay=0; }
          /**** GRADIENT VECTOR CALCULATION ****/
for(j=1;j<on+1;j++)
 { sum=0; for(i=1;i<iv+1;i++){dd=on*(i-1)+j;gb3[j]=(yd[dd]-y[dd])+sum; sum=gb3[j]; }}
for(p=1;p<N2+1;p++) {for(j=1;j<on+1;j++){ sum=0; for(i=1;i<iv+1;i++){ dd=on*(i-1)+j;
        b=N2*(i-1)+p; gw3[p][j]=(yd[dd]-y[dd])*z2[b] + sum; sum=gw3[p][j]; }}}
for(p=1;p<N2+1;p++) { sum=0; for(j=1;j<on+1;j++) { for(i=1;i<iv+1;i++) { dd=on*(i-1)+j;
        b=N2*(i-1)+p; ex=((1+exp(-2*cons*x2[b]))); ex1=((1-exp(-2*cons*x2[b])));
        der=((2*cons*exp(-2*cons*x2[b])*ex)+(2*cons*exp(-2*cons*x2[b])*ex1))/(ex*ex);
        gb2[p]=(yd[dd]-y[dd])*w[3][p][j]*der+sum; sum=gb2[p]; }}}
for(k=1;k<N1+1;k++)
{ for(p=1;p<N2+1;p++) {          sum=0; for(j=1;j<on+1;j++) {for(i=1;i<iv+1;i++) {
        dd=on*(i-1)+j;  b=N2*(i-1)+p; c=N1*(i-1)+k;
        ex=((1+exp(-2*cons*x2[b]))); ex1=((1-exp(-2*cons*x2[b])));
        der=((2*cons*exp(-2*cons*x2[b])*ex)+(2*cons*exp(-2*cons*x2[b])*ex1))/(ex*ex);
        gw2[k][p]=(yd[dd]-y[dd])*w[3][p][j]*z1[c]*der+sum;  sum=gw2[k][p]; }}}}
for(k=1;k<N1+1;k++)
{ sum=0;  for(p=1;p<N2+1;p++) {for(j=1;j<on+1;j++) { for(i=1;i<iv+1;i++) { dd=on*(i-1)+j;
        b=N2*(i-1)+p; c=N1*(i-1)+k;  ex=((1+exp(-2*cons*x2[b])));
        der=(4*cons*exp(-2*cons*x2[b]))/(ex*ex);
        ex1=((1+exp(-2*cons*x1[c]))); der1=(4*cons*exp(-2*cons*x1[c]))/(ex1*ex1);
        gb1[k]=(yd[dd]-y[dd])*w[3][p][j]*w[2][k][p]*der*der1+sum;  sum=gb1[k]; }}}}
for(n=1;n<in+1;n++)
{ for(k=1;k<N1+1;k++) { sum=0; for(p=1;p<N2+1;p++) { for(j=1;j<on+1;j++) {
  for(i=1;i<iv+1;i++) {dd=on*(i-1)+j; b=N2*(i-1)+p;          c=N1*(i-1)+k; a=in*(i-1)+n;
        ex=((1+exp(-2*cons*x2[b]))); ex1=((1-exp(-2*cons*x2[b])));
        der=((2*cons*exp(-2*cons*x2[b])*ex)+(2*cons*exp(-2*cons*x2[b])*ex1))/(ex*ex);
        ex=((1+exp(-2*cons*x1[c]))); ex1=((1-exp(-2*cons*x1[c])));
        der1=((2*cons*exp(-2*cons*x1[c])*ex)+(2*cons*exp(-2*cons*x1[c])*ex1))/(ex*ex);
        gw1[n][k]=(yd[dd]-y[dd])*w[3][p][j]*w[2][k][p]*der*der1*u[a]+sum;
        sum=gw1[n][k];  }} } } }
 /****     CALCULATION OF NORM       ****/
sum=0;
for(j=1;j<on+1;j++)  {top=gb3[j]*gb3[j]+sum; sum=top; }
for(p=1;p<N2+1;p++) {for(j=1;j<on+1;j++)          {top=gw3[p][j]*gw3[p][j]+sum; sum=top; }}
for(p=1;p<N2+1;p++) {top=gb2[p]*gb2[p]+sum;sum=top; }
for(k=1;k<N1+1;k++) {for(p=1;p<N2+1;p++) {top=gw2[k][p]*gw2[k][p]+sum;          sum=top;
}}
for(k=1;k<N1+1;k++) {top=gb1[k]*gb1[k]+sum; sum=top; }
for(n=1;n<in+1;n++) {for(k=1;k<N1+1;k++) {top=gw1[n][k]*gw1[n][k]+sum;
        sum=top; }}
NORM=sqrt(sum);    printf("\n NORM = %lg",NORM);
 /***** Determination of Gradient Vector *****/
a=1;  for(n=1;n<in+1;n++) { for(k=1;k<N1+1;k++) {          GJ[a]=gw1[n][k]; a=a+1; }}
for(k=1;k<N1+1;k++)  {for(p=1;p<N2+1;p++) {GJ[a]=gw2[k][p]; a=a+1; }}
for(p=1;p<N2+1;p++) {for(j=1;j<on+1;j++) {GJ[a]=gw3[p][j]; a=a+1; }}
for(k=1;k<N1+1;k++) {GJ[a]=gb1[k]; a=a+1; }
```

```
for(p=1;p<N2+1;p++) {GJ[a]=gb2[p]; a=a+1; }
for(j=1;j<on+1;j++) {GJ[a]=gb3[j]; a=a+1; }
/******* X vector *******/
a=1; for(n=1;n<in+1;n++) {for(k=1;k<N1+1;k++) {        X[a]=w[1][n][k];        a=a+1; }}
for(k=1;k<N1+1;k++) {for(p=1;p<N2+1;p++) { X[a]=w[2][k][p]; a=a+1; }}
for(p=1;p<N2+1;p++) {for(j=1;j<on+1;j++) {X[a]=w[3][p][j]; a=a+1; }}
for(k=1;k<N1+1;k++){X[a]=bt[1][k]; a=a+1; }
for(p=1;p<N2+1;p++) {X[a]=bt[2][p]; a=a+1; }
for(j=1;j<on+1;j++) {X[a]=bt[3][j]; a=a+1; }
for(a=1;a<gb+1;a++) {d[a]=GJ[a]-(Bk*(old[a]-old1[a])); }
for(a=1;a<gb+1;a++) {old1[a]=old[a]; old[a]=X[a];}

                    /****   ONE DIMENSIONAL SEARCH PART   ****/
ALM=5/NORM; /* search parameter ∝ is chosen initially */
for(h=1;h<20;h++) /* iterations of one dimensional searc part, limited to 20*/
 { BIR:
   for(a=1;a<gb+1;a++) {X[a]=old[a]-(ALM*d[a]); }
a=1; for(n=1;n<in+1;n++) { for(k=1;k<N1+1;k++) {        w[1][n][k]=X[a];        a=a+1; }}
for(k=1;k<N1+1;k++) {for(p=1;p<N2+1;p++) { w[2][k][p]=X[a]; a=a+1; }}
for(p=1;p<N2+1;p++) { for(j=1;j<on+1;j++) {w[3][p][j]=X[a]; a=a+1; }}
for(k=1;k<N1+1;k++) {bt[1][k]=X[a]; a=a+1; }
for(p=1;p<N2+1;p++) { bt[2][p]=X[a]; a=a+1;}
for(j=1;j<on+1;j++) { bt[3][j]=X[a]; a=a+1; }
   /*************************************/
for(ii=1;ii<1+iv;ii++)
{for(k=1;k<N1+1;k++) {sum=0;for(n=1;n<in+1;n++) { a=in*(ii-1)+n; sum=w[1][n][k]*u[a]+sum;
        } c=N1*(ii-1)+k; x1[c]=sum+bt[1][k]; z1[c]=tanh(cons*x1[c]); }
for(p=1;p<N2+1;p++) { sum=0; for(k=1;k<N1+1;k++) {
        c=N1*(ii-1)+k; sum=w[2][k][p]*z1[c]+sum; }
        b=N2*(ii-1)+p; x2[b]=sum+bt[2][p]; z2[b]=tanh(cons*x2[b]); }
for(j=1;j<on+1;j++) { sum=0; for(p=1;p<N2+1;p++) { b=N2*(ii-1)+p;
        sum=w[3][p][j]*z2[b]+sum; }
        dd=on*(ii-1)+j;  yd[dd]=sum+bt[3][j]; }}
 CST=0;  for(i=1;i<iv+1;i++) { for(j=1;j<on+1;j++) { dd=on*(i-1)+j;
        CST=(y[dd]-yd[dd])*(y[dd]-yd[dd])+sum;  sum=CST;} }  GALM=CST/2;
for(a=1;a<gb+1;a++) { X[a]=old[a]-((ALM*d[a])/2); }
a=1; for(n=1;n<in+1;n++) {for(k=1;k<N1+1;k++) {        w[1][n][k]=X[a]; a=a+1; }}
for(k=1;k<N1+1;k++) {for(p=1;p<N2+1;p++) {w[2][k][p]=X[a]; a=a+1; }}
for(p=1;p<N2+1;p++) {for(j=1;j<on+1;j++) {w[3][p][j]=X[a];        a=a+1; }}
for(k=1;k<N1+1;k++) {bt[1][k]=X[a]; a=a+1; }
for(p=1;p<N2+1;p++) {bt[2][p]=X[a]; a=a+1; }
for(j=1;j<on+1;j++) { bt[3][j]=X[a]; a=a+1; }
                /*********************************/
for(ii=1;ii<1+iv;ii++) {
for(k=1;k<N1+1;k++) { sum=0; for(n=1;n<in+1;n++) { a=in*(ii-1)+n;
        sum=w[1][n][k]*u[a]+sum; }
        c=N1*(ii-1)+k;  x1[c]=sum+bt[1][k]; z1[c]=tanh(cons*x1[c]); }
for(p=1;p<N2+1;p++)
        {sum=0; for(k=1;k<N1+1;k++) { c=N1*(ii-1)+k; sum=w[2][k][p]*z1[c]+sum; }
        x2[c]=sum+bt[2][p];  b=N2*(ii-1)+p; z2[b]=tanh(cons*x2[b]); }
for(j=1;j<on+1;j++) { sum=0; for(p=1;p<N2+1;p++)        { b=N2*(ii-1)+p;
        sum=w[3][p][j]*z2[b]+sum; }      dd=on*(ii-1)+j; yd[dd]=sum+bt[3][j]; } }
 CST=0;  for(i=1;i<iv+1;i++) {for(j=1;j<on+1;j++) { dd=on*(i-1)+j;
        CST=(y[dd]-yd[dd])*(y[dd]-yd[dd])+sum;  sum=CST; } } GALM2=CST/2;
```

```
ALMT=ALM; if(cost<=GALM2) {ALM=ALM/2; }
if(cost>GALM2) { if(GALM<=GALM2) { ALM=2*ALM; }}
/*if(GALM>GALM2) goto LOOP;*/ if((cost>GALM2)&&(GALM>GALM2)) goto LOOP;}
LOOP:          /**quadratik fit, new value of ∝ is found from -B/2A where f=Ax^2+Bx+C**/
        B=(4*GALM2-GALM-3*cost)/ALM;
        A=(GALM-cost-ALM*B)/(ALM*ALM);
        AL=-B/(2*A);
for(a=1;a<gb+1;a++) { X[a]=old[a]-(AL*d[a]); }
a=1; for(n=1;n<in+1;n++) {for(k=1;k<N1+1;k++) {w[1][n][k]=X[a]; a=a+1; }}
for(k=1;k<N1+1;k++) {for(p=1;p<N2+1;p++) {w[2][k][p]=X[a]; a=a+1; }}
for(p=1;p<N2+1;p++) {for(j=1;j<on+1;j++) {w[3][p][j]=X[a]; a=a+1;}}
for(k=1;k<N1+1;k++) {bt[1][k]=X[a]; a=a+1; }for(p=1;p<N2+1;p++) {bt[2][p]=X[a]; a=a+1; }
for(j=1;j<on+1;j++) {bt[3][j]=X[a]; a=a+1;}
for(ii=1;ii<1+iv;ii++)
{ for(k=1;k<N1+1;k++) { sum=0; for(n=1;n<in+1;n++) { a=in*(ii-1)+n;
sum=w[1][n][k]*u[a]+sum; }
        c=N1*(ii-1)+k; x1[c]=sum+bt[1][k]; z1[c]=tanh(cons*x1[c]);        }
for(p=1;p<N2+1;p++) { sum=0;for(k=1;k<N1+1;k++){ c=N1*(ii-1)+k;
        sum=w[2][k][p]*z1[c]+sum; }
        b=N2*(ii-1)+p; x2[b]=sum+bt[2][p];    z2[b]=tanh(cons*x2[b]);}
for(j=1;j<on+1;j++) {sum=0; for(p=1;p<N2+1;p++) {b=N2*(ii-1)+p;
        sum=w[3][p][j]*z2[b]+sum;}
        dd=on*(ii-1)+j; yd[dd]=sum+bt[3][j];}}
CST=0; for(i=1;i<iv+1;i++) {for(j=1;j<on+1;j++){dd=on*(i-1)+j;CST=(y[dd]-yd[dd])*(y[dd]-
        yd[dd])+sum; sum=CST; }}GAL2=CST/2;
if(GAL2>GALM2) AL=ALM/2;
if((cost<GAL2) && (GAL2<GALM)) AL=0.05*ALM; if (AL<1e-5) AL=1e-5;
printf("AL= %lg\n", AL);
for(a=1;a<gb+1;a++){X[a]=old[a]-AL*d[a]; }
a=1; for(n=1;n<in+1;n++) { for(k=1;k<N1+1;k++) {w[1][n][k]=X[a]; a=a+1; }}
for(k=1;k<N1+1;k++){for(p=1;p<N2+1;p++) {w[2][k][p]=X[a];    a=a+1; }}
for(p=1;p<N2+1;p++) {for(j=1;j<on+1;j++) {w[3][p][j]=X[a]; a=a+1; }}
for(k=1;k<N1+1;k++) { bt[1][k]=X[a]; a=a+1; }for(p=1;p<N2+1;p++) {        bt[2][p]=X[a];
        a=a+1; }
for(j=1;j<on+1;j++) {bt[3][j]=X[a]; a=a+1; }fare=0; say++;
if(say>sayi) { printf("\n Write the value of Momentum constant: \n"); scanf("%lg",&Bk);
fare=1; say=0; } if (fare>=1){printf("iteration number=\n"); scanf("%d",&sayi); }
}
SON:
say=0; }
```

A.2 Data Sets Used for Training the Networks

1) Data set used for training network NN1 (Sec.3.8.1) for computation of force-mmf curves for symmetrically slotted structures ($\lambda/g=0.0172$m) is presented below. As discussed in Sec.3.8.1, inputs to this network are λ/g, t/λ, x_n and mmf (F), and output is force (S).

λ/g	t/λ	x_n	F	S	λ/g	t/λ	x_n	F	S
40	0.5	0	302	0	40	0.4	0.6	302	109.7
40	0.5	0	481	0	40	0.4	0.6	481	275.9
40	0.5	0	825	0	40	0.4	0.6	825	719.2
40	0.5	0	1235	0	40	0.4	0.6	1235	1215
40	0.4	0	302	0	40	0.3	0.6	302	108
40	0.4	0	481	0	40	0.3	0.6	481	275
40	0.4	0	825	0	40	0.3	0.6	825	705
40	0.4	0	1235	0	40	0.3	0.6	1235	1210
40	0.3	0	302	0	40	0.5	0.8	302	90.8
40	0.3	0	481	0	40	0.5	0.8	481	229.4
40	0.3	0	825	0	40	0.5	0.8	825	582.7
40	0.3	0	1235	0	40	0.5	0.8	1235	931.7
40	0.5	1	302	0	40	0.4	0.8	302	100
40	0.5	1	481	0	40	0.4	0.8	481	252
40	0.5	1	825	0	40	0.4	0.8	825	695.8
40	0.5	1	1235	0	40	0.4	0.8	1235	1137
40	0.4	1	302	0	40	0.3	0.8	302	20.6
40	0.4	1	481	0	40	0.3	0.8	481	52.2
40	0.4	1	825	0	40	0.3	0.8	825	153.8
40	0.4	1	1235	0	40	0.3	0.8	1235	334.4
40	0.3	1	302	0	70	0.5	0	175	0
40	0.3	1	481	0	70	0.5	0	470	0
40	0.3	1	825	0	70	0.5	0	707	0
40	0.3	1	1235	0	70	0.4	0	175	0
40	0.5	0.2	302	95.6	70	0.4	0	220	0
40	0.5	0.2	481	233.5	70	0.4	0	470	0
40	0.5	0.2	825	386.9	70	0.4	0	707	0
40	0.5	0.2	1235	515.8	70	0.3	0	175	0
40	0.4	0.2	302	99.	70	0.3	0	220	0
40	0.4	0.2	481	245.7	70	0.3	0	470	0
40	0.4	0.2	825	396.2	70	0.3	0	707	0
40	0.4	0.2	1235	532.6	70	0.5	1	175	0
40	0.3	0.2	302	99.1	70	0.5	1	220	0
40	0.3	0.2	481	245.7	70	0.5	1	470	0
40	0.3	0.2	825	389.7	70	0.5	1	707	0
40	0.3	0.2	1235	490.5	70	0.4	1	175	0
40	0.5	0.4	302	103	70	0.4	1	275	0
40	0.5	0.4	481	261	70	0.4	1	469	0
40	0.5	0.4	825	620.5	70	0.4	1	707	0
40	0.5	0.4	1235	930.8	70	0.4	1	1235	0
40	0.4	0.4	302	106.6	70	0.3	1	175	0
40	0.4	0.4	481	270	70	0.3	1	275	0
40	0.4	0.4	825	632	70	0.3	1	470	0
40	0.4	0.4	1235	943.6	70	0.3	1	707	0
40	0.3	0.4	302	109.7	70	0.3	1	235	0
40	0.3	0.4	481	276.8	70	0.5	0.2	175	64.4
40	0.3	0.4	825	722.7	70	0.5	0.2	220	101.9
40	0.3	0.4	1235	1008	70	0.5	0.2	470	301.6
40	0.5	0.6	302	102	70	0.5	0.2	707	454.4
40	0.5	0.6	481	258	70	0.4	0.2	175	65.7
40	0.5	0.6	825	676.9	70	0.4	0.2	220	103.7
40	0.5	0.6	1235	1103.8	70	0.4	0.2	470	319.8

λ/g	t/λ	x_n	F	S	λ/g	t/λ	x_n	F	S
70	0.4	0.2	707	481.5	100	0.3	0	125	0
70	0.3	0.2	175	65.6	100	0.3	0	200	0
70	0.3	0.2	220	103.6	100	0.3	0	350	0
70	0.3	0.2	470	333	100	0.3	0	525	0
70	0.3	0.2	707	485.8	100	0.5	1	125	0
70	0.5	0.4	175	67.5	100	0.5	1	200	0
70	0.5	0.4	220	106.6	100	0.5	1	350	0
70	0.5	0.4	470	425	100	0.5	1	525	0
70	0.5	0.4	707	724.6	100	0.4	1	125	0
70	0.4	0.4	175	68.6	100	0.4	1	200	0
70	0.4	0.4	220	108.3	100	0.4	1	350	0
70	0.4	0.4	470	443	100	0.4	1	525	0
70	0.4	0.4	707	764.7	100	0.3	1	125	0
70	0.3	0.4	175	69.6	100	0.3	1	200	0
70	0.3	0.4	220	110	100	0.3	1	350	0
70	0.3	0.4	469	453	100	0.3	1	525	0
70	0.3	0.4	707	807.9	100	0.5	0.2	125	50.59
70	0.5	0.6	175	69	100	0.5	0.2	200	128
70	0.5	0.6	275	167	100	0.5	0.2	350	283.5
70	0.5	0.6	470	431.6	100	0.5	0.2	525	468
70	0.5	0.6	707	766	100	0.4	0.2	125	51
70	0.4	0.6	175	70	100	0.4	0.2	200	128.5
70	0.4	0.6	275	172	100	0.4	0.2	350	284.5
70	0.4	0.6	470	449.29	100	0.4	0.2	525	454.4
70	0.4	0.6	707	802.7	100	0.3	0.2	125	51
70	0.3	0.6	175	67	100	0.3	0.2	200	129.4
70	0.3	0.6	275	165	100	0.3	0.2	350	308.3
70	0.3	0.6	470	481.2	100	0.3	0.2	525	499.6
70	0.3	0.6	707	836.8	100	0.5	0.4	125	52.2
70	0.5	0.8	175	62.7	100	0.5	0.4	200	133
70	0.5	0.8	275	153	100	0.5	0.4	350	350
70	0.5	0.8	470	399.5	100	0.5	0.4	525	618.5
70	0.5	0.8	707	689.9	100	0.4	0.4	125	52.6
70	0.4	0.8	175	66.2	100	0.4	0.4	200	134.2
70	0.4	0.8	275	162	100	0.4	0.4	350	358.1
70	0.4	0.8	470	459	100	0.4	0.4	525	638
70	0.4	0.8	707	793	100	0.3	0.4	125	53.1
70	0.4	0.8	900	987	100	0.3	0.4	200	135.3
70	0.4	0.8	1235	1340	100	0.3	0.4	350	364.8
70	0.3	0.8	175	7.9	100	0.3	0.4	525	647
70	0.3	0.8	275	18	100	0.5	0.6	125	51.7
70	0.3	0.8	470	56.6	100	0.5	0.6	200	131.7
70	0.3	0.8	707	128.1	100	0.5	0.6	350	351.3
70	0.3	0.8	1235	375.9	100	0.5	0.6	525	622.3
100	0.5	0	125	0	100	0.4	0.6	125	52.8
100	0.5	0	200	0	100	0.4	0.6	200	134.7
100	0.5	0	350	0	100	0.4	0.6	350	361
100	0.5	0	525	0	100	0.4	0.6	525	637.9
100	0.4	0	125	0	100	0.3	0.6	125	51.5
100	0.4	0	200	0	100	0.3	0.6	200	131.3
100	0.4	0	350	0	100	0.3	0.6	350	375.7
100	0.4	0	525	0	100	0.3	0.6	525	619.7

λ/g	t/λ	x_n	F	S	λ/g	t/λ	x_n	F	S
100	0.5	0.8	125	49.2	150	0.5	0.4	380	471.8
100	0.5	0.8	200	125.4	150	0.4	0.4	90	41.6
100	0.5	0.8	350	333.5	150	0.4	0.4	150	113.6
100	0.5	0.8	525	576.2	150	0.4	0.4	250	266.8
100	0.4	0.8	125	50.3	150	0.4	0.4	380	479.1
100	0.4	0.8	200	128.7	150	0.3	0.4	90	41.6
100	0.4	0.8	350	366.5	150	0.3	0.4	150	113.8
100	0.4	0.8	525	598.5	150	0.3	0.4	250	270.6
100	0.3	0.8	125	4.2	150	0.3	0.4	380	481.8
100	0.3	0.8	200	10.7	150	0.5	0.6	90	41.3
100	0.3	0.8	350	32.9	150	0.5	0.6	150	112.6
100	0.3	0.8	525	73.9	150	0.5	0.6	250	262.7
100	0.3	0.8	1200	268	150	0.5	0.6	380	470.5
150	0.5	0	90	0	150	0.4	0.6	90	41.6
150	0.5	0	150	0	150	0.4	0.6	150	113.8
150	0.5	0	250	0	150	0.4	0.6	250	270.2
150	0.5	0	380	0	150	0.4	0.6	380	480.7
150	0.4	0	90	0	150	0.3	0.6	90	40.6
150	0.4	0	150	0	150	0.3	0.6	150	112.5
150	0.4	0	250	0	150	0.3	0.6	250	283.2
150	0.4	0	380	0	150	0.3	0.6	380	461.2
150	0.3	0	90	0	150	0.5	0.8	90	39.6
150	0.3	0	150	0	150	0.5	0.8	150	108.1
150	0.3	0	250	0	150	0.5	0.8	250	254.5
150	0.3	0	380	0	150	0.5	0.8	380	443.8
150	0.5	1	90	0	150	0.4	0.8	90	40.0
150	0.5	1	150	0	150	0.4	0.8	150	111.2
150	0.5	1	250	0	150	0.4	0.8	250	279.8
150	0.5	1	380	0	150	0.4	0.8	380	453
150	0.4	1	90	0	150	0.3	0.8	90	2.1
150	0.4	1	150	0	150	0.3	0.8	150	5.8
150	0.4	1	250	0	150	0.3	0.8	250	16.2
150	0.4	1	380	0	150	0.3	0.8	380	37.4
150	0.3	1	90	0	150	0.3	0.8	900	210
150	0.3	1	150	0	200	0.5	0	70	0
150	0.3	1	250	0	200	0.5	0	110	0
150	0.3	1	380	0	200	0.5	0	190	0
150	0.5	0.2	90	40.7	200	0.5	0	270	0
150	0.5	0.2	150	109.3	200	0.4	0	70	0
150	0.5	0.2	250	229.5	200	0.4	0	110	0
150	0.5	0.2	380	392	200	0.4	0	190	0
150	0.4	0.2	90	40.8	200	0.4	0	270	0
150	0.4	0.2	150	110	200	0.3	0	70	0
150	0.4	0.2	250	234.3	200	0.3	0	110	0
150	0.4	0.2	380	400	200	0.3	0	190	0
150	0.3	0.2	90	40.8	200	0.3	0	270	0
150	0.3	0.2	150	110.8	200	0.5	1	70	0
150	0.3	0.2	250	234.3	200	0.5	1	110	0
150	0.3	0.2	379	414.9	200	0.5	1	190	0
150	0.5	0.4	90	41.4	200	0.5	1	270	0
150	0.5	0.4	150	112.7	200	0.4	1	70	0
150	0.5	0.4	250	262.7	200	0.4	1	110	0

λ/g	t/λ	x_n	F	S	λ/g	t/λ	x_n	F	S
200	0.4	1	190	0	200	0.3	0.8	190	8
200	0.4	1	270	0	200	0.3	0.8	270	16
200	0.3	1	70	0	200	0.3	0.8	550	66.2
200	0.3	1	110	0	200	0.3	0.8	700	107.5
200	0.3	1	190	0	250	0.5	0	55	0
200	0.3	1	270	0	250	0.5	0	90	0
200	0.5	0.2	70	33.2	250	0.5	0	160	0
200	0.5	0.2	110	80.2	250	0.5	0	230	0
200	0.5	0.2	190	189.8	250	0.4	0	55	0
200	0.5	0.2	270	306.4	250	0.4	0	90	0
200	0.4	0.2	70	33.3	250	0.4	0	160	0
200	0.4	0.2	110	80.5	250	0.4	0	230	0
200	0.4	0.2	190	192.8	250	0.3	0	55	0
200	0.4	0.2	270	311	250	0.3	0	90	0
200	0.3	0.2	70	33.3	250	0.3	0	160	0
200	0.3	0.2	110	80.5	250	0.3	0	230	0
200	0.3	0.2	190	192.8	250	0.5	1	55	0
200	0.3	0.2	270	311	250	0.5	1	90	0
200	0.5	0.4	70	33.6	250	0.5	1	160	0
200	0.5	0.4	110	81.7	250	0.5	1	230	0
200	0.5	0.4	190	200.8	250	0.4	1	55	0
200	0.5	0.4	270	329.8	250	0.4	1	90	0
200	0.4	0.4	70	33.7	250	0.4	1	160	0
200	0.4	0.4	110	82.1	250	0.4	1	230	0
200	0.4	0.4	190	204.6	250	0.3	1	55	0
200	0.4	0.4	270	336.9	250	0.3	1	90	0
200	0.3	0.4	70	33.8	250	0.3	1	160	0
200	0.3	0.4	110	82.5	250	0.3	1	230	0
200	0.3	0.4	190	209	250	0.5	0.2	55	25.8
200	0.3	0.4	270	342.7	250	0.5	0.2	90	67.4
200	0.5	0.6	70	33.6	250	0.5	0.2	160	168
200	0.5	0.6	110	81.7	250	0.5	0.2	230	276.2
200	0.5	0.6	190	203.3	250	0.4	0.2	55	25.8
200	0.5	0.6	270	333.9	250	0.4	0.2	90	67.6
200	0.4	0.6	70	33.8	250	0.4	0.2	160	169.6
200	0.4	0.6	110	82.5	250	0.4	0.2	230	278.4
200	0.4	0.6	190	209	250	0.3	0.2	55	25.8
200	0.4	0.6	270	342.5	250	0.3	0.2	90	67.8
200	0.3	0.6	70	32.9	250	0.3	0.2	160	172.9
200	0.3	0.6	110	80.9	250	0.3	0.2	230	284.4
200	0.3	0.6	190	216	250	0.5	0.4	55	26
200	0.3	0.6	270	327	250	0.5	0.4	90	68.2
200	0.5	0.8	70	32.8	250	0.5	0.4	160	172.7
200	0.5	0.8	110	80	250	0.5	0.4	230	285.7
200	0.5	0.8	190	201.2	250	0.4	0.4	55	26.1
200	0.5	0.8	270	326.6	250	0.4	0.4	90	68.5
200	0.4	0.8	70	32.5	250	0.4	0.4	160	175.5
200	0.4	0.8	110	80.3	250	0.4	0.4	230	290.7
200	0.4	0.8	190	214	250	0.3	0.4	55	26.1
200	0.4	0.8	270	322.9	250	0.3	0.4	90	68.9
200	0.3	0.8	70	1	250	0.3	0.4	160	179.6
200	0.3	0.8	110	2.6	250	0.3	0.4	230	296.4

λ/g	t/λ	x_n	F	S	λ/g	t/λ	x_n	F	S
250	0.5	0.6	55	26	200	0.4	0.2	860	800.3
250	0.5	0.6	90	68.3	200	0.4	0.2	1204	896.6
250	0.5	0.6	160	174.8	250	0.4	0.2	860	824.7
250	0.5	0.6	230	289	250	0.4	0.2	1204	922.1
250	0.4	0.6	55	26.1	70	0.4	0.4	1204	1182.3
250	0.4	0.6	90	69	100	0.4	0.4	1204	1314.1
250	0.4	0.6	160	179.5	150	0.4	0.4	1204	1385.9
250	0.4	0.6	230	296	200	0.4	0.4	1204	1419.7
250	0.3	0.6	55	25.4	250	0.4	0.4	860	1188.5
250	0.3	0.6	90	68.8	250	0.4	0.4	1204	1441.1
250	0.3	0.6	160	186.4	150	0.4	0.6	1204	1620.2
250	0.3	0.6	230	279.5	200	0.4	0.6	1204	1659.8
250	0.5	0.8	55	25.6	250	0.4	0.6	1204	1683
250	0.5	0.8	90	67.4	250	0.4	0.8	1204	1574.5
250	0.5	0.8	160	174.7	70	0.5	0.2	1204	631.4
250	0.5	0.8	230	286.2	100	0.5	0.2	1204	746.1
250	0.4	0.8	55	25.2	150	0.5	0.2	1204	811.7
250	0.4	0.8	90	67.4	200	0.5	0.2	1204	850.9
250	0.4	0.8	160	185.3	250	0.5	0.2	1204	876.9
250	0.4	0.8	230	277.3	70	0.5	0.4	1204	1141.9
250	0.3	0.8	55	0.6	100	0.5	0.4	1204	1278.7
250	0.3	0.8	90	1.6	150	0.5	0.4	1204	353.5
250	0.3	0.8	160	5.1	200	0.5	0.4	1204	1382.9
250	0.3	0.8	230	10.6	250	0.5	0.4	1204	1404.2
250	0.3	0.8	550	60	150	0.5	0.6	1204	1516.9
250	0.3	0.8	750	109	200	0.5	0.6	1204	553.5
40	0.3	0.2	1204	490.2	250	0.5	0.6	1204	1576
70	0.3	0.2	1204	628.6	100	0.5	0.8	1204	1263.2
100	0.3	0.2	860	711.74	150	0.5	0.8	1204	1336.9
100	0.3	0.2	1204	773.7	200	0.5	0.8	1204	1379.9
150	0.3	0.2	860	783.3	250	0.5	0.8	1204	1405.1
150	0.3	0.2	1204	844.52					
200	0.3	0.2	860	825.26					
200	0.3	0.2	1204	888.7					
250	0.3	0.2	688	741.84					
250	0.3	0.2	1204	910.9					
70	0.3	0.4	1204	1235.8					
100	0.3	0.4	1204	1355.4					
150	0.3	0.4	1204	1430.2					
200	0.3	0.4	1204	1472.3					
250	0.3	0.4	860	1258.2					
250	0.3	0.4	1204	1497.1					
150	0.3	0.6	1204	1591.2					
200	0.3	0.6	1204	1623.5					
250	0.3	0.6	1204	1640					
200	0.3	0.8	1204	329.7					
250	0.3	0.8	1204	328.7					
70	0.4	0.2	1204	659.6					
100	0.4	0.2	860	672.9					
100	0.4	0.2	1204	768.2					
150	0.4	0.2	860	759.9					
150	0.4	0.2	1204	854.7					

2) Data set used for training network NN2 (Sec.3.8.2) for computation of B_t-mmf curves for symmetrically slotted structures (λ/g=0.0172) is presented below. As discussed in Sec.3.8.2, inputs to this network are λ/g, t/λ, x_n and flux density (B_t), and output is mmf (F).

λ/g	t/λ	x_n	B_t	F	λ/g	t/λ	x_n	B_t	F
40	0.5	0	0	0	40	0.4	0.2	1.896	1235
40	0.5	0	0.99	302	40	0.3	0.2	0	0
40	0.5	0	1.48	481	40	0.3	0.2	0.682	302
40	0.5	0	1.72	825	40	0.3	0.2	1.064	481
40	0.5	0	1.87	1235	40	0.3	0.2	1.207	825
40	0.4	0	0	0	40	0.3	0.2	1.395	1235
40	0.4	0	1.03	302	40	0.5	0.4	0	0
40	0.4	0	1.56	481	40	0.5	0.4	0.759	302
40	0.4	0	1.77	825	40	0.5	0.4	1.199	481
40	0.4	0	1.89	1235	40	0.5	0.4	1.608	825
40	0.3	0	0	0	40	0.5	0.4	1.788	1235
40	0.3	0	1.1	302	40	0.4	0.4	0	0
40	0.3	0	1.7	481	40	0.4	0.4	0.739	302
40	0.3	0	1.8	825	40	0.4	0.4	1.158	481
40	0.3	0	1.94	1235	40	0.4	0.4	1.567	825
40	0.5	1	0	0	40	0.4	0.4	1.777	1235
40	0.5	1	0.4	302	40	0.3	0.4	0	0
40	0.5	1	0.65	481	40	0.3	0.4	0.708	302
40	0.5	1	1.11	825	40	0.3	0.4	1.107	481
40	0.5	1	1.49	1235	40	0.3	0.4	1.605	825
40	0.4	1	0	0	40	0.3	0.4	1.854	1235
40	0.4	1	0.284	302	40	0.5	0.6	0	0
40	0.4	1	0.453	481	40	0.5	0.6	0.624	302
40	0.4	1	0.777	825	40	0.5	0.6	0.976	481
40	0.4	1	1.164	1235	40	0.5	0.6	1.379	825
40	0.3	1	0	0	40	0.5	0.6	1.63	1235
40	0.3	1	0.28	302	40	0.4	0.6	0	0
40	0.3	1	0.45	481	40	0.4	0.6	0.555	302
40	0.3	1	0.77	825	40	0.4	0.6	0.868	481
40	0.3	1	1.16	1235	40	0.4	0.6	1.283	825
40	0.5	0.2	0	0	40	0.4	0.6	1.6	1235
40	0.5	0.2	0.889	302	40	0.3	0.6	0	0
40	0.5	0.2	1.388	481	40	0.3	0.6	0.437	302
40	0.5	0.2	1.717	825	40	0.3	0.6	0.696	481
40	0.5	0.2	1.877	1235	40	0.3	0.6	1.13	825
40	0.4	0.2	0	0	40	0.3	0.6	1.626	1235
40	0.4	0.2	0.908	302	40	0.5	0.8	0	0
40	0.4	0.2	1.417	481	40	0.5	0.8	0.49	302
40	0.4	0.2	1.726	825	40	0.5	0.8	0.78	481

λ/g	t/λ	x_n	B_t	F	λ/g	t/λ	x_n	B_t	F
40	0.5	0.8	1.167	825	70	0.4	0.2	1.827	707
40	0.5	0.8	1.497	1235	70	0.3	0.2	0	0
40	0.4	0.8	0	0	70	0.3	0.2	0.828	175
40	0.4	0.8	0.368	302	70	0.3	0.2	1.037	220
40	0.4	0.8	0.594	481	70	0.3	0.2	1.666	470
40	0.4	0.8	0.999	825	70	0.3	0.2	1.855	707
40	0.4	0.8	1.388	1235	70	0.5	0.4	0	0
40	0.3	0.8	0	0	70	0.5	0.4	0.695	175
40	0.3	0.8	0.3	302	70	0.5	0.4	0.873	220
40	0.3	0.8	0.48	481	70	0.5	0.4	1.449	470
40	0.3	0.8	0.82	825	70	0.5	0.4	1.619	707
40	0.3	0.8	1.25	1235	70	0.4	0.4	0	0
70	0.5	0	0	0	70	0.4	0.4	0.649	175
70	0.5	0	0.86	175	70	0.4	0.4	0.819	220
70	0.5	0	1.7	470	70	0.4	0.4	1.378	470
70	0.5	0	1.86	707	70	0.4	0.4	1.578	707
70	0.4	0	0	0	70	0.3	0.4	0	0
70	0.4	0	1	175	70	0.3	0.4	0.579	175
70	0.4	0	1.25	220	70	0.3	0.4	0.728	220
70	0.4	0	1.72	470	70	0.3	0.4	1.267	470
70	0.4	0	1.89	707	70	0.3	0.4	1.545	707
70	0.3	0	0	0	70	0.5	0.6	0	0
70	0.3	0	1.02	175	70	0.5	0.6	0.54	175
70	0.3	0	1.3	220	70	0.5	0.6	0.83	275
70	0.3	0	1.75	470	70	0.5	0.6	1.156	470
70	0.3	0	1.93	707	70	0.5	0.6	1.358	707
70	0.5	1	0	0	70	0.4	0.6	0	0
70	0.5	1	0.286	175	70	0.4	0.6	0.45	175
70	0.5	1	0.35	220	70	0.4	0.6	0.998	470
70	0.5	1	0.75	470	70	0.4	0.6	1.221	707
70	0.5	1	1.04	707	70	0.3	0.6	0	0
70	0.4	1	0	0	70	0.3	0.6	0.3	175
70	0.4	1	0.17	175	70	0.3	0.6	0.47	275
70	0.4	1	0.26	275	70	0.3	0.6	0.785	470
70	0.4	1	0.46	470	70	0.3	0.6	1.103	707
70	0.4	1	0.69	707	70	0.5	0.8	0	0
70	0.4	1	1.21	1235	70	0.5	0.8	0.394	175
70	0.3	1	0	0	70	0.5	0.8	0.6	275
70	0.3	1	0.17	175	70	0.5	0.8	0.876	470
70	0.3	1	0.26	275	70	0.5	0.8	1.104	707
70	0.3	1	0.45	470	70	0.4	0.8	0	0
70	0.3	1	0.68	707	70	0.4	0.8	0.248	175
70	0.3	1	1.19	1235	70	0.4	0.8	0.39	275
70	0.5	0.2	0	0	70	0.4	0.8	0.659	470
70	0.5	0.2	0.839	175	70	0.4	0.8	0.923	707
70	0.5	0.2	1.059	220	70	0.4	0.8	1.11	900
70	0.5	0.2	1.649	470	70	0.4	0.8	1.42	1235
70	0.5	0.2	1.809	707	70	0.3	0.8	0	0
70	0.4	0.2	0	0	70	0.3	0.8	0.18	175
70	0.4	0.2	0.839	175	70	0.3	0.8	0.28	275
70	0.4	0.2	1.049	220	70	0.3	0.8	0.48	470
70	0.4	0.2	1.658	470	70	0.3	0.8	0.73	707

λ/g	t/λ	x_n	B_t	F	λ/g	t/λ	x_n	B_t	F
70	0.3	0.8	1.27	1235	100	0.4	0.4	0.606	125
100	0.5	0	0	0	100	0.4	0.4	0.949	200
100	0.5	0	0.963	125	100	0.4	0.4	1.285	350
100	0.5	0	1.448	200	100	0.4	0.4	1.464	525
100	0.5	0	1.732	350	100	0.3	0.4	0	0
100	0.5	0	1.892	525	100	0.3	0.4	0.515	125
100	0.4	0	0	0	100	0.3	0.4	0.809	200
100	0.4	0	0.982	125	100	0.3	0.4	1.134	350
100	0.4	0	1.464	200	100	0.3	0.4	1.359	525
100	0.4	0	1.749	350	100	0.5	0.6	0	0
100	0.4	0	1.914	525	100	0.5	0.6	0.497	125
100	0.3	0	0	0	100	0.5	0.6	0.779	200
100	0.3	0	1.013	125	100	0.5	0.6	1.064	350
100	0.3	0	1.498	200	100	0.5	0.6	1.225	525
100	0.3	0	1.772	350	100	0.4	0.6	0	0
100	0.3	0	1.946	525	100	0.4	0.6	0.395	125
100	0.5	1	0	0	100	0.4	0.6	0.621	200
100	0.5	1	0.216	125	100	0.4	0.6	0.873	350
100	0.5	1	0.346	200	100	0.4	0.6	1.058	525
100	0.5	1	0.59	350	100	0.3	0.6	0	0
100	0.5	1	0.808	525	100	0.3	0.6	0.227	125
100	0.4	1	0	0	100	0.3	0.6	0.362	200
100	0.4	1	0.124	125	100	0.3	0.6	0.621	350
100	0.4	1	0.199	200	100	0.3	0.6	0.859	525
100	0.4	1	0.349	350	100	0.5	0.8	0	0
100	0.4	1	0.523	525	100	0.5	0.8	0.335	125
100	0.3	1	0	0	100	0.5	0.8	0.526	200
100	0.3	1	0.125	125	100	0.5	0.8	0.75	350
100	0.3	1	0.2	200	100	0.5	0.8	0.925	525
100	0.3	1	0.351	350	100	0.4	0.8	0	0
100	0.3	1	0.526	525	100	0.4	0.8	0.186	125
100	0.5	0.2	0	0	100	0.4	0.8	0.297	200
100	0.5	0.2	0.825	125	100	0.4	0.8	0.51	350
100	0.5	0.2	1.287	200	100	0.4	0.8	0.71	525
100	0.5	0.2	1.644	350	100	0.3	0.8	0	0
100	0.5	0.2	1.804	525	100	0.3	0.8	0.134	125
100	0.4	0.2	0	0	100	0.3	0.8	0.214	200
100	0.4	0.2	0.815	125	100	0.3	0.8	0.375	350
100	0.4	0.2	1.271	200	100	0.3	0.8	0.563	525
100	0.4	0.2	1.638	350	100	0.3	0.8	1.072	1200
100	0.4	0.2	1.8	525	70	0.3	0	2.06	1204
100	0.3	0.2	0	0	100	0.3	0	2.2	1204
100	0.3	0.2	0.783	125	70	0.4	0	2.06	1204
100	0.3	0.2	1.225	200	100	0.4	0	2.16	1204
100	0.3	0.2	1.617	350	70	0.5	0	2.01	1204
100	0.3	0.2	1.797	525	100	0.5	0	2.12	1204
100	0.5	0.4	0	0	70	0.3	0.2	1.98	1204
100	0.5	0.4	0.663	125	100	0.3	0.2	2.12	1204
100	0.5	0.4	1.038	200	70	0.4	0.2	2.02	1204
100	0.5	0.4	1.38	350	100	0.4	0.2	2.07	1204
100	0.5	0.4	1.539	525	70	0.5	0.2	1.9	1204
100	0.4	0.4	0	0	100	0.5	0.2	2	1204

λ/g	t/λ	x_n	B_t	F	λ/g	t/λ	x_n	B_t	F
70	0.3	0.4	1.9	1204	70	0.3	0.4	1.4	570
100	0.3	0.4	1.96	1204	70	0.3	0.4	1.7	880
70	0.4	0.4	1.76	1204	100	0.3	0.4	0.4	97
100	0.4	0.4	1.8	1204	100	0.3	0.4	0.7	173
70	0.5	0.4	1.86	1204	100	0.3	0.4	1.5	640
100	0.5	0.4	1.9	1204	100	0.3	0.4	1.6	740
70	0.3	0.6	1.78	1204	100	0.3	0.4	1.7	860
100	0.3	0.6	1.85	1204	100	0.3	0.4	1.8	980
70	0.4	0.6	1.64	1204	40	0.3	0.6	0.6	420
100	0.4	0.6	1.66	1204	40	0.3	0.6	0.9	640
70	0.5	0.6	1.66	1204	40	0.3	0.6	1.3	944
100	0.5	0.6	1.7	1204	40	0.3	0.6	1.5	1100
100	0.4	0.8	1.57	1204	70	0.3	0.6	0.6	370
70	0.5	0.8	1.54	1204	70	0.3	0.6	1	640
100	0.5	0.8	1.56	1204	70	0.3	0.6	1.2	780
70	0.3	1	1.34	1204	70	0.3	0.6	1.4	920
100	0.3	1	1.37	1204	70	0.3	0.6	1.6	1070
70	0.4	1	1.36	1204	100	0.3	0.6	0.5	280
100	0.4	1	1.38	1204	100	0.3	0.6	1	607
70	0.5	1	1.51	1204	100	0.3	0.6	1.2	734
100	0.5	1	1.53	1204	100	0.3	0.6	1.4	880
40	0.3	0.2	1.2	600	100	0.3	0.6	1.6	1010
40	0.3	0.2	0.8	350	40	0.3	0.8	0.2	200
40	0.3	0.2	0.9	396	40	0.3	0.8	0.6	600
70	0.3	0.2	0.6	124	40	0.3	0.8	1	1000
70	0.3	0.2	0.7	150	70	0.3	0.8	1.2	200
70	0.3	0.2	0.8	168	70	0.3	0.8	1.4	686
70	0.3	0.2	1.1	232	70	0.3	0.8	1.6	986
70	0.3	0.2	1.2	262	100	0.3	0.8	1	1100
70	0.3	0.2	1.3	290	100	0.3	0.8	0.8	843
70	0.3	0.2	1.4	324	40	0.4	0.2	0.5	164
70	0.3	0.2	1.5	370	40	0.4	0.2	1.2	386
70	0.3	0.2	1.6	424	40	0.4	0.2	1.6	660
70	0.3	0.2	1.8	610	40	0.4	0.2	1.8	940
70	0.3	0.2	1.9	820	70	0.4	0.2	0.5	106
100	0.3	0.2	0.5	80	70	0.4	0.2	1.3	308
100	0.3	0.2	0.9	145	100	0.4	0.2	1.4	235
100	0.3	0.2	1	160	100	0.4	0.2	1.7	400
100	0.3	0.2	1.1	178	40	0.4	0.4	1.3	470
100	0.3	0.2	1.3	222	70	0.4	0.4	1.5	520
100	0.3	0.2	1.4	256	70	0.4	0.4	1.7	1040
100	0.3	0.2	1.7	420	100	0.4	0.4	1.1	250
100	0.3	0.2	1.9	650	100	0.4	0.4	1.6	720
100	0.3	0.2	2	860	40	0.4	0.6	1.4	980
40	0.3	0.4	0.5	208	70	0.4	0.6	1.4	900
40	0.3	0.4	0.8	334	70	0.4	0.6	1.5	1000
40	0.3	0.4	0.9	380	100	0.4	0.6	1.4	880
40	0.3	0.4	1.3	590	100	0.4	0.6	1.5	1020
40	0.3	0.4	1.4	654	40	0.4	0.8	0.8	660
40	0.3	0.4	1.5	740	40	0.4	0.8	1.2	1020
40	0.3	0.4	1.7	970	70	0.4	0.8	1	800
70	0.3	0.4	1	318	70	0.4	0.8	1.2	980

λ/g	t/λ	x_n	B_t	F	λ/g	t/λ	x_n	B_t	F
100	0.4	0.8	1.2	944	150	0.3	0.2	0.81	90
100	0.4	0.8	1.4	1080	150	0.3	0.2	1.277	150
40	0.5	0.2	0.5	166	150	0.3	0.2	1.582	250
40	0.5	0.2	1.6	660	150	0.3	0.2	1.758	380
70	0.5	0.2	0.5	106	150	0.5	0.4	0.682	90
70	0.5	0.2	1.2	254	150	0.5	0.4	1.068	150
70	0.5	0.2	1.5	354	150	0.5	0.4	1.321	250
100	0.5	0.2	1.5	264	150	0.5	0.4	1.463	380
40	0.5	0.4	0.5	200	150	0.4	0.4	0.612	90
40	0.5	0.4	1	400	150	0.4	0.4	0.966	150
40	0.5	0.4	1.4	620	150	0.4	0.4	1.215	250
70	0.5	0.4	1	256	150	0.4	0.4	1.37	380
100	0.5	0.4	1.7	780	150	0.3	0.4	0.497	90
100	0.5	0.6	0.6	150	150	0.3	0.4	0.789	150
100	0.5	0.6	1.4	710	150	0.3	0.4	1.023	250
40	0.5	0.8	1.3	980	150	0.3	0.4	1.213	380
70	0.5	0.8	1	604	150	0.5	0.6	0.498	90
70	0.5	0.8	1.3	930	150	0.5	0.6	0.787	150
100	0.5	0.8	1	595	150	0.5	0.6	0.995	250
100	0.5	0.8	1.3	890	150	0.5	0.6	1.132	380
150	0.5	0	1.02	90	150	0.4	0.6	0.38	90
150	0.5	0	1.508	150	150	0.4	0.6	0.605	150
150	0.5	0	1.756	250	150	0.4	0.6	0.789	250
150	0.5	0	1.912	379	150	0.4	0.6	0.943	380
150	0.4	0	1.037	90	150	0.3	0.6	0.185	90
150	0.4	0	1.526	150	150	0.3	0.6	0.308	150
150	0.4	0	1.776	250	150	0.3	0.6	0.501	250
150	0.4	0	1.94	380	150	0.3	0.6	0.702	380
150	0.3	0	1.063	90	150	0.5	0.8	0.316	90
150	0.3	0	1.55	150	150	0.5	0.8	0.503	150
150	0.3	0	1.805	250	150	0.5	0.8	0.663	250
150	0.3	0	1.982	380	150	0.5	0.8	0.804	380
150	0.5	1	0.174	90	150	0.4	0.8	0.15	90
150	0.5	1	0.29	150	150	0.4	0.8	0.25	150
150	0.5	1	0.469	250	150	0.4	0.8	0.408	250
150	0.5	1	0.643	379	150	0.4	0.8	0.576	380
150	0.4	1	0.097	90	150	0.3	0.8	0.108	90
150	0.4	1	0.161	150	150	0.3	0.8	0.179	150
150	0.4	1	0.268	250	150	0.3	0.8	0.299	250
150	0.4	1	0.408	380	150	0.3	0.8	0.455	380
150	0.3	1	0.102	90	150	0.3	0.8	1.077	900
150	0.3	1	0.169	150	200	0.5	0	1.048	70
150	0.3	1	0.282	250	200	0.5	0	1.491	110
150	0.3	1	0.429	380	200	0.5	0	1.762	190
150	0.5	0.2	0.869	90	200	0.5	0	1.9	270
150	0.5	0.2	1.35	150	200	0.4	0	1.062	70
150	0.5	0.2	1.629	250	200	0.4	0	1.512	110
150	0.5	0.2	1.779	380	200	0.4	0	1.78	190
150	0.4	0.2	0.847	90	200	0.4	0	1.924	270
150	0.4	0.2	1.324	150	200	0.3	0	1.085	70
150	0.4	0.2	1.613	250	200	0.3	0	1.52	110
150	0.4	0.2	1.771	380	200	0.3	0	1.78	190

λ/g	t/λ	x_n	B_t	F	λ/g	t/λ	x_n	B_t	F
200	0.3	0	1.924	270	200	0.5	0.8	0.718	270
200	0.5	1	0.149	70	200	0.4	0.8	0.127	70
200	0.5	1	0.234	110	200	0.4	0.8	0.199	110
200	0.5	1	0.392	190	200	0.4	0.8	0.336	190
200	0.5	1	0.517	270	200	0.4	0.8	0.452	270
200	0.4	1	0.084	70	200	0.3	0.8	0.096	70
200	0.4	1	0.132	110	200	0.3	0.8	0.151	110
200	0.4	1	0.227	190	200	0.3	0.8	0.261	190
200	0.4	1	0.323	270	200	0.3	0.8	0.371	270
200	0.3	1	0.091	70	200	0.3	0.8	0.756	550
200	0.3	1	0.143	110	200	0.3	0.8	0.962	700
200	0.3	1	0.247	190	250	0.5	0	1.023	55
200	0.3	1	0.351	270	250	0.5	0	1.507	90
200	0.5	0.2	0.876	70	250	0.5	0	1.781	160
200	0.5	0.2	1.292	110	250	0.5	0	1.917	230
200	0.5	0.2	1.587	190	250	0.4	0	1.036	55
200	0.5	0.2	1.709	270	250	0.4	0	1.519	90
200	0.4	0.2	0.848	70	250	0.4	0	1.8	160
200	0.4	0.2	1.258	110	250	0.4	0	1.943	230
200	0.4	0.2	1.563	190	250	0.3	0	1.056	55
200	0.4	0.2	1.693	270	250	0.3	0	1.54	90
200	0.3	0.2	0.801	70	250	0.3	0	1.829	160
200	0.3	0.2	1.197	110	250	0.3	0	1.984	230
200	0.3	0.2	1.515	190	250	0.5	1	0.128	55
200	0.3	0.2	1.659	270	250	0.5	1	0.209	90
200	0.5	0.4	0.687	70	250	0.5	1	0.357	160
200	0.5	0.4	1.02	110	250	0.5	1	0.475	230
200	0.5	0.4	1.276	190	250	0.4	1	0.074	55
200	0.5	0.4	1.39	270	250	0.4	1	0.121	90
200	0.4	0.4	0.61	70	250	0.4	1	0.215	160
200	0.4	0.4	0.912	110	250	0.4	1	0.309	230
200	0.4	0.4	1.162	190	250	0.3	1	0.083	55
200	0.4	0.4	1.281	270	250	0.3	1	0.135	90
200	0.3	0.4	0.482	70	250	0.3	1	0.24	160
200	0.3	0.4	0.728	110	250	0.3	1	0.345	230
200	0.3	0.4	0.962	190	250	0.5	0.2	0.852	55
200	0.3	0.4	1.101	270	250	0.5	0.2	1.289	90
200	0.5	0.6	0.496	70	250	0.5	0.2	1.582	160
200	0.5	0.6	0.742	110	250	0.5	0.2	1.703	230
200	0.5	0.6	0.949	190	250	0.4	0.2	0.821	55
200	0.5	0.6	1.052	270	250	0.4	0.2	1.252	90
200	0.4	0.6	0.371	70	250	0.4	0.2	1.552	160
200	0.4	0.6	0.559	110	250	0.4	0.2	1.682	230
200	0.4	0.6	0.744	190	250	0.3	0.2	0.771	55
200	0.4	0.6	0.857	270	250	0.3	0.2	1.185	90
200	0.3	0.6	0.162	70	250	0.3	0.2	1.495	160
200	0.3	0.6	0.255	110	250	0.3	0.2	1.638	230
200	0.3	0.6	0.429	190	250	0.5	0.4	0.663	55
200	0.3	0.6	0.575	270	250	0.5	0.4	1.013	90
200	0.5	0.8	0.305	70	250	0.5	0.4	1.261	160
200	0.5	0.8	0.46	110	250	0.5	0.4	1.372	230
200	0.5	0.8	0.617	190	250	0.4	0.4	0.585	55

λ/g	t/λ	x_n	B_t	F	λ/g	t/λ	x_n	B_t	F
250	0.4	0.4	0.901	90	250	0.3	0.2	2.32	1204
250	0.4	0.4	1.144	160	150	0.4	0.2	2.1	1204
250	0.4	0.4	1.26	230	200	0.4	0.2	2.05	860
250	0.3	0.4	0.456	55	200	0.4	0.2	2.15	1204
250	0.3	0.4	0.71	90	250	0.4	0.2	2.04	688
250	0.3	0.4	0.939	160	250	0.4	0.2	2.19	1204
250	0.3	0.4	1.073	230	150	0.5	0.2	2.08	1204
250	0.5	0.6	0.475	55	200	0.5	0.2	2.06	860
250	0.5	0.6	0.732	90	200	0.5	0.2	2.1	1204
250	0.5	0.6	0.933	160	250	0.5	0.2	2.04	688
250	0.5	0.6	1.034	230	250	0.5	0.2	2.16	1204
250	0.4	0.6	0.349	55	150	0.3	0.4	2.02	1204
250	0.4	0.6	0.545	90	200	0.3	0.4	1.82	860
250	0.4	0.6	0.724	160	200	0.3	0.4	2.06	1204
250	0.4	0.6	0.833	230	250	0.3	0.4	1.7	688
250	0.3	0.6	0.141	55	250	0.3	0.4	2.08	1204
250	0.3	0.6	0.23	90	150	0.4	0.4	1.81	1204
250	0.3	0.6	0.398	160	200	0.4	0.4	1.76	860
250	0.3	0.6	0.54	230	200	0.4	0.4	1.83	1204
250	0.5	0.8	0.288	55	250	0.4	0.4	1.72	688
250	0.5	0.8	0.45	90	250	0.4	0.4	1.85	1204
250	0.5	0.8	0.604	160	150	0.5	0.4	1.93	1204
250	0.5	0.8	0.702	230	200	0.5	0.4	1.8	860
250	0.4	0.8	0.113	55	200	0.5	0.4	1.96	1204
250	0.4	0.8	0.185	90	250	0.5	0.4	1.72	688
250	0.4	0.8	0.32	160	250	0.5	0.4	1.98	1204
250	0.4	0.8	0.436	230	150	0.3	0.6	1.89	1204
250	0.3	0.8	0.087	55	200	0.3	0.6	1.5	860
250	0.3	0.8	0.142	90	200	0.3	0.6	1.93	1204
250	0.3	0.8	0.252	160	250	0.3	0.6	1.35	688
250	0.3	0.8	0.362	230	250	0.3	0.6	1.95	1204
250	0.3	0.8	0.866	550	150	0.4	0.6	1.69	1204
250	0.3	0.8	1.181	750	200	0.4	0.6	1.46	860
150	0.3	0	2.28	1204	200	0.4	0.6	1.74	1204
200	0.3	0	2.28	860	250	0.4	0.6	1.38	688
200	0.3	0	2.36	1204	250	0.4	0.6	1.76	1204
250	0.3	0	2.3	688	150	0.5	0.6	1.73	1204
250	0.3	0	2.4	1204	200	0.5	0.6	1.55	860
150	0.4	0	2.26	1204	200	0.5	0.6	1.75	1204
200	0.4	0	2.22	860	250	0.5	0.6	1.5	688
200	0.4	0	2.32	1204	250	0.5	0.6	1.76	1204
250	0.4	0	2.2	688	150	0.3	0.8	1.52	1204
250	0.4	0	2.35	1204	200	0.3	0.8	1.16	860
150	0.5	0	2.23	1204	200	0.3	0.8	1.54	1204
200	0.5	0	2.2	860	250	0.3	0.8	1.55	1204
200	0.5	0	2.26	1204	150	0.4	0.8	1.58	1204
250	0.5	0	2.2	688	200	0.4	0.8	1.19	860
250	0.5	0	2.31	1204	200	0.4	0.8	1.59	1204
150	0.3	0.2	2.25	1204	250	0.4	0.8	1.11	688
200	0.3	0.2	2.19	860	250	0.4	0.8	1.6	1204
200	0.3	0.2	2.3	1204	150	0.5	0.8	1.57	1204
250	0.3	0.2	2.17	688	200	0.5	0.8	1.36	860

λ/g	t/λ	x_n	B_t	F
200	0.5	0.8	1.58	1204
250	0.5	0.8	1.25	688
250	0.5	0.8	1.59	1204
150	0.3	1	1.4	1204
200	0.3	1	1.14	860
200	0.3	1	1.42	1204
250	0.3	1	1.03	688
250	0.3	1	1.47	1204
150	0.4	1	1.4	1204
200	0.4	1	1.15	860
200	0.4	1	1.44	1204
250	0.4	1	0.98	688
250	0.4	1	1.5	1204
150	0.5	1	1.54	1204
200	0.5	1	1.32	860
200	0.5	1	1.55	1204
250	0.5	1	1.13	688
250	0.5	1	1.56	1204
200	0.5	0	0	0
200	0.5	1	0	0
200	0.5	0.2	0	0
200	0.5	0.4	0	0
200	0.5	0.6	0	0
200	0.5	0.8	0	0
150	0.5	0	0	0
150	0.5	1	0	0
150	0.5	0.2	0	0
150	0.5	0.4	0	0
150	0.5	0.6	0	0
150	0.5	0.8	0	0
250	0.5	0	0	0
250	0.5	1	0	0
250	0.5	0.2	0	0
250	0.5	0.4	0	0
250	0.5	0.6	0	0
250	0.5	0.8	0	0

A.3. Algorithm for Force Data Set Production of Asymmetrically Slotted Structures:

(The following algorithm is explained in Sec 3.9.3. In order to run the program succesfully, related weight files have to be contained in the same directory.)

UNEQUAL.C

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
main()
{
FILE *fout; FILE *ff;  FILE *fyf;FILE *fys;
double *u,tsl,trl,ldg,xn,F1,F2,F,lr,la,lb,ts,tr,x,xa,xb,g;
double S1,S2,P1,P2,Bts,Btr,S,P,a,b,d,CF,p1,p2;     double OUTPUT();
int lc,i,count,tsc,trc,Btc,xnc,in,on,N1,N2,k,p,j;
double wf[4][11][11],btf[11][11],wyf[4][11][11],byf[11][11],ws[4][11][11],bts[11][11] ;
if((u=(double*) malloc(5*sizeof(double)))==NULL ) exit(1);
if((fout=fopen("unqforce","a"))==NULL) { /* Output data is written to file unqforce*/
        printf("can not open the file\n");    exit(0);  }
if((ff=fopen("mmf_force.dat","r"))==NULL) { /*weight file  produced from NN1*/
        printf("can not open the file\n");    exit(0);  }
if((fyf=fopen("Bt_mmf.dat","r"))==NULL) {  /* weight file of NN2 for λ/g in the range [40-100]
*/
        printf("can not open the file\n");    exit(0);  }
if((fys=fopen("ysl1.dat","r"))==NULL) {  /* weight file of NN2 for λ/g in the range [100-250] */
        printf("can not open the file\n");    exit(0);  }

/*****************************force NN*******************************/
fscanf(ff,"%d %d %d %d\n",&in ,&N1,&N2, &on);
for(i=1;i<in+1;i++){for(k=1;k<N1+1;k++){fscanf(ff,"%lg \n",&wf[1][i][k]);}}
for(k=1;k<N1+1;k++){for(p=1;p<N2+1;p++){fscanf(ff,"%lg \n",&wf[2][k][p]);}}
for(p=1;p<N2+1;p++){for(j=1;j<on+1;j++){fscanf(ff,"%lg \n",&wf[3][p][j]);}}
for(k=1;k<N1+1;k++){fscanf(ff," %lg \n",&btf[1][k]);}
for(p=1;p<N2+1;p++){fscanf(ff,"%lg \n",&btf[2][p]);}
for(j=1;j<on+1;j++){fscanf(ff,"%lg \n",&btf[3][j]);}

/*********************** BT-MMF 1*******************************/
fscanf(fyf,"%d %d %d %d\n",&in ,&N1,&N2, &on);
for(i=1;i<in+1;i++){for(k=1;k<N1+1;k++){fscanf(fyf,"%lg \n",&wyf[1][i][k]);}}
for(k=1;k<N1+1;k++){for(p=1;p<N2+1;p++){fscanf(fyf,"%lg \n",&wyf[2][k][p]);}}
for(p=1;p<N2+1;p++){for(j=1;j<on+1;j++){fscanf(fyf,"%lg \n",&wyf[3][p][j]);}}
for(k=1;k<N1+1;k++){fscanf(fyf," %lg \n",&byf[1][k]);}
for(p=1;p<N2+1;p++){fscanf(fyf,"%lg \n",&byf[2][p]);}
for(j=1;j<on+1;j++){fscanf(fyf,"%lg \n",&byf[3][j]);}
/***********************  BT-MMF 2          ***********************/
fscanf(fys,"%d %d %d %d\n",&in ,&N1,&N2, &on);
for(i=1;i<in+1;i++){for(k=1;k<N1+1;k++){fscanf(fys,"%lg \n",&ws[1][i][k]);}}
for(k=1;k<N1+1;k++){for(p=1;p<N2+1;p++){fscanf(fys,"%lg \n",&ws[2][k][p]);}}
```

```
for(p=1;p<N2+1;p++){for(j=1;j<on+1;j++){fscanf(fys,"%lg \n",&ws[3][p][j]);}}
for(k=1;k<N1+1;k++){fscanf(fys," %lg \n",&bts[1][k]);}
for(p=1;p<N2+1;p++){fscanf(fys,"%lg \n",&bts[2][p]);}
for(j=1;j<on+1;j++){fscanf(fys,"%lg \n",&bts[3][j]);}

/*****************************************************************/
 for(lc=0;lc<6;lc++)  { if(lc==0) ldg=40; if(lc==1) ldg=70;  if(lc==2) ldg=100;
 if(lc==3) ldg=150;  if(lc==4) ldg=200;  if(lc==5) ldg=250;
for(tsc=0;tsc<3;tsc++)   { if(tsc==0) tsl=0.3;   if(tsc==1) tsl=0.4; if(tsc==2) tsl=0.5;
for(trc=0;trc<3;trc++)  { if(trc==0) trl=0.3;   if(trc==1) trl=0.4; if(trc==2) trl=0.5;
for(xnc=0;xnc<4;xnc++)   {if(xnc==0) xn=0.2; if(xnc==1) xn=0.4; if(xnc==2) xn=0.6;
if(xnc==3) xn=0.8;  for(Btc=0;Btc<10;Btc++){
if(Btc==0) Bts=0.2;if(Btc==1) Bts=0.4; if(Btc==2) Bts=0.6;if(Btc==3) Bts=0.8;
if(Btc==4) Bts=1  ;if(Btc==5) Bts=1.2; if(Btc==6) Bts=1.4;if(Btc==7) Bts=1.6;
if(Btc==8) Bts=1.8;if(Btc==9) Bts=2;   /*if(tsl==trl) goto  SON;*/
lr=0.0172;
ts=tsl*lr;
x=(xn*lr)/2 ;
tr=trl*lr;
g=lr/ldg;
d=lr-((ts/2)+(tr/2)+x);/*asymmetrical teeth*/
d=d/g;
a=ts+x+(25*g);
if (a>=lr) la=a;
if(a<lr) la=lr;
b=tr+x+(25*g);
if (b>=lr) lb=b;
if(b<lr) lb=lr;
for(i=0;i<5;i++) u[i]=0;


/* FOR GEOMETRY A */
 u[1]=(la/g)/250; u[3]=(x*2)/la;
u[2]=ts/la; u[4]=Bts/2;
count=1;
if(u[1]<0.4) F1=OUTPUT(u,wyf,byf,count);
if(u[1]>=0.4) F1=OUTPUT(u,ws,bts,count);
u[4]=F1/1500;
count=2;
 S1=OUTPUT(u,wf,btf,count);

/*FOR GEOMETRY B*/
Btr=(Bts*ts)/tr;
u[2]=tr/lb; u[1]=(lb/g)/250; u[4]=Btr/2;
u[3]=(x*2)/lb;
count=1;
if(u[1]<0.4) F2=OUTPUT(u,wyf,byf,count);
if(u[1]>=0.4) F2=OUTPUT(u,ws,bts,count);
u[4]=F2/1500;
count=2;
S2=OUTPUT(u,wf,btf,count);
/*torque correction part, optional*/
/*  if(d<25) {  if(Bts>=1.1) CF=(-0.0021*d*d*d)+(0.1414*d*d)-(3.6153*d)+34.6808;
   if(Bts<=0.5) CF=(0.0157*d*d)-(0.8381*d)+11.1889;
```

```
  if((Bts<1.1)&&(Bts>0.5))   {     p1=(0.0157*d*d)-(0.8381*d)+11.1889;
                                   p2=(-0.0021*d*d*d)+(0.1414*d*d)-(3.6153*d)+34.6808;
     CF=p1+((p2-p1)/(0.6/(Bts-0.5)));  }   CF=(100-CF)/100;  } else CF=1;
/*RESULT PART*/
S=((S1+S2)*CF)/2;            F=(F1+F2)/2;

/*********************************************************/
fprintf(fout,"%10lg %10lg %10lg %10lg %10lg %10lg \n",(ldg/250) ,tsl,trl ,xn,F/1500,S/1500);
/*fprintf(fout,"GEOMETRY A: La= %lg La/g= %lg ts/La=%lg Xna= %lg \n",la,la/g,ts/la,(x*2)/la);
fprintf(fout,"Bts=%lg F1=%lg S1=%lg \n",Bts,F1,S1);
fprintf(fout,"GEOMETRY B: Lb= %lg Lb/g= %lg tr/Lb=%lg Xnb= %lg \n",lb,lb/g,tr/lb,(x*2)/lb);
fprintf(fout,"Btr=%lg F2=%lg S2=%lg \n",Btr,F2,S2);
fprintf(fout," CF=%lg   F=%lg   S=%lg   \n",CF,F,S);*/
SON:   k=5;  }}}}} fclose(fout);  }

double OUTPUT(u,w,bt,count)  /* function computes neural network outputs*/
#define boy 11
double *u,w[4][boy][boy],bt[boy][boy];   int count; {   double big,t;
int a,b,c,i,j,jj,nn,l,kk,ii,N1,N2,iv,in,on,n,k,p;/**x1,*x2,*z1,*z2,*yd;*/
double ut,yt,sum,ex1,out,cons,ex,x1[11],x2[11],z1[11],z2[11],yd[2];
int dimu,dimy,dimx1,dimx2,dd,dim; if(count==1) N1=N2=6; if(count==2) N1=N2=10;   iv=1;
in=4; on=1;  /*dimu=iv*in; dimy=iv*on; dimx1=iv*N1; dimx2=iv*N2;
if((yd=(double*) malloc(dimy*sizeof(double)))==NULL ) exit(1);
if((x1=(double*) malloc(dimx1*sizeof(double)))==NULL ) exit(1);
if((z1=(double*) malloc(dimx1*sizeof(double)))==NULL ) exit(1);
if((x2=(double*) malloc(dimx2*sizeof(double)))==NULL ) exit(1);
if((z2=(double*) malloc(dimx2*sizeof(double)))==NULL ) exit(1);*/

/***          OUTPUT VECTOR CALCULATION          *****/
cons=10;  for(ii=1;ii<iv+1;ii++)

   {  /**    CALCULATION OF THE STATE VECTORS X1 AND Z1   ***/
      for(k=1;k<N1+1;k++) {  sum=0;          for(n=1;n<in+1;n++) {
      a=in*(ii-1)+n;  sum=w[1][n][k]*u[a]+sum;          } c=N1*(ii-1)+k;
x1[c]=sum+bt[1][k];
            z1[c]=tanh(cons*x1[c]);   }

 /***   CALCULATION OF THE STATE VECTORS X2 AND Z2   ***/
      for(p=1;p<N2+1;p++) {  sum=0;          for(k=1;k<N1+1;k++) {
      c=N1*(ii-1)+k;   sum=w[2][k][p]*z1[c]+sum;  } b=N2*(ii-1)+p;
      x2[b]=sum+bt[2][p];  z2[b]=tanh(cons*x2[b]);  }

/***     CALCULATION OF THE OUTPUT VECTORS          ***/
      for(j=1;j<on+1;j++) {sum=0; for(p=1;p<N2+1;p++) {
      b=N2*(ii-1)+p;   sum=w[3][p][j]*z2[b]+sum;  }  dd=on*(ii-1)+j;
      yd[dd]=sum+bt[3][j];  }}
for(ii=1;ii<iv+1;ii++) {  for(i=1;i<in+1;i++)  {  a=in*(ii-1)+i;  printf(" in= %lg " ,u[a]); }
for(j=1;j<on+1;j++)  { dd=on*(ii-1)+j;  /*if (count==1) yd[dd]=(yd[dd]*1500);/*result  is mmf*/
/*if ((count==2)||(count==4)) yd[dd]=(yd[dd]*1500);*/
yd[dd]=yd[dd]*1500;  printf(" out= %lg  ",yd[dd] ); }
printf("\n"); }
out=yd[1];
return(out);
}
```

A.3. Force Calculation Algorithm:

Below algorithm is used to obtain static torque curves of any structure in udss form. For succesfull running of the program related weight files of asymmetrically slotted structures have to be in the same directory with the program.

FORCEUDSS.C

```
cls();
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define boy 21
main()
{
FILE *ff;  double OUTPUT();
double u[6],w[4][boy][boy],bt[boy][boy],S,ba;  int k,in,N1,N2,on,j,p,i;
//if((u=(double*) malloc(6*sizeof(double)))==NULL ) exit(1);
in=5; on=1; N1=20; N2=10;    for(i=0;i<6;i++) u[i]=0;

/*************************************************************************/
if((ff=fopen("weight1.dat","r"))==NULL) { printf("can not open the file\n");    exit(0);  }
/* weight1.dat contains weights of network that is used for asymmetrically slotted structures  force
calculation */

/*****************************force NN*************************************/
fscanf(ff,"%d %d %d %d\n",&in ,&N1,&N2, &on);
for(i=1;i<in+1;i++){for(k=1;k<N1+1;k++){fscanf(ff,"%lg \n",&w[1][i][k]);}}
for(k=1;k<N1+1;k++){for(p=1;p<N2+1;p++){fscanf(ff,"%lg \n",&w[2][k][p]);}}
for(p=1;p<N2+1;p++){for(j=1;j<on+1;j++){fscanf(ff,"%lg \n",&w[3][p][j]);}}
for(k=1;k<N1+1;k++){fscanf(ff," %lg \n",&bt[1][k]);}
for(p=1;p<N2+1;p++){fscanf(ff,"%lg \n",&bt[2][p]);}
for(j=1;j<on+1;j++){fscanf(ff,"%lg \n",&bt[3][j]);}
/*************************************************************************/

u[1]=(200.0/250);  /* λr/g value and normalized with 250 for NN conversion
u[3]=0.4;          /*  tg/λr  */
u[2]=0.5;          /* tr/λr  */
u[4]=0.1;          /*  normalized position starting from 0.1 */
ba=((30000*0.0172)/1500); /* normalized mmf, written in udss form (300000) */
u[5]=ba;                  /* normalized mmf*/
for(i=1;i<10;i++)/* for 10 different normalized positions (0.1,0.2,...,1) neural network function is
called*/
{
S=OUTPUT(u,w,bt);
u[4]=u[4]+0.1;  /*increase normalized position*/
}}

double OUTPUT(u,w,bt)  /* Neural Network function */
```

```
#define boy 21
double u[6],w[4][boy][boy],bt[boy][boy];
{  double big,t;
int a,b,c,i,j,jj,nn,l,kk,ii,N1,N2,iv,in,on,n,k,p;/**x1,*x2,*z1,*z2,*yd;*/
double ut,yt,sum,ex1,out,cons,ex,x1[21],x2[11],z1[21],z2[11],yd[2];
int dimu,dimy,dimx1,dimx2,dd,dim;
N1=20;N2=10;   iv=1; in=5; on=1;
cons=10;          for(ii=1;ii<iv+1;ii++)
    {
        for(k=1;k<N1+1;k++)
        {sum=0; for(n=1;n<in+1;n++) {a=in*(ii-1)+n; sum=w[1][n][k]*u[a]+sum; }
                c=N1*(ii-1)+k;   x1[c]=sum+bt[1][k]; z1[c]=tanh(cons*x1[c]); }
        for(p=1;p<N2+1;p++)
        { sum=0; for(k=1;k<N1+1;k++) {
                        c=N1*(ii-1)+k; sum=w[2][k][p]*z1[c]+sum;}
                b=N2*(ii-1)+p; x2[b]=sum+bt[2][p]; z2[b]=tanh(cons*x2[b]);          }
                for(j=1;j<on+1;j++)
                { sum=0; for(p=1;p<N2+1;p++)
                { b=N2*(ii-1)+p; sum=w[3][p][j]*z2[b]+sum; }
                dd=on*(ii-1)+j;  yd[dd]=sum+bt[3][j];
        }}
for(ii=1;ii<iv+1;ii++)
{
for(i=1;i<in+1;i++)
{ a=in*(ii-1)+i; printf(" in= %lg " ,u[a]); }
for(j=1;j<on+1;j++)
{dd=on*(ii-1)+j;
yd[dd]=(2*(yd[dd]*1500))/0.0172; /* calculate force for udss , λr =1m, Lc=1m */
printf(" out= %lg  ",yd[dd] ); }
printf("\n");
}
out=yd[1];
return(out); /* return force */
/* In this algorithm, if output is multiplied with motor radious, torque may be found*/

}
```

APPENDIX B

OPTIMIZATION ALGORITHM AND GRAPHS PRESENTING THE EFFECTS OF λ/g AND t_s/λ ON TORQUE RIPPLE

B.1. Augmented Lagrangian Algorithm:

In order to run the optimization program, which is explained in Chapter 4, succesfully, weight file obtained from training the asymmetrically slotted structures (Sec.3.9.4), have to be contained in the same directory.

RIPPLEOPT.C

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

main()
{
double FXNS(),FXFE(),FXFI(),AUGLAG(),GFF(),GFE(),GFI(),GALAG();
double DFPRV(),DMAX(),OUTPUT();
double AL,F,FS,EPS1,EPS2,EPS3,*A,*B,*C,*D,W1,W2,W3,WF,WMAX,*GO,*XP,NG;
int N,NE,NI,NBV,i,*XB,ICONV,NBD,NN,IUP,IC1,K,IVAR,itot,NGE;
double *FE,*FI,*FES,*FIS,*GAL,*HE,*HI,*HVL,*HVU,*XT,*X,*R,*GE,*GI,*GF;
double *H,*DLG,*DLX,*dfp,GMAG,EPD,EPT,EPV,RHO,SLOPE,RMAG;
double D1,D2,D3,Y1,Y2,Y3,YS,Y,DT,DELX,DELG,TEMP,DS;
int  IBSD,IT,IFAIL,KSRC,NSRC;

/***WRITE  THE  VALUES OF N= NUMBER OF VARIABLES, NE=# OF  EQUALITY***
****CONSTRAINTS, NI= # OF INEQUALITY (<=) CONSTRAINTS, NBV= #OF
BOUNDED****
**********************      VARIABLES   ***************************/
N=3; NE=0; NI=1; NBV=3;
/*********************************************************************/
if((XB=(int*) malloc(N*sizeof(int)))==NULL) exit(1);
/***********ENTER THE VALUES  FOR  BOUNDED VARIABLES*****************/
/*********          FROM I=0 TO N-1              **********/
/*  XB(I)=0 IF  X(I) IS  NOT BOUNDED  1 IF  IS  BOUNDED  FROM  BELOW
ONLY , 2  IF IS BOUNDED FROM ABOVE ONLY, 3 IF  BOUNDED  FROM  BOTH***/
```

```
XB[0]=3; XB[1]=3; XB[2]=3;
/*************************************************************************/
if((A=(double*) malloc(NE*sizeof(double)))==NULL) exit(1);
if((B=(double*) malloc(NI*sizeof(double)))==NULL) exit(1);
/**********ENTER THE  RIGHT HAND SIDE OF EQUALITY A(0,,NE) AND********/
/****************INEQ. CONST B(0, ,NI) RIGHT  HAND SIDE**************/
                          /*RIGHT SIDE OF EQUALITY CONSTRAINTS*/
B[0]=0;             /*RIGHT SIDE OF INEQUALITY CONSTRAINTS*/
/*************************************************************************/
if((C=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
if((D=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
/*****ENTER  THE UPPER AND LOWER  BOUNDS  C(0,,N) LOWER BOUND*****/
/** D(0,,N) UPPER  BOUND  VALUES, DON'T WRITE  IF  NOT  BOUNDED  *****/
C[0]=100; C[1]=0.3; C[2]=0.3;
D[0]=250; D[1]=0.5; D[2]=0.5;
/*************************************************************************/
NN=N*N;  NSRC=(2*N)+1;
/*if((FE=(double*) malloc(NE*sizeof(double)))==NULL) exit(1);*/
if((FI=(double*) malloc(NI*sizeof(double)))==NULL) exit(1);
//if((FES=(double*) malloc(NE*sizeof(double)))==NULL) exit(1);
if((FIS=(double*) malloc(NI*sizeof(double)))==NULL) exit(1);
if((GAL=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
if((HVL=(double*) malloc(NBV*sizeof(double)))==NULL) exit(1);
if((HVU=(double*) malloc(NBV*sizeof(double)))==NULL) exit(1);
if((GF=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
//if((GE=(double*) malloc(NE*N*sizeof(double)))==NULL) exit(1);
if((GI=(double*) malloc(NI*N*sizeof(double)))==NULL) exit(1);
if((X=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
if((XT=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
if((XP=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
//if((HE=(double*) malloc(NE*sizeof(double)))==NULL) exit(1);
if((HI=(double*) malloc(NI*sizeof(double)))==NULL) exit(1);
if((GO=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
if((dfp=(double*) malloc((N+NN+3)*sizeof(double)))==NULL) exit(1);
if((H=(double*) malloc(NN*sizeof(double)))==NULL) exit(1);
if((DLG=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
if((DLX=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
if((R=(double*) malloc(N*sizeof(double)))==NULL) exit(1);
/*INITIAL CONDITIONS*/
EPS1=1e-6; EPS2=1e-4; EPS3=1e-4;
X[0]=100; X[1]=0.3; X[2]=0.3;  /*initial states*/
for(i=0;i<N;i++) DLX[i]=DLG[i]=0;
for(i=0;i<NN;i++) H[i]=0; for(i=0;i<(NN+2);i++) dfp[i]=0;
if ((NE+NI+NBV)!=0){ W1=1; W2=1; W3=1; WF=2; WMAX=32;}
for(i=0;i<NE;i++) HE[i]=0; for(i=0;i<NI;i++) HI[i]=0;/*alfa-beta*/
for(i=0;i<N;i++) { HVL[i]=0; HVU[i]=0;} /*bounded variable multipliers*/
F=FXNS(X); printf("F=%lg ",F);/*COST FUNCTION INITIALIZATION*/
if(NE!=0){ for(i=0;i<NE;i++)  FE[i]=FXFE(X,i);}/*left side of equality constr*/
if(NI!=0){ for(i=0;i<NI;i++)  FI[i]=FXFI(X,i);}/*left side of inequality costr*/
/*printf("  FE= %lg FI=%lg\n" ,FE[0],FI[0]);*/
RHO=GMAG=0;IBSD=IT=itot=0;
AL=AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV);
/*AUGMENTED LAGRANGE FUNCT*/
/*printf(" AL= %lg\n",AL);*/
```

```
if(NE!=0){ for(i=0;i<NE*N;i++)  GE[i]=GFE(X,i);}
if(NI!=0){ for(i=0;i<NI*N;i++)  GI[i]=GFI(X,i);}
for(i=0;i<N;i++) GF[i]=GFF(X,i);
*GAL=GALAG(X,A,B,C,D,FE,FI,GF,GE,GI,HE,HI,HVL,HVU,W1,W2,W3,XB,GAL,N,NE,NI,N
BV);
GMAG=0;
for(i=0;i<N;i++){ GO[i]=GAL[i];/* printf(" GAL=%lg ",GAL[i]);*/
GMAG=GMAG+GAL[i]*GAL[i]; }
GMAG=sqrt(GMAG); /* NORM OF GRADIENT printf(" GMAG=%lg \n",GMAG);*/
EPD=0.1*EPS1; EPT=10*EPS2; EPV=EPS2;
ICONV=IFAIL=NBD=0; KSRC=0;
FS=F;
if(NE!=0){ for(i=0;i<NE;i++) FES[i]=FE[i]; }
if(NI!=0){ for(i=0;i<NE;i++) FIS[i]=FI[i]; }
NG=1; RHO=0;
        /*    ITERATIONS          for(itot=0;itot<50;itot++){*/
 ST10:
 if(itot==200) goto  ST80; printf("  itot = %d  ",itot);
 *dfp=DFPRV(GAL,RHO,DLG,DLX,H,R,dfp,NG,KSRC,IFAIL,N);
for(i=0;i<NN;i++) H[i]=dfp[i]; SLOPE=dfp[NN];RMAG= dfp[NN+1];NG=dfp[NN+2];
for(i=NN+3;i<(NN+N+3);i++) R[(i-NN-3)]=dfp[i];
for(i=0;i<(N+NN+3);i++)  printf("df= %lg   ",dfp[i]);
if(NG==2) goto ST12; if(NG>2) goto ST14;
NBD=NBD+1;
if(NBD>=5)  goto  ST80; else  goto ST14;
ST12: NBD=0;
ST14:
/*SEARCH ALGORITHM-CONDUCTS SEARCH ALONG R TO FIND STEPSIZE RHO THAT
MAXIMIZES AUGMENTED LAGRANGIAN, AL=F(X+RHO*R)*/
DS=0; YS=AL; D1=0; IC1=0; Y1=AL; IUP=0; D2=1;
if(NG==1) D2=0.1; if((KSRC==0)&&(IFAIL>0)) D2=0.05;
if(RMAG>=200) D2=10/RMAG; if(D2<0.001) D2=0.001;
/*VALUE-EVALUATES AUGMANTED LAGRANGIAN Y AT STEPSIZE D*/
DT=D2;/*printf(" D2=%lg \n" , D2);*/
/*VALUE*/ for(i=0;i<N;i++) { XT[i]=X[i]+DT*R[i]; XP[i]=X[i]; X[i]=XT[i];}
F=FXNS(X);for(i=0;i<NE;i++)  FE[i]=FXFE(X,i);
 for(i=0;i<NI;i++)  FI[i]=FXFI(X,i);
Y=AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV);
if(Y>YS) IUP=1; if(Y>=YS) { for(i=0;i<NE;i++)FES[i]=FE[i];
for(i=0;i<NI;i++) FIS[i]=FI[i]; FS=F; DS=DT; YS=Y;} /*VALUE*/
/*printf("y=%lg\n",Y);*/
Y2=Y; K=1; for(i=0;i<N;i++){  X[i]=XP[i]; printf(" X=%lg",X[i]);}
/*UNIDIRECTIONAL SEARCH*/
D3=DMAX(D1,D2,D3,Y1,Y2,Y3,SLOPE,K);/* printf(" D3=%lg \n" , D3);*/
if((Y2-Y1)<0) goto S50; if((Y2-Y1)==0) goto S200; goto S20;
S20:
if (D3<=0) goto S200; if((D3>=(0.9*D2))&&(D3<=(1.1*D2))) goto S410;
S25:
if(D3<=(5*D2)) goto S340; if(D3>(100*D2)) D3=100*D2;
DT=D3;
/*VALUE*/ for(i=0;i<N;i++) { XT[i]=X[i]+DT*R[i]; XP[i]=X[i]; X[i]=XT[i];}
F=FXNS(X);for(i=0;i<NE;i++)  FE[i]=FXFE(X,i);
 for(i=0;i<NI;i++)  FI[i]=FXFI(X,i);
Y=AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV);
```

```
if(Y>YS) IUP=1; if(Y>=YS){ for(i=0;i<NE;i++)FES[i]=FE[i];
for(i=0;i<NI;i++)FIS[i]=FI[i]; FS=F; DS=DT; YS=Y;} /*VALUE*/
Y3=Y; for(i=0;i<N;i++) X[i]=XP[i];
if((Y3-Y2)<0) goto S200; if((Y3-Y2)==0) goto S200; goto S30;
S30:
D2=D3; Y2=Y3; goto S300;
S50:
if(D3>(0.2*D2)) goto S100; if(D3<(0.01*D2)) D3=0.01*D2;
DT=D3;
/*VALUE*/ for(i=0;i<N;i++) { XT[i]=X[i]+DT*R[i]; XP[i]=X[i]; X[i]=XT[i];}
F=FXNS(X);for(i=0;i<NE;i++) FE[i]=FXFE(X,i);
 for(i=0;i<NI;i++) FI[i]=FXFI(X,i);
Y=AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV);
if(Y>YS) IUP=1; if(Y>=YS) {for(i=0;i<NE;i++)FES[i]=FE[i];
for(i=0;i<NI;i++)FIS[i]=FI[i]; FS=F; DS=DT; YS=Y;} /*VALUE*/
Y3=Y; D2=D3; Y2=Y3;  for(i=0;i<N;i++) X[i]=XP[i];
if((Y2-Y1)<0) goto S60; if((Y2-Y1)==0) goto S200; goto S300;
S60:
if(IC1>=0) goto S410; IC1++; K=1;
D3=DMAX(D1,D2,D3,Y1,Y2,Y3,SLOPE,K);/* printf(" D3=%lg \n" , D3);*/
goto S50;
S100:
DT=D3;
/*VALUE*/ for(i=0;i<N;i++) {XT[i]=X[i]+DT*R[i]; XP[i]=X[i]; X[i]=XT[i];}
F=FXNS(X);for(i=0;i<NE;i++) FE[i]=FXFE(X,i);
 for(i=0;i<NI;i++) FI[i]=FXFI(X,i);
Y=AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV);
if(Y>YS) IUP=1; if(Y>=YS){ for(i=0;i<NE;i++)FES[i]=FE[i];
for(i=0;i<NI;i++)FIS[i]=FI[i]; FS=F; DS=DT; YS=Y;} /*VALUE*/
Y3=Y; for(i=0;i<N;i++) X[i]=XP[i];
if((Y3-Y1)<0) goto S120; if((Y3-Y1)==0) goto S410; goto S350;
S120:
D2=0.2*D3; IC1++;  DT=D2;
/*VALUE*/ for(i=0;i<N;i++) {XT[i]=X[i]+DT*R[i]; XP[i]=X[i]; X[i]=XT[i];}
F=FXNS(X);for(i=0;i<NE;i++) FE[i]=FXFE(X,i);
for(i=0;i<NI;i++) FI[i]=FXFI(X,i);
Y=AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV);
if(Y>YS) IUP=1; if(Y>=YS){ for(i=0;i<NE;i++)FES[i]=FE[i];
for(i=0;i<NI;i++)FIS[i]=FI[i]; FS=F; DS=DT; YS=Y;} /*VALUE*/
Y2=Y; for(i=0;i<N;i++) X[i]=XP[i];
if((Y2-Y1)<0) goto S150; if((Y2-Y1)==0) goto S410; goto S240;
S150:
if(IC1>10) goto S240;
S160:
D3=D2; Y3=Y2; goto S120;
S200:
D3=5*D2; IC1++;
DT=D3;
/*VALUE*/ for(i=0;i<N;i++) {XT[i]=X[i]+DT*R[i]; XP[i]=X[i]; X[i]=XT[i];}
F=FXNS(X);for(i=0;i<NE;i++) FE[i]=FXFE(X,i);
 for(i=0;i<NI;i++) FI[i]=FXFI(X,i);
Y=AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV);
if(Y>YS) IUP=1; if(Y>=YS){ for(i=0;i<NE;i++)FES[i]=FE[i];
for(i=0;i<NI;i++)FIS[i]=FI[i]; FS=F; DS=DT; YS=Y;} /*VALUE*/
```

```
Y3=Y; for(i=0;i<N;i++) X[i]=XP[i];
if((Y3-Y2)<0) goto S220; if((Y3-Y2)==0) goto S210; goto S210;
S210:
if(IC1>10) goto S240; D1=D2; Y1=Y2; D2=D3; Y2=Y3; goto S200;
S220:
if(IUP<=0) goto S410; goto S240;
S240:
K=2; RHO=DMAX(D1,D2,D3,Y1,Y2,Y3,SLOPE,K);/* printf(" Rho=%lg " ,RHO);*/
goto S400;
S300:
K=1; D3=DMAX(D1,D2,D3,Y1,Y2,Y3,SLOPE,K);/* printf(" D3=%lg " , D3);*/
if((D3>=(0.9*D2))&&(D3<=(1.1*D2))) goto S410; if (D3<=0) goto S200;
if(IC1>10) goto S410; IC1++; goto S25;
S340:
DT=D3;
/*VALUE*/ for(i=0;i<N;i++) {XT[i]=X[i]+DT*R[i]; XP[i]=X[i]; X[i]=XT[i]; }
FXNS(X);for(i=0;i<NE;i++) FE[i]=FXFE(X,i);
for(i=0;i<NI;i++) FI[i]=FXFI(X,i);
Y=AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV);
if(Y>YS) IUP=1; if(Y>=YS){ for(i=0;i<NE;i++)FES[i]=FE[i];
for(i=0;i<NI;i++)FIS[i]=FI[i]; FS=F; DS=DT; YS=Y;} /*VALUE*/
Y3=Y; for(i=0;i<N;i++) X[i]=XP[i];
if((Y3-Y2)<0) goto S240; if((Y3-Y2)==0) goto S350; goto S350;
S350:
K=2; RHO=DMAX(D1,D2,D3,Y1,Y2,Y3,SLOPE,K);/* printf(" Rho=%lg " ,RHO);*/
if((RHO>=(0.9*D3))&&(RHO<=(1.1*D3))) goto S410;
/*BEST STEPSIZE RHO FOUND FOR THIS SEARCH*/
S400:
DT=RHO;
/*VALUE*/ for(i=0;i<N;i++) {XT[i]=X[i]+DT*R[i]; XP[i]=X[i]; X[i]=XT[i]; }
F=FXNS(X);for(i=0;i<NE;i++) FE[i]=FXFE(X,i);
 for(i=0;i<NI;i++) FI[i]=FXFI(X,i);
Y=AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV);
if(Y>YS) IUP=1; if(Y>=YS){ for(i=0;i<NE;i++)FES[i]=FE[i];
for(i=0;i<NI;i++)FIS[i]=FI[i]; FS=F; DS=DT; YS=Y;} /*VALUE*/
AL=Y; for(i=0;i<N;i++) X[i]=XP[i];
S410:
RHO=DS; for(i=0;i<N;i++) XT[i]=X[i]+RHO*R[i];
for(i=0;i<NE;i++) FE[i]=FES[i];
for(i=0;i<NI;i++) FI[i]=FIS[i]; F=FS; AL=YS;
for(i=0;i<N;i++) printf("  LX=%lg \n",XT[i]); printf("F=%lg\n" ,F );
                          /*END OF SEARCH */

if(RHO==0) goto  ST15;
if(NE!=0){ for(i=0;i<NE*N;i++)  GE[i]=GFE(XT,i);}
if(NI!=0){ for(i=0;i<NI*N;i++)  GI[i]=GFI(XT,i);}
for(i=0;i<N;i++) GF[i]=GFF(XT,i);                    NGE++;
*GAL=GALAG(XT,A,B,C,D,FE,FI,GF,GE,GI,HE,HI,HVL,HVU,W1,W2,W3,XB,GAL,N,NE,NI,
NBV);
ST15:
/*COMPUTES NORM(XNEW-XOLD), NORM(GNEW-GOLD)*/
DELG=GMAG=0; for(i=0;i<N;i++) { DLX[i]=XT[i]-X[i]; DLG[i]=GAL[i]-GO[i];
DELG=DELG+DLG[i]*DLG[i];  GMAG=GMAG+GAL[i]*GAL[i];X[i]=XT[i]; GO[i]=GAL[i];}
DELX=RHO*RMAG; DELG=sqrt(DELG); GMAG=sqrt(GMAG);/***delta***/
```

```
itot++;  KSRC++;
ST20:
if(IUP==0) goto ST24; IFAIL=0;
if(DELX>EPS3)  goto  ST28;        if(GMAG<EPV) goto  ST40;
if(DELG>EPV) goto ST28;           if(NG!=1)  goto  ST26;
if((DELG<EPD)&&(KSRC==1)) goto  ST80; goto  ST26;
ST24:
IFAIL++; if(IFAIL==4) goto  ST80;          if((NG==1)||(RHO==0)) goto  ST40;
ST26:
IBSD++;
ST28:
if(KSRC>N)  goto  ST30;           if(IBSD==2)  KSRC=N;  goto  ST10;
ST30:
if(KSRC==NSRC) goto  ST40;        if(GMAG<=EPV) goto  ST40;
ST35:
if(IBSD<3)  goto  ST10; RHO=0;
ST40:
/*MULTIPLIERS AND  PENALTY WEIGHTS UPDATED BY UPDATE RULES*/
for(i=0;i<NE;i++) HE[i]=HE[i]-2*W1*(A[i]-FE[i]);/*equal□ty  cons.  update*/
for(i=0;i<NI;i++)  { TEMP=B[i]-FI[i] ; /*inequality  cons.  update*/
if(HI[i]<=0) {if(TEMP<0) HI[i]=-2*W3*TEMP;}
else  { HI[i]=HI[i]-2*W2*TEMP;  if(HI[i]<0)  HI[i]=0;} }
if(NBV!=0){/*BOUNDED  VARIABLE UPDATE*/
for(i=0;i<N;i++){ IVAR=XB[i];
if((IVAR==1)||(IVAR==3)){         TEMP=X[i]-C[i];
if(HVL[i]<=0)&&(TEMP<0)) HVL[i]=-2*W3*TEMP;
if(HVL[i]>0) {  HVL[i]=HVL[i]-2*W2*TEMP; if(HVL[i]<0) HVL[i]=0;}}
if((IVAR==2)||(IVAR==3)){
TEMP=D[i]-X[i];
if(HVU[i]>0){ HVU[i]=HVU[i]-2*W2*TEMP; if(HVU[i]<0) HVU[i]=0;}
else { if(TEMP<0) HVU[i]=-2*W3*TEMP; }}    }}
/*update we□ghts*/
 W1=WF*W1;   W2=WF*W2;  W3=WF*W3 ;
if(W1>WMAX) W1=WMAX;
if(W2>WMAX) W2=WMAX;
if(W3>WMAX) W3=WMAX;
F=FXNS(X);/* printf("F=%lg ",F);*/
if(NE!=0){ for(i=0;i<NE;i++)  FE[i]=FXFE(X,i);}
if(NI!=0){ for(i=0;i<NI;i++)  FI[i]=FXFI(X,i);}
AL=AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV);
if(NE!=0){ for(i=0;i<NE*N;i++)  GE[i]=GFE(X,i);}
if(NI!=0){ for(i=0;i<NI*N;i++)  GI[i]=GFI(X,i);}
for(i=0;i<N;i++) GF[i]=GFF(X,i);
*GAL=GALAG(X,A,B,C,D,FE,FI,GF,GE,GI,HE,HI,HVL,HVU,W1,W2,W3,XB,GAL,N,NE,NI,N
BV);
GMAG=0;for(i=0;i<N;i++) GO[i]=GAL[i];
ST60:
for(i=0;i<N;i++) GMAG=GMAG+GAL[i]*GAL[i]; GMAG=sqrt(GMAG);
/*printf("GMAG=%lg  DELX=%lg" ,AL,DELX  );*/
ST70:
KSRC=0; IBSD=0; if(GMAG<EPT)  EPV=EPS1;
if((GMAG>EPS1)||(DELX>EPS3)) goto  ST10;
ICONV=1;
ST80:
```

```
if(ICONV==1) printf("CONVERGENCE  OCCURED\n");
if(ICONV==0)  printf(" POOR CONVERGENCE!\n");
printf(" alfa= %lg\n", HE[0]);
for(i=0;i<N;i++) printf(" X= %lg ",X[i]);
printf (" GMAG = %5lg F= %5lg\n",GMAG,F);
printf (" FE = %5lg FI1= %5lg FI2= %5lg FI3= %5lg  \n",FE[0],FI[0],FI[1],FI[2]);
}

/*FUNCTION COMPUTES AUGMENTED LAGRANGE*/
double AUGLAG(X,FE,FI,HE,HI,HVL,HVU,A,B,C,D,F,XB,W1,W2,W3,N,NE,NI,NBV)
double *X,*FE,*FI,*HE,*HI,*HVL,*HVU,*A,*B,*C,*D,F,W1,W2,W3;
int *XB,N,NE,NI,NBV;
{
double temp,P1,P2,P3,flag,ALG;
int ivar,i;
flag=P1=P2=P3=temp=0;

/*EQUALITY CONSTRAINT COMPONENTS*/
if(NE!=0) { for(i=0;i<NE;i++) { temp=A[i]-FE[i]; flag=flag+HE[i]*temp;
P1=P1+temp*temp;}}
/*INEQUALITY CONSTRAINT COMPONENTS*/
if(NI!=0) { for(i=0;i<NI;i++)
{  temp=B[i]-FI[i ];
  if((HI[i]<=0) && (temp<0))P3=P3+temp*temp;
  if(HI[i]>0) {flag=flag+HI[i]*temp; P2=P2+temp*temp;}}}
/*BOUNDED VARIABLES*/
if(NBV!=0)
{  for (i=0;i<N;i++) { ivar=XB[i];
 if((ivar==1)||(ivar==3)) {  temp=X[i]-C[i];
 if((HVL[i]<=0) && (temp<0))P3=P3+temp*temp;
  if(HVL[i]>0) {flag=flag+HVL[i]*temp; P2=P2+temp*temp;}}
 if((ivar==2)||(ivar==3)) {  temp=D[i]-X[i];
 if((HVU[i]<=0) && (temp<0)) P3=P3+temp*temp;
  if(HVU[i]>0) {flag=flag+HVU[i]*temp; P2=P2+temp*temp;}}}}
/* AL EQUALS THE SUM OF ALL COMPONENTS */
ALG=F+flag-W1*P1-W2*P2-W3*P3;        /*printf("FLAG= %lg  ALG=%lg\n",flag, ALG);*/
return (ALG);}


/*FUNCTION COMPUTES AUGMENTED LAGRANGE GRADIENT*/

double
GALAG(X,A,B,C,D,FE,FI,GF,GE,GI,HE,HI,HVL,HVU,W1,W2,W3,XB,GAL,N,NE,NI,NBV)
double *GAL,*X,*A,*B,*C,*D,*FE,*FI,*GF,*GE,*GI,*HE,*HI,*HVL,*HVU,W1,W2,W3;
int *XB,N,NE,NI,NBV;
{
int i,indx,k,ivar;
double glag,gew1,giw2,giw3,temp;
for(k=0;k<N;k++)
{
glag=gew1=giw2=giw3=0;
if(NE!=0) {indx=k;/*equality constraints*/
            for(i=0;i<NE;i++) { glag=glag+HE[i]*GE[indx];
            gew1=gew1+(A[i]-FE[i])*GE[indx]; indx=indx+N;}}
```

```
if(NI!=0) { indx=k; /*inequality constraints*/
          for(i=0;i<NI;i++) { temp=B[i]-FI[i];
if((HI[i]<=0)&&(temp<0)){ giw3=giw3+temp*GI[indx]; indx=indx+N;}
if(HI[i]>0) {glag=glag+HI[i]*GI[indx]; giw2=giw2+temp*GI[indx];
indx=indx+N;}}}
          if(NBV!=0) { ivar=XB[k];
          if((ivar==1)||(ivar==3)) {/*lower bound variables*/
          temp=X[k]-C[k];
 if(HVL[k]<=0)&&(temp<0)) giw3=giw3-temp;
 if(HVL[k]>0) { glag=glag-HVL[k]; giw2=giw2-temp;}}
 if((ivar==2)||(ivar==3)){/*upper bound variables*/
 temp=D[k]-X[k];
 if((HVU[k]<=0)&&(temp<0)) giw3=giw3+temp;
 if(HVU[k]>0) { glag=glag+HVU[k]; giw2=giw2+temp;}}}
/*GAL EQUALS THE SUM OF THE COMPONENTS*/
GAL[k]= GF[k]-glag+2*(W1*gew1+W2*giw2+W3*giw3);
/*printf(" gew1=%lg  ",gew1);*/
}
return (*GAL);}

/*COMPUTES SEARCH DIRECTION R, VIA DFP METHOD WITH RESETS OF R=GAL*/

double DFPRV(GAL,RHO,DLG,DLX,H,R,dfp,NG,KSRC,IFAIL,N)
double *dfp,*GAL,*H,*DLX,*DLG,RHO,*R,NG;  int IFAIL,KSRC,N;
{
double SLOPE,con1,con2,con3,con4,con5,GAMA,RMAG,TFM,TEMP;
int NN,i,k,j,INDX,ISS,itern,L;
NG=NG+1;  NN=N*N;ISS=1;
if(RHO==0) goto A;
if((KSRC==0)&&(IFAIL>0)) goto A;
if((KSRC!=0)||(NG<=N)) { printf("  NG= %lg",NG); goto B;}
A:
NG=1; SLOPE=0; INDX=0;  for(i=0;i<NN;i++) H[i]=0;
for(i=0;i<N;i++) { H[INDX+i]=1; R[i]=GAL[i]; SLOPE=SLOPE+(R[i]*R[i]);
INDX=INDX+N;} RMAG=sqrt(SLOPE);
for(i=0;i<NN;i++) dfp[i]=H[i]; dfp[NN]=SLOPE; dfp[NN+1]=RMAG;dfp[NN+2]=NG;
for(i=(NN+3);i<(N+NN+3);i++)  dfp[i]=R[(i-NN-3)];
return (*dfp);
B:/*R VIA DFP OR DFPSS*/
con1=con2=con3=con4=INDX=0;
for(i=0;i<N;i++) { TFM=0;
          for(j=0;j<N;j++) TFM=TFM+H[INDX+j]*DLG[j];
R[i]=TFM; con1=con1+DLX[i]*DLG[i]; con2=con2+DLG[i]*R[i];
if(ISS!=0){  TEMP=GAL[i]-DLG[i]; con3=con3+TEMP*DLX[i]; con4=con4+TEMP*R[i];}
INDX=INDX+N;}
if((con1==0)||(con2==0)) goto A;   if((ISS!=0)&&(con4!=0)) {
for(i=0;i<N;i++) DLG[i]=R[i]/con2-DLX[i]/con1;  con5=con2/2;
GAMA=-con3/con4; } INDX=0;
for(k=0;k<N;k++) { itern=k*N;     for(j=k;j<N;j++) { L=INDX+j;
          if((ISS!=0)&&(con4!=0))
H[L]=(H[L]-R[k]*R[j]/con2+con5*DLG[k]*DLG[j])*GAMA-DLX[k]*DLX[j]/con1;
  else  H[L]=H[L]-DLX[k]*DLX[j]/con1-R[k]*R[j]/con2;
  H[itern+k]=H[L]; itern=itern+N;}
  INDX=INDX+N;}   INDX=0; for(i=0;i<N;i++){ TFM=0; for(j=0;j<N;j++)
```

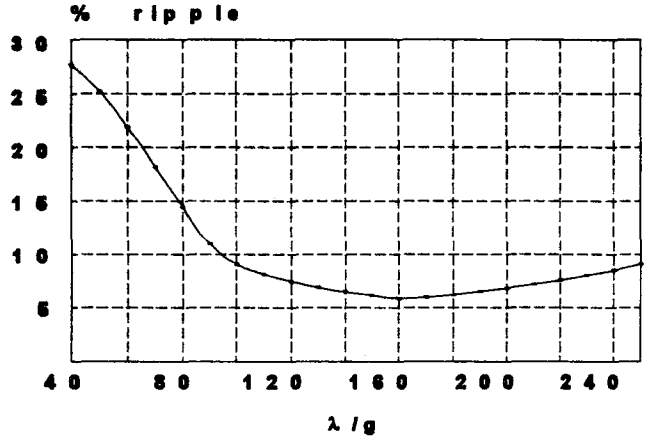
```
  TFM=TFM+H[INDX+j]*GAL[j]; R[i]=TFM; INDX=INDX+N;}
/*TEST FOR A  GOOD R; IF R BAD, SET R=GAL*/
RMAG=SLOPE=0;
for(i=0;i<N;i++){ RMAG=RMAG+R[i]*R[i]; SLOPE=SLOPE+R[i]*GAL[i];}
RMAG=sqrt(RMAG);
for(i=0;i<NN;i++) dfp[i]=H[i]; dfp[NN]=SLOPE; dfp[NN+1]=RMAG;dfp[NN+2]=NG;
for(i=(NN+3);i<(N+NN+3);i++)  dfp[i]=R[(i-NN-3)];
if(SLOPE>0) return (*dfp); else goto A;
}

double DMAX(D1,D2,D3,Y1,Y2,Y3,SLOPE,K)
double D1,D2,D3,Y1,Y2,Y3,SLOPE; int  K;

{/*ASSUMES QUADRATIC FORM AL=A*(D1-ALF)**2+B*(D1-ALF)+C
 MAXIMUM ALF=D1-B/2A FOUND BY QUADRATIC FIT OF DATA*/
double tp1,dif,tp2,d21,d31,ALF;
if(K==1) { tp1=-SLOPE*D2; dif=Y2-Y1+tp1;if(dif!=0){ ALF=(tp1*D2)/(2*dif);
return (ALF);} else return (25*D2);}
if(K==2) { d21=D2-D1; d31=D3-D1; tp1=d31*(Y2-Y1); tp2=d21*(Y3-Y1);
dif=tp1-tp2; if(dif==0)  return (25*D2);
ALF=D1-0.5*(d21*tp2-d31*tp1)/dif; return (ALF);   }
}


double FXNS(X)  /* FUNCTION USED FOR CALCULATION OF F(X)*/
#define boy 21
double *X;
{
double F;

/*****  WRITE  THE FUNCTION TO  BE  MAXIMIZED  **********/
FILE *ff;
double u[6],TR[10],w[4][boy][boy],bt[boy][boy],S1,S2,xn,max,RPL;
int k,in,N1,N2,on,j,p,i;
in=5; on=1; N1=20; N2=10;
for(i=0;i<6;i++) u[i]=0;
/*************************************************************************/
if((ff=fopen("suq1.dat","r"))==NULL) {
        printf("can not open the file\n");    exit(0);  }
/*****************************force NN*************************************/
fscanf(ff,"%d %d %d %d\n",&in ,&N1,&N2, &on);
for(i=1;i<in+1;i++){for(k=1;k<N1+1;k++){fscanf(ff,"%lg  \n",&w[1][i][k]);}}
for(k=1;k<N1+1;k++){for(p=1;p<N2+1;p++){fscanf(ff,"%lg  \n",&w[2][k][p]);}}
for(p=1;p<N2+1;p++){for(j=1;j<on+1;j++){fscanf(ff,"%lg  \n",&w[3][p][j]);}}
for(k=1;k<N1+1;k++){fscanf(ff," %lg  \n",&bt[1][k]);}
for(p=1;p<N2+1;p++){fscanf(ff,"%lg  \n",&bt[2][p]);}
for(j=1;j<on+1;j++){fscanf(ff,"%lg  \n",&bt[3][j]);}
/*************************************************************************/
X[0]=(X[0]/250);
u[1]=X[0];
u[2]=X[1];
u[3]=X[2];
/*Enter the mmf value for a UDSS*/
u[5]=(50000*0.0172)/1500;
```

```
xn=0.1;  max=0;
for(i=0;i<10;i++)
{
        u[4]=xn;          S1=OUTPUT(u,w,bt);   if(xn<=0.5) u[4]=(xn+0.5);else u[4]=(xn-0.5);
S2=OUTPUT(u,w,bt);     if(S1>=S2) TR[i]=S1; else TR[i]=S2;        xn=xn+0.1;
if(TR[i]>=max) max=TR[i];/*peak value of torque ripple curve*/
printf("torque = %lg \n",TR[i]);
}
RPL=0;
for(i=0;i<10;i++)  RPL=RPL+0.5*pow(((max-TR[i])/max),2);
printf("ripple function = %lg ",RPL);   fclose (ff);    return RPL;
}

double OUTPUT(u,w,bt)
#define boy 21
double u[6],w[4][boy][boy],bt[boy][boy];

{
double big,t;
int a,b,c,i,j,jj,nn,l,kk,ii,N1,N2,iv,in,on,n,k,p;/**x1,*x2,*z1,*z2,*yd;*/
double ut,yt,sum,ex1,out,cons,ex,x1[21],x2[11],z1[21],z2[11],yd[2];
int dimu,dimy,dimx1,dimx2,dd,dim;
N1=20;N2=10;   iv=1; in=5; on=1;                    cons=10;
for(ii=1;ii<iv+1;ii++)
    {    for(k=1;k<N1+1;k++)     {
                sum=0;  for(n=1;n<in+1;n++)      {    a=in*(ii-1)+n;
                        sum=w[1][n][k]*u[a]+sum;}
                c=N1*(ii-1)+k;   x1[c]=sum+bt[1][k];   z1[c]=tanh(cons*x1[c]);   }
        for(p=1;p<N2+1;p++)
        {       sum=0;  for(k=1;k<N1+1;k++)     {  c=N1*(ii-1)+k;
                        sum=w[2][k][p]*z1[c]+sum;   }
                b=N2*(ii-1)+p;   x2[b]=sum+bt[2][p];   z2[b]=tanh(cons*x2[b]);}
                for(j=1;j<on+1;j++)
                { sum=0;  for(p=1;p<N2+1;p++)  {          b=N2*(ii-1)+p;
        sum=w[3][p][j]*z2[b]+sum; }
                dd=on*(ii-1)+j;   yd[dd]=sum+bt[3][j];  }}
for(ii=1;ii<iv+1;ii++)
{  for(i=1;i<in+1;i++)  { a=in*(ii-1)+i;      //printf(" in= %lg " ,u[a]); }
for(j=1;j<on+1;j++) {      dd=on*(ii-1)+j;   yd[dd]=(yd[dd]*1500)/0.0172;
//printf(" out= %lg  ",yd[dd] );       }    //printf("\n"); }
out=yd[1];  return(out);  }


double FXFE(X,i) /*CALCULATION OF EQUALITY CONSTRAINTS*/
double *X; int i;
/***ENTER THE LEFT HAND  SIDE OF EQUALITY  CONSTRAINTS FROM I=0 TO NE***/
{        }

double FXFI(X,i) /*CALCULATION OF INEQUALITY CONSTRAINTS*/
double *X; int i;
/*** ENTER THE LEFT HAND  SIDE OF INEQUALITY  CONSTRAINTS FROM I=0 TO
NI***/
{ if(i==0) return (X[1]-X[2])}
```

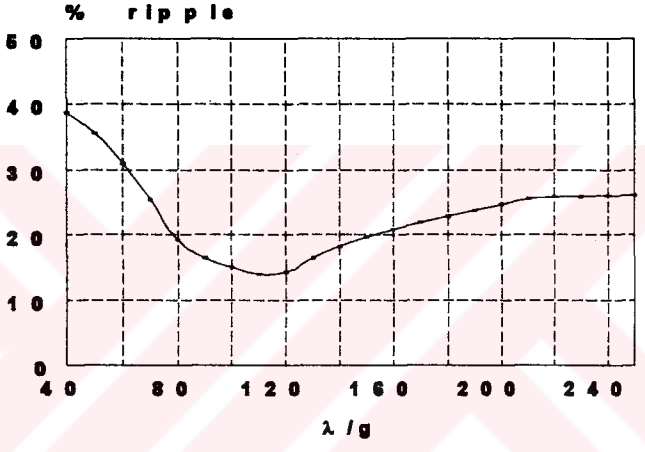
```
double GFE(X,i) /*calculation of equality consraint gradient*/
double *X; int i;
/*ENTER THE GRADIENT COMPONENTS FROM I=0 TO N*NE*/
/*NE(0)(0),NE(0)(1), .. FIRST # OF EQ. C. SECOND #OF VARIABLE*/
{       }

double GFI(X,i) /*calculation of inequality consraint gradient*/
double *X; int i;
/*ENTER THE GRADIENT COMPONENTS FROM I=0 TO N*NI*/
{ if(i==0) return (0);
  if(i==1) return (1);
  if(i==2) return (-1);     }

double GFF(X,i) /*calculation of function gradient*/
double *X; int i;
/*ENTER THE GRADIENT COMPONENTS OF F FROM I=1 TO N*/

{ if(i==0) return (0.5);
  if(i==1) return (1);
  if(i==2) return (0.5);
  if(i==3) return (1);
   }
```

B2. Graphs Presenting the Effect of λ/g and t_s/λ on Torque Ripple for Different mmf values

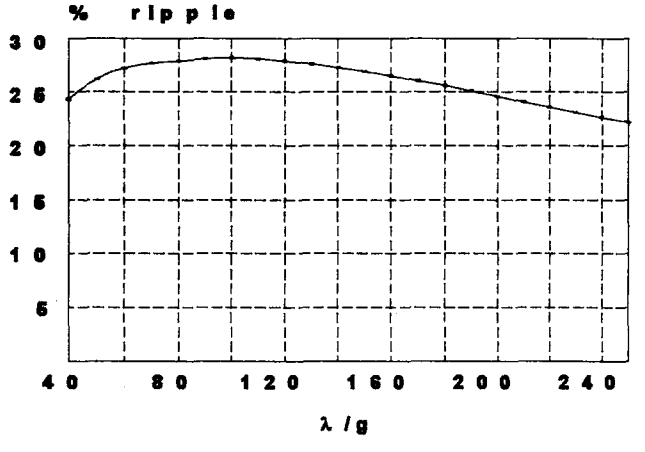
Following graphs are skethed to present the outstanding results of this optimization study as discussed in Chapter 4. The first four set of graps (Fig. B1-B4) shows percentage ripple as a function of λ/g, each for different mmf values (30, 40, 60, 70 kA for udss). Each set includes six different graphs for six different combinations of t_s/λ and t_r/λ. The next four set of graphs (Fig. B5-B8) for the same set of mmf values shows percentage ripple as a function of t_s/λ, each for three different values of t_r/λ (0.3, 0.4, 0.5). Each graph in this second set of four shows four different curves, each for a different value of λ/g.



(a) t_s/λ=0.4, t_r/λ=0.5

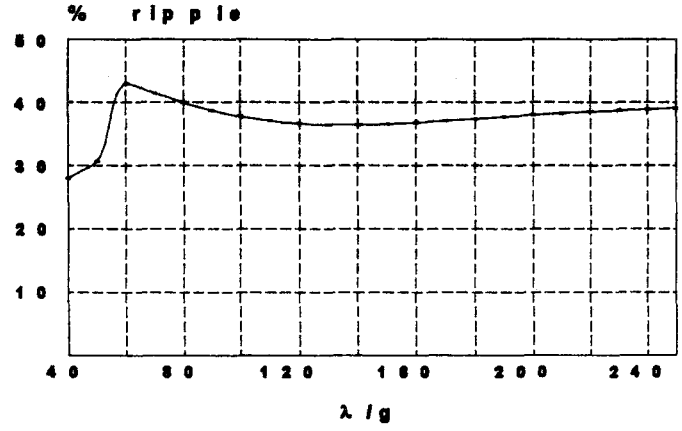


(b) t_s/λ=0.3, t_r/λ=0.4

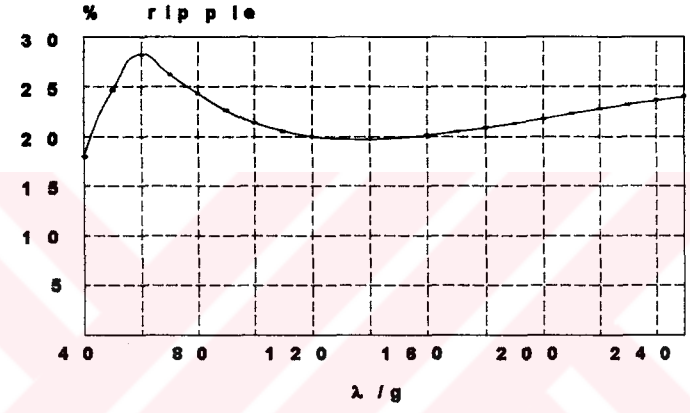


(c) t_s/λ=0.3, t_r/λ=0.5

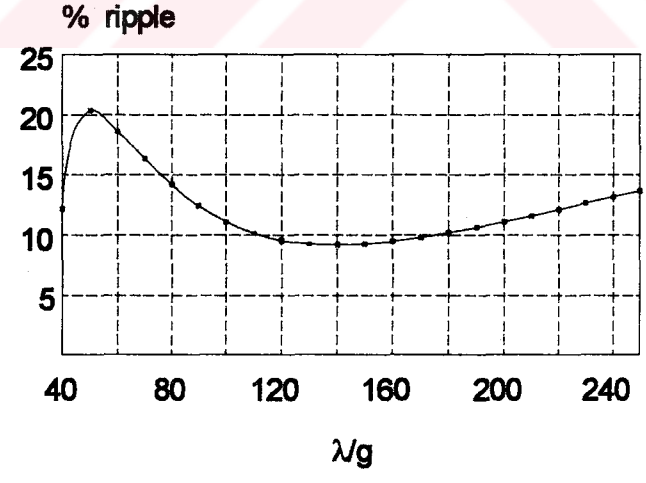
Figure B.1 Effect of λ/g on torque ripple for different t_s/λ, t_r/λ values at 30 kA



(d) $t_s/\lambda=0.3$, $t_r/\lambda=0.3$

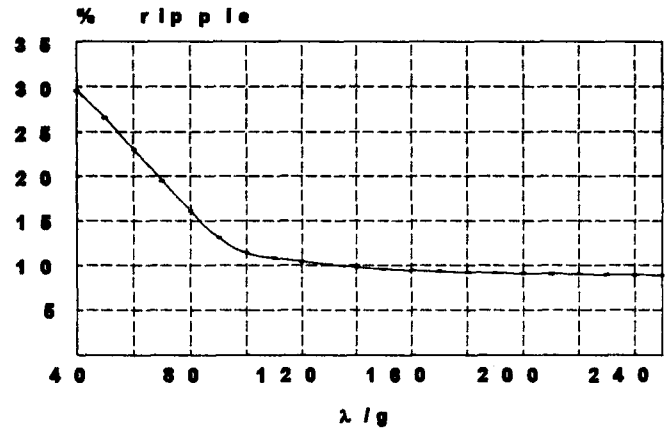


(e) $t_s/\lambda=0.4$, $t_r/\lambda=0.4$

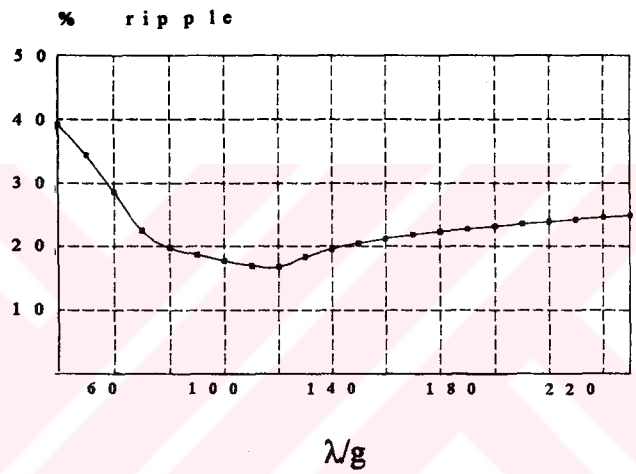


(a) $t_s/\lambda=0.5$, $t_r/\lambda=0.5$

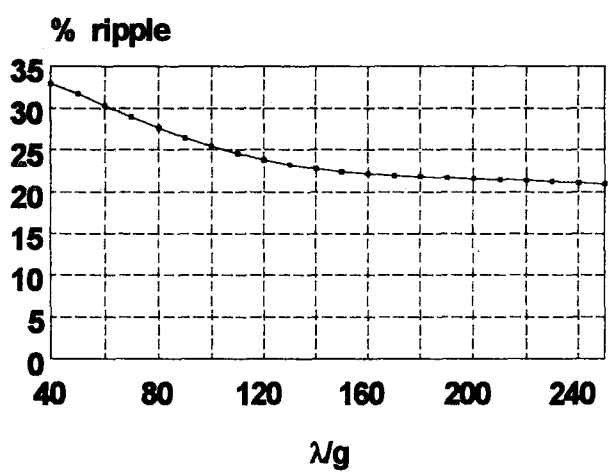
Figure B.1 Cont'd



(a) $t_s/\lambda=0.4$, $t_r/\lambda=0.5$

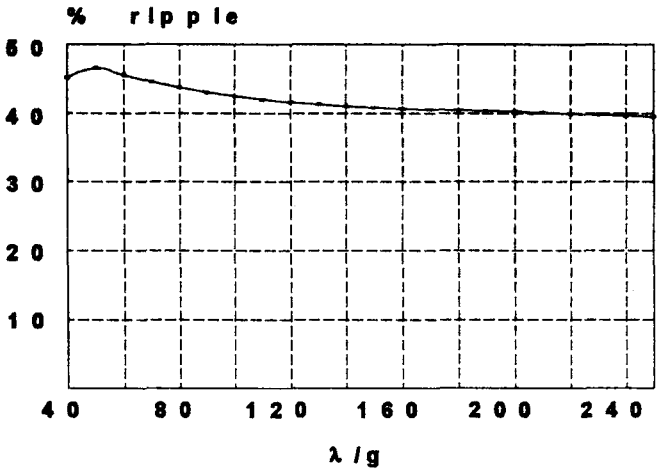


(b) $t_s/\lambda=0.3$, $t_r/\lambda=0.4$

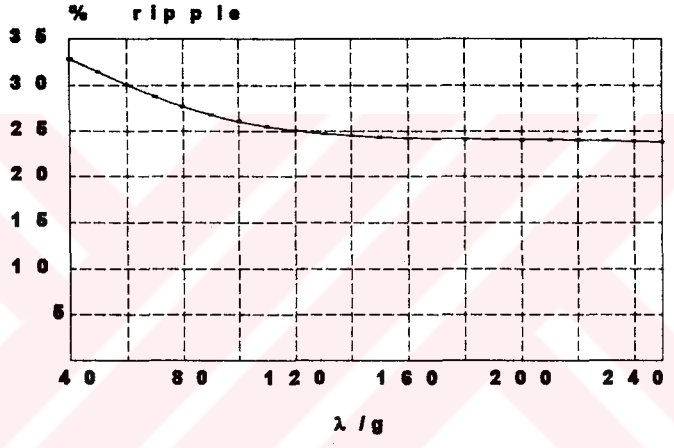


(c) $t_s/\lambda=0.3$, $t_r/\lambda=0.5$

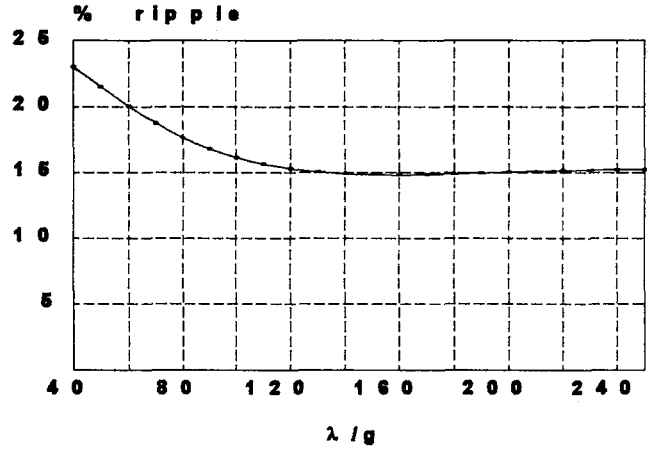
Figure B.2 Effect of λ/g on torque ripple for different t_s/λ, t_r/λ values at 40 kA



(d) t_o/λ=0.3, t_r/λ=0.3

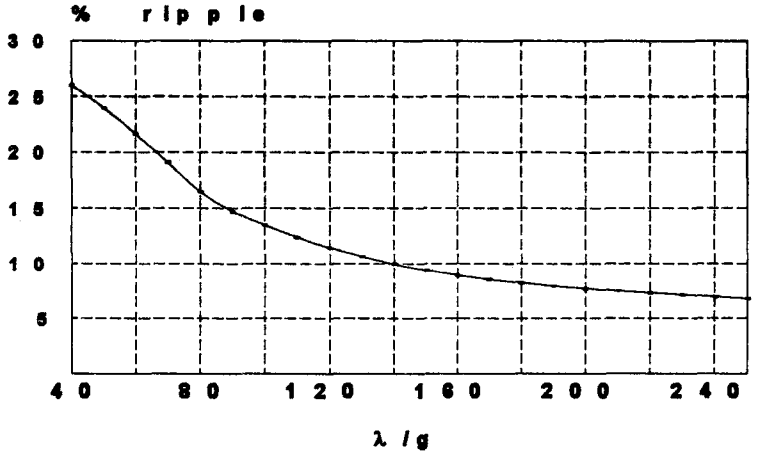


(e) t_o/λ=0.4, t_r/λ=0.4

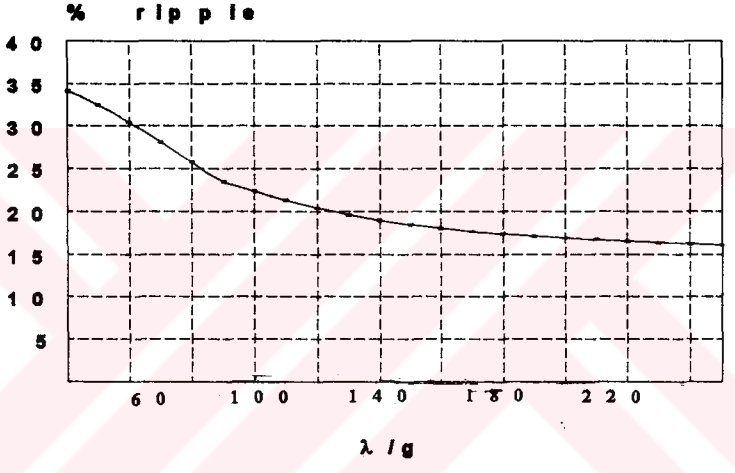


(f) t_o/λ=0.5, t_r/λ=0.5

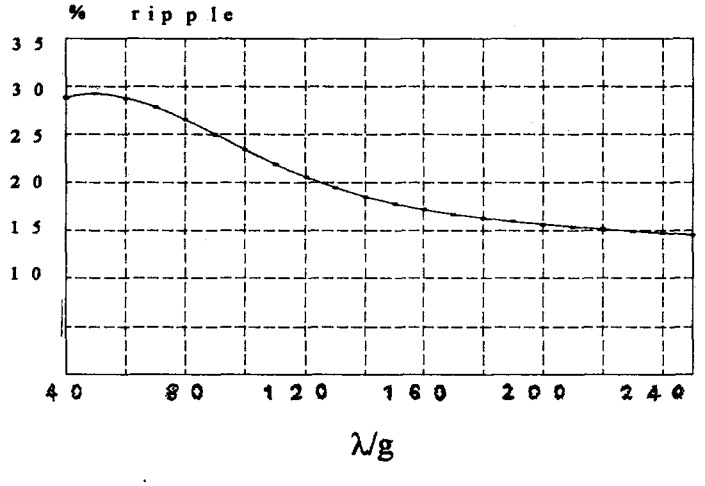
Figure B.2 Cont'd



(a) $t_s/\lambda=0.4$, $t_r/\lambda=0.5$

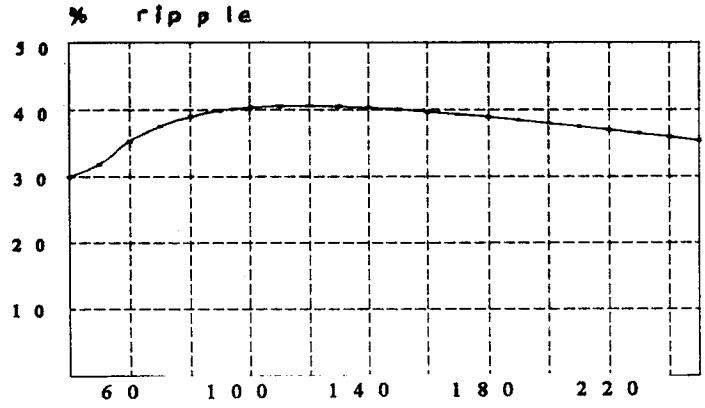


$t_s/\lambda=0.3$, $t_r/\lambda=0.4$



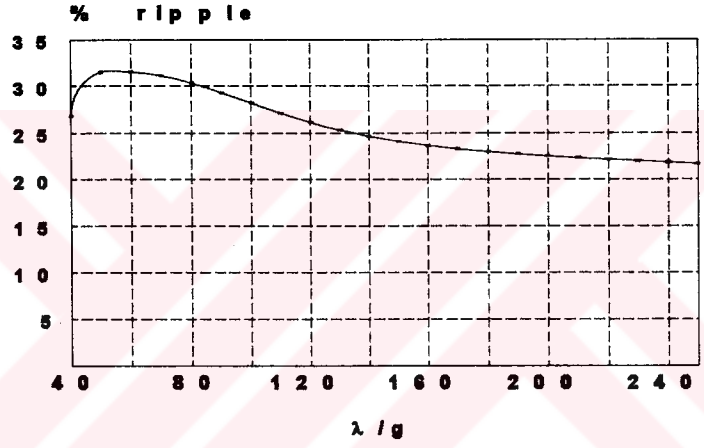
(c) $t_s/\lambda=0.3$, $t_r/\lambda=0.5$

Figure B.3 Effect of λ/g on torque ripple for different t_s/λ, t_r/λ values at 60 kA

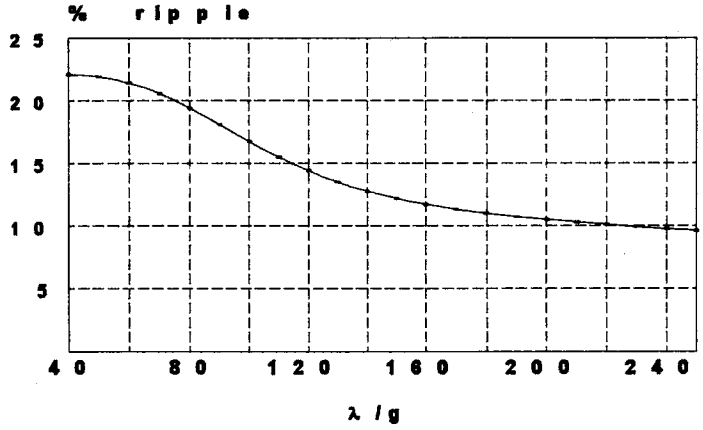


λ/g

(d) $t_s/\lambda=0.3$, $t_r/\lambda=0.3$

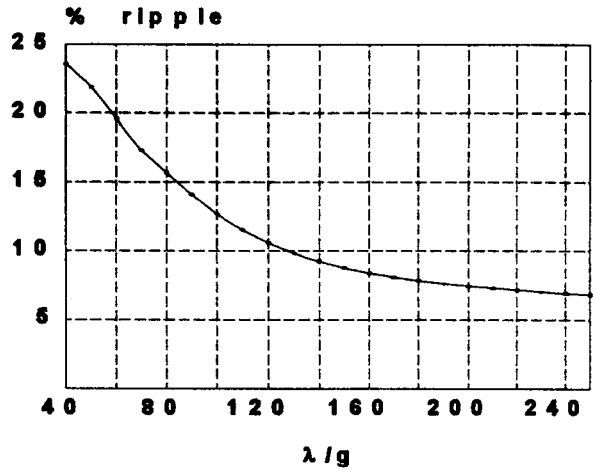


(e) $t_s/\lambda=0.4$, $t_r/\lambda=0.4$

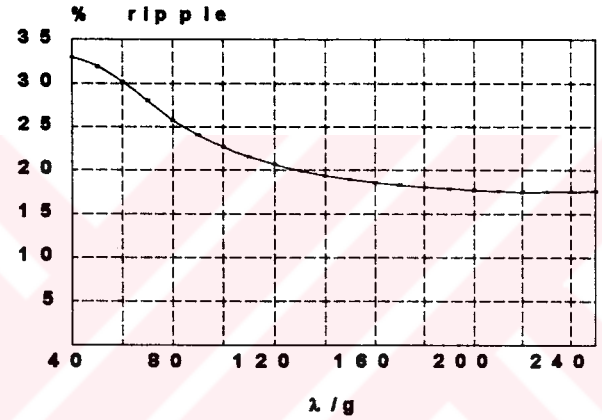


(f) $t_s/\lambda=0.5$, $t_r/\lambda=0.5$

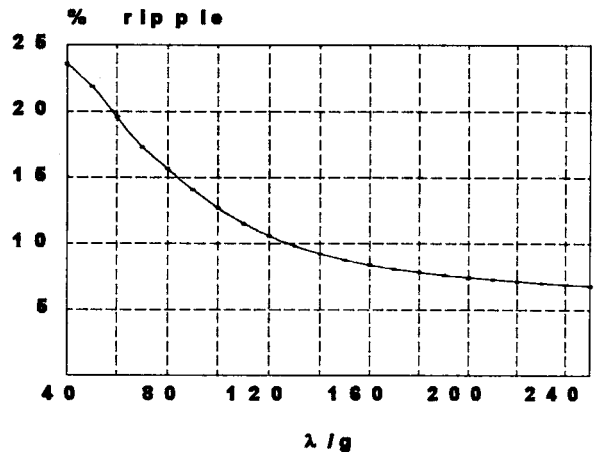
Figure B.3 Cont'd



(a) $t_s/\lambda=0.4$, $t_r/\lambda=0.5$

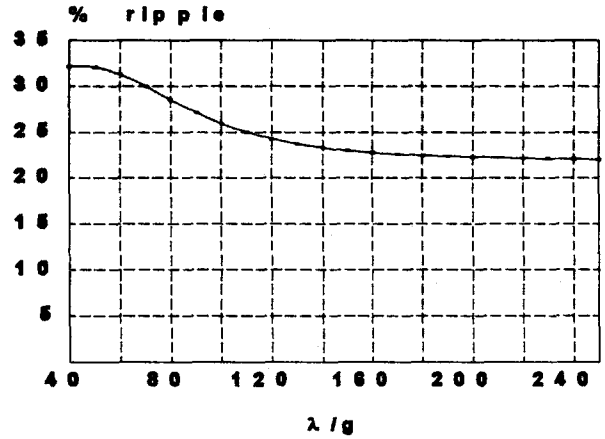


(b) $t_s/\lambda=0.3$, $t_r/\lambda=0.4$

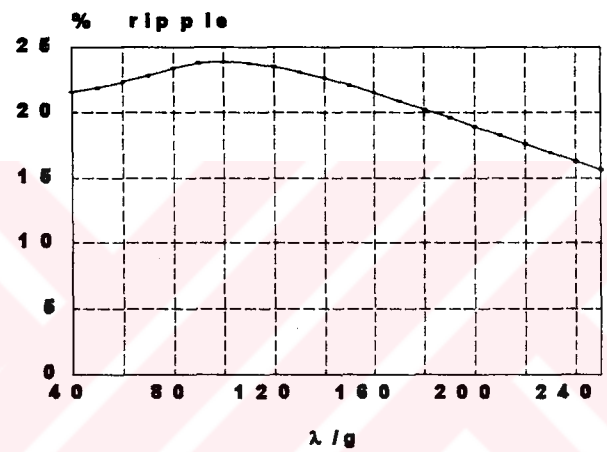


(c) $t_s/\lambda=0.3$, $t_r/\lambda=0.5$

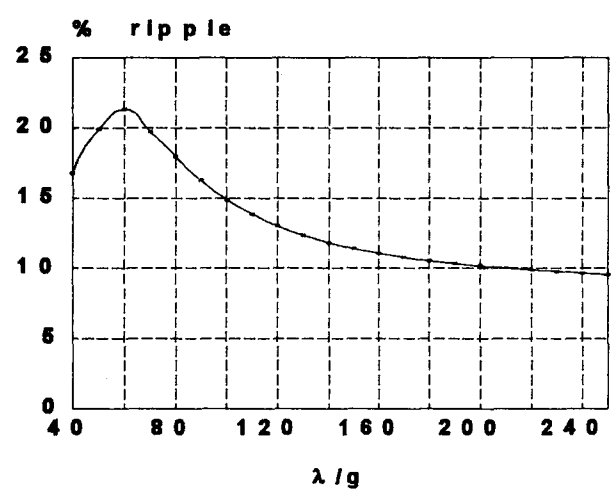
Figure B.4 Effect of λ/g on torque ripple for different t_s/λ, t_r/λ values at 70 kA



(d) $t_s/\lambda=0.3$, $t_r/\lambda=0.3$

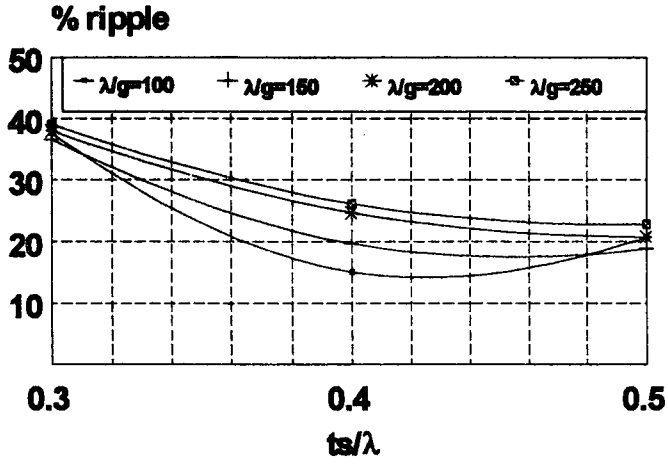


(e) $t_s/\lambda=0.4$, $t_r/\lambda=0.4$

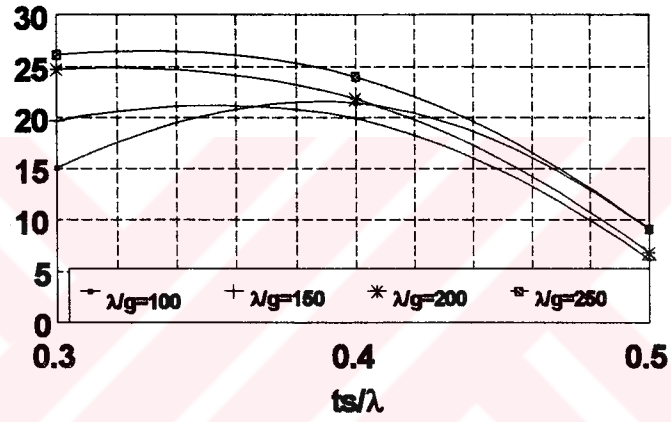


(f) $t_s/\lambda=0.5$, $t_r/\lambda=0.5$

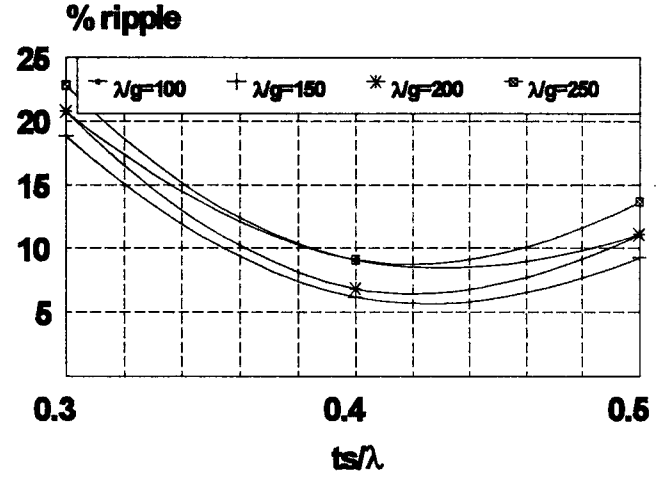
Figure B.4 Cont'd



(a) t_s/λ=0.3

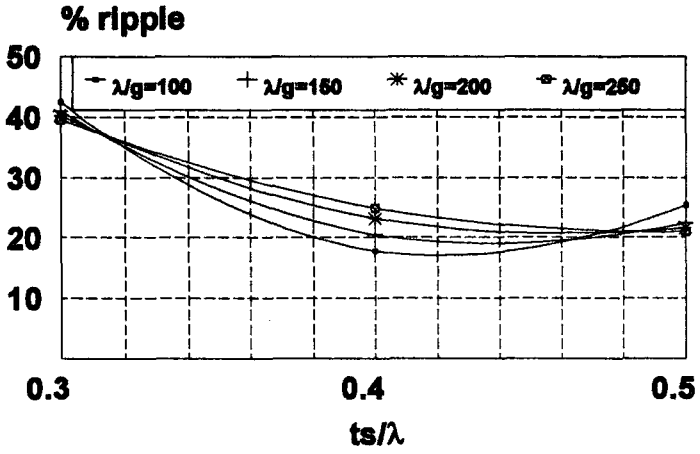


(b) t_s/λ=0.4

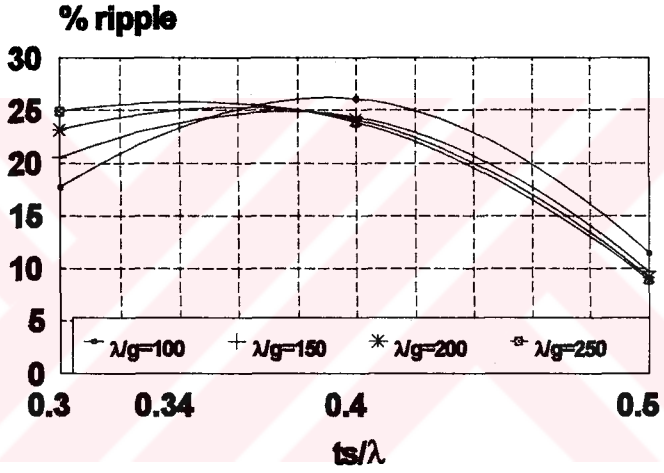


(c) t_s/λ=0.5

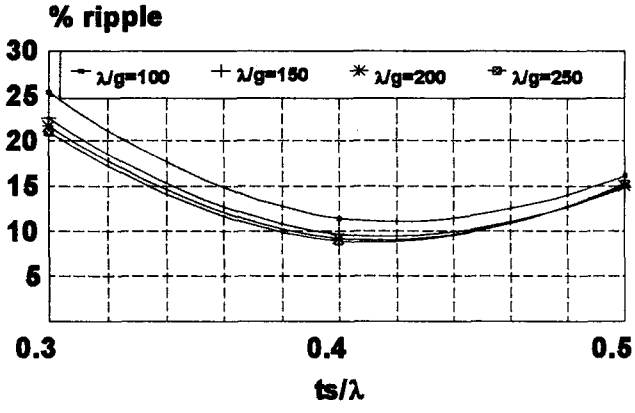
Figure B.5 Effect of t_s/λ on torque ripple for different λ/g values at 30 kA



(a) t_s/λ=0.3

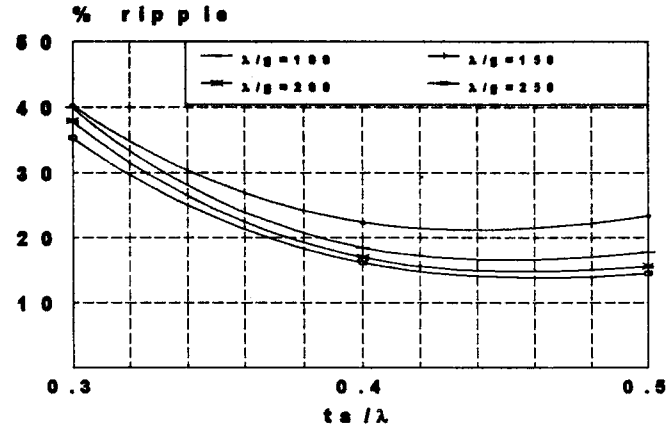


(b) t_s/λ=0.4

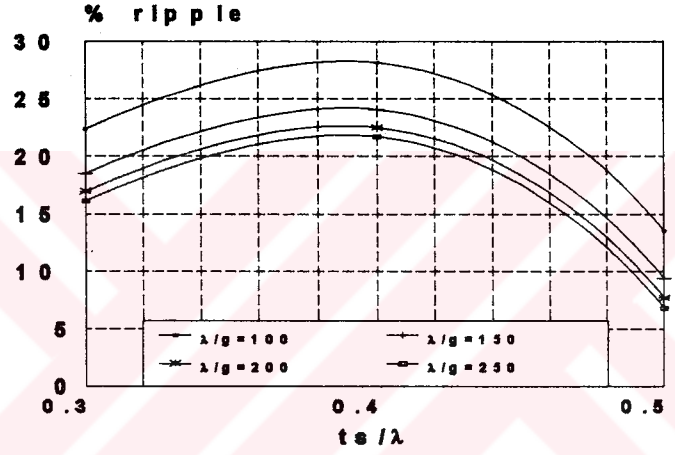


(c) t_s/λ=0.5

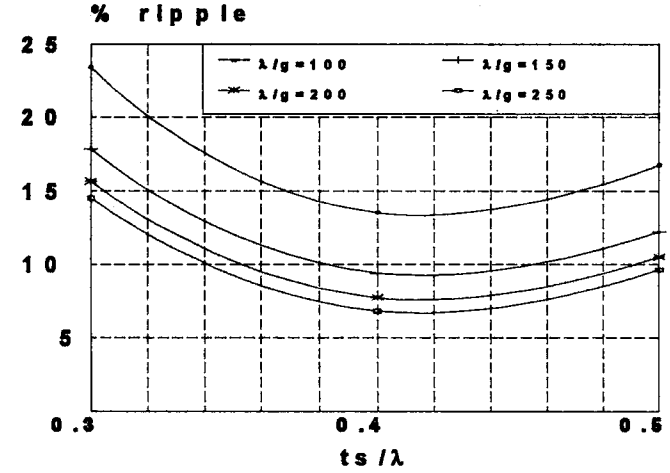
Figure B.6 Effect of t_s/λ on torque ripple for different λ/g values at 40 kA



(a) $t_r/\lambda=0.3$

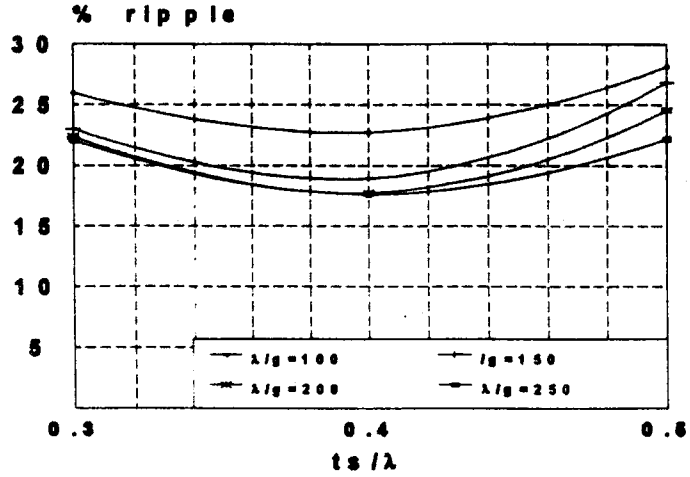


(b) $t_r/\lambda=0.4$

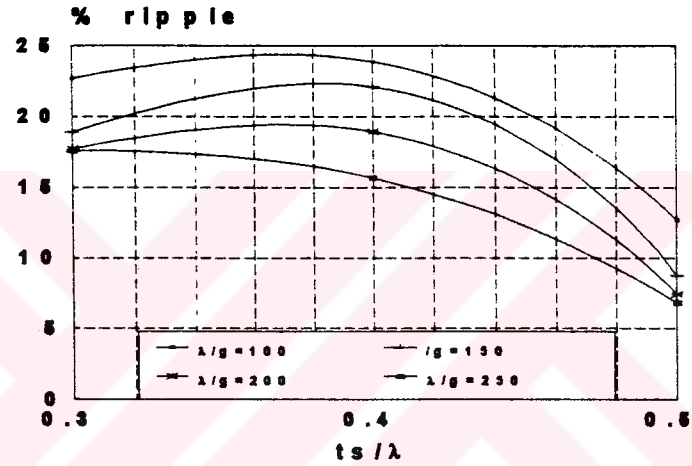


(c) $t_r/\lambda=0.5$

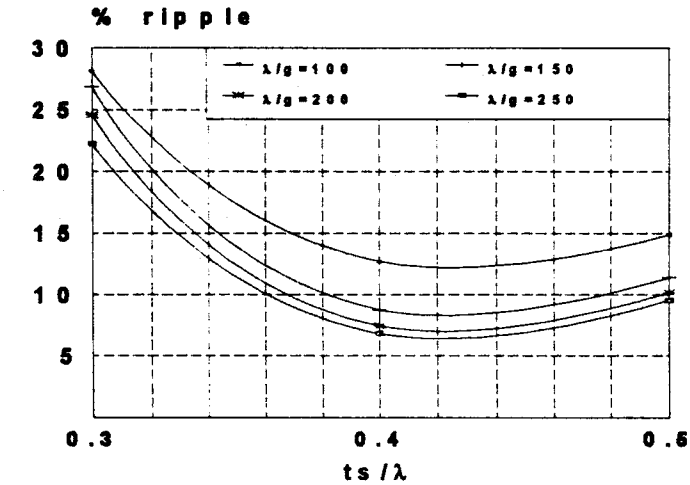
Figure B.7 Effect of t_s/λ on torque ripple for different λ/g values at 60 kA



(a) t_s/λ=0.3



(b) t_s/λ=0.4



(c) t_s/λ=0.5

Figure B.8 Effect of t_s/λ on torque ripple for different λ/g values at 70 kA