

**OPTIMIZATION OF DESIGN PARAMETERS FOR A VARIABLE
FREQUENCY 3-PHASE INDUCTION MOTOR**

By

VISHNUVARTHA S/O APPLANAIDU

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical and Electronics Engineering)

JUNE 2009

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

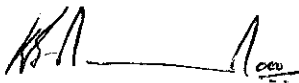
OPTIMIZATION OF DESIGN PARAMETERS FOR A VARIABLE FREQUENCY 3-PHASE
INDUCTION MOTOR

by

Vishnuvartha s/o Applanaidu

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by,



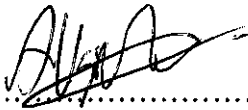
(A.P. Dr.K.S Rama Rao)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2009

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



.....
(VISHNUVARTHA A/L APPLANAIDU)

ABSTRACT

This report describes the method of optimizing a variable frequency 3-phase Induction Motor. The objective of this project is to come up with a final design of a 3-phase induction motor that is optimum in weight and cost and maximum in efficiency (weight and cost) without violating the constraints provided. The design of induction motor chosen for this purpose is a squirrel cage design motor as it is more robust and is widely used in the industry. C-language is used to describe the design of the machine. The machine design mainly involves non linear equations such as the magnetizing current flows and motor losses .The objective function achieved from the design are cost, efficiency and weight. The optimization technique used to optimize this objective functions is Genetic Algorithm (GA) which is a Non –Linear Programming technique. Twelve (12) design variables are identified and used in the design process. The objective of this project is to come up with the optimized variable which produces the highest motor efficiency, minimum weight and cost. The results of the optimized values will be compared with the values obtained before optimizing. The comparison is shown in scatter line graph. The final design obtained is the improved the improved version of the design parameters compared with the one used as the input to the optimizing program.

TABLE OF CONTENTS

ABSTRACT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	vii
LIST OF ABBREVIATIONS.....	viii
CHAPTER 1: INTRODUCTION.....	
1.1 Background of Study.....	1
1.2 Problem Statement.....	2
1.3 Project Objective.....	3
CHAPTER 2: LITERATURE REVIEW	
2.1 3-Phase Induction Motor.....	4
2.2 Wound Rotor Induction Motor.....	4
2.3 Squirrel Cage Induction Motor.....	5
2.4 Variable Frequency 3-Phase Induction Motor.....	6

2.5	Induction Motor Equivalent Circuit.....	9
2.6	Optimization Technique.....	11
2.7	Genetic Algorithm (GA).....	12
CHAPTER 3:	METHODOLOGY/PROJECT WORK.....	16
3.1	Project Flow.....	16
3.2	Software and Tools.....	21
CHAPTER 4:	RESULTS AND DISCUSSION.....	24
4.1	Results and Calculations.....	24
4.2	Discussion.....	33
CHAPTER 5:	CONCLUSION AND RECOMMENDATIONS.....	35
5.1	Relevancies to Objective.....	35
5.2	Recommendations.....	35
5.3	Conclusion.....	36
REFERENCES	37
APPENDICES		

LIST OF FIGURES

- Figure 2.1: Common Induction Motors^[3]
- Figure 2.2: Induction Motor equivalent Circuit at fundamental component and corresponding n^{th} harmonic component
- Figure 3.1: Flow chart of procedure identification of the project
- Figure 3.2: Example of small programs used.
- Figure 3.3: Executed results of design program
- Figure 3.4: Microsoft Visual C++ 6.0
- Figure 4.1: Cost Graph Vs Number of Generations(x100) for 60Hz machine
- Figure 4.2: Cost Graph Vs Number of Generations(x100) for 50Hz machine
- Figure 4.3: Weight Comparison Graph
- Figure 4.4: Cost Comparison graph
- Figure 4.5: Efficiency Comparison Graph

LIST OF TABLES

Table 3.1: List of Design Variables Used

Table 4.1: Starting Value for the Design variable and the Constraints Involved

Table 4.2: Optimized Design Parameters for a 60 Hz machine according to number of generations

Table 4.3: Optimized Design Parameters for a 50 Hz machine according to number of generations

Table 4.4: Optimized Design which is chosen after 600 generations

ABBREVIATIONS AND NOMENCLATURES

UTP Universiti Teknologi Petronas

GA Genetic Algorithm

Emf Electromagnetic Force

Hz Hertz

ACKNOWLEDGEMENT

First and foremost I would like to express my greatest gratitude to my project supervisor, Associate Professor Dr.K.S Rama Rao for his expertise, guidance, attention, support and advices regarding the project. It has been my greatest honor to be working under him during the entire project length.

I would also like to pass my special thank you to the entire Universiti Teknologi Petronas Electrical and Electronics Department for many constructive inputs and invaluable help.

A special thanks to the staff and management of Information Resource Center (IRC) of Universiti Teknologi Petronas (UTP) for your world class service when it comes to finding books and journals.

To my family all I can say is love and thank you to all of you. Without you enormous support, concern and care all my effort in completing this final year project would not be successful.

Last but not least, my appreciation goes out to the individuals and groups that have helped in any possible way to complete this project.

CHAPTER 1

INTRODUCTION

1.1 Background Of Study

The most common type of Alternating Current (AC) motor that is being used at present-day is induction motor. Induction motor is very much favored due to its rugged characteristics and requires very much less maintenance compared to other type of motors available. Common uses of induction motor in the industry application are lifts, cranes, pumps and lathes [2]. As the time goes on the design of induction motor has improved tremendously and the improvement of an 3-phase induction motor nowadays involves more on the optimal design parameters of the machine.

The main issue in the existing 3-phase induction motor design is efficiency. For induction motor operating at variable frequency losses in form of hysteresis and eddy losses are common in the stator and also in the rotor iron and additional losses in the winding copper. Apart from those mentioned losses, the temperature increase of the rotor due to current distribution in the rotor slots also contribute to inefficiency energy usage in the motor [5]. Variable speed drives induction motor which operates at variable frequency. Apart from energy efficiency the design in this report also aimed to reduce the cost of material used and perhaps to reduce the size of the machine. The existing conventional programs are not good enough for design calculations to calculate motors of variable speed motors this method is not practical and could not be used. This project is meant to design the induction

motor with the optimum parameters. This is method is favored due to the limitations of the existing conventional calculation.

1.2 PROBLEM STATEMENT

The efficiencies of motors and motor drive system used throughout the world have a large impact on the world .Study shows that energy consumed by electric motors accounts for nearly half of all electricity generated in the world today. This kind of impact means the inefficiency of motors result in large amount of energy being wasted [5] . Energy efficiency issues are debated worldwide these days and more and more countries are trying to save energy to avoid wastage. Along with sky rocketing fuel price in the world market, manufacturers are being pressured to design and build more energy efficient motors to save cost and energy. Before the 1970s the design improvement was focused on the material used to build the induction motor as energy was cheap. After the 1970s the fuel price started to climb and thus increasing the cost of machine operation [1]. This issue has driven the manufacturers of induction motors in the world to come up with more efficient motors which uses energy efficiently. Thus this project is to design high frequency Induction motor with optimum parameters which will be more energy efficient. However the word efficient in the present world does not refer to energy efficiency alone. As the price of raw material such as copper, iron and lead keep going up the race of efficient design takes a new twist for the good [4]. To combat ever expensive raw material engineers are required to come up with designs that are cheaper, uses less material and also lighter. Using lesser material will also affect the performance parameters of the machine such as the efficiency and power factor. It is important that a good design can produce an almost perfect compromise between factors such as cost, weight and efficiency. This is vital as different sectors of industry have deterrent preferences when it comes to design of Induction Motor.

1.3 PROJECT OBJECTIVES

The objective of this project is to:

- Design a variable frequency induction motor with the most optimal design to increase overall efficiency which is in terms of size, materials and cost of the machine.

- To use an Optimization method to produce optimum parameters for the design

- To come up with design parameters which are as optimum as possible to achieve high efficiency standard in the possible aspects mentioned.

CHAPTER 2

LITERATURE REVIEW

2.1 3-Phase Induction Motor

3- Phase Induction motor is a electrical machine which alternating current is fed to the stator directly while current is supplied to the rotor by induction phenomenon from the stator. The motor is excited by a balanced 3-phase source and will produce a magnetic field in the air gap which will rotate at a synchronous speed .The synchronous speed is determined by the number of poles and the frequency applied to the stator [1] .The rotor available at present days is of two types. One is the wound rotor and the other the squirrel cage based rotor.

2.2 Wound Rotor Induction Motor

Wound Rotor induction motor consist of a stator core with a three phase windings with slip rings, brushes and brush holders, and two end shields to house that support the rotor shaft [2] .

The stator contains a three phase windings placed in the slots of a laminated steel core. The winding consists of formed coils arranged and connected so that there are three single phase windings connected either in wyes or delta.

The rotor consists of a cylindrical core composed of steel laminations. Slots cut into the cylindrical core hold the formed coils of wire for the rotor windings[1].

2.3 Squirrel Cage Induction Motor

Most of the present day Induction motor is using the squirrel cage base design rotor. The design of this type of induction motor is basically small in size for a given horsepower rating when compared with other types of motors. It has a very good speed regulation under varying load conditions. This type of induction motor is widely used because of its rugged construction and reliable operation [2]. The motor frame usually is made of cast steel. The stator core is pressed directly into the frame. The two end shields housing the bearings are bolted to the cast steel frame. Bearings which support the rotor shaft are either sleeve bearings or ball bearings [5].

The stator for this type of machine contains three-phase windings mounted in the slots of a laminated steel core. In the induction machine the winding consist of formed coils of wire connected so that there are 3-single phase windings spaced 120 electrical degree apart [2]. These three windings are connected internally. Three or nine leads from the three-phase stator windings are brought out to a terminal box mounted on the frame of the motor for single or dual voltage connections.

The rotor consists of steel punching or laminations arranged in a cylindrical core. Copper or aluminum bars are mounted near the surface of the motor. The bars are brazed or welded to two copper end rings in one piece from aluminum.

Squirrel- cage motor has a few advantages compared to a phase-wound induction motor. The former has a slightly higher efficiency compared to the wound rotor family motors.[2] Cage design also is relatively cheaper and easy to construct.

Squirrel cage machine is preferred nowadays due to its better space factor for rotor slots which results less copper loss from the machine. The clearest difference of this design compared to the wound rotor apart from the cage structure is the bare end rings which provide a large space for insertion of fans for cooling down purpose.

2.4 Variable Frequency 3-Phase Induction Motor

The design of Induction Motor has improved in many ways since its invention in 1880s. However the improvement in the design did not lead to improvement in energy usage efficiency [5]. The primary objective of the design improvement at this time was aimed to reduce the material cost to build the machine. The real effort to improve energy efficiency of the machine came after the sky-rocketing fuel in the 1970s which caused electricity to be expensive. The two main parts in an induction motor is the stator and the rotor. There are two types of rotor that can be placed inside the stator. The first type is the squirrel cage rotor which also known as cage rotors while the second type of rotor is called wound rotor. The word induction is used because the rotor voltage is induced in the rotor windings and not physically connected using wires. The main aspect of an induction machine is that no dc field current is needed to operate the machine. The variable frequency in this project means the machine operates at different frequencies [5].

Motor efficiency is basically the measure of the ability of an electric motor to convert electrical energy to mechanical energy [2]. As an example when a certain amount of electrical power is supplied to an induction motor the output energy is taken out of the rotating shaft. Thus the only power absorbed by the motor is the losses which takes place while making the conversion from electrical to mechanical energy [10].

Efficiency = [mechanical energy out /electrical energy in] X 100%

Mechanical energy out = Electrical energy In – Total motor losses

Electrical energy in= Mechanical energy out + Motor losses

So to reduce the power consumption and motor losses are to be reduced in order to increase the motors efficiency.

There are few types of losses that can occur in motor during its operation. The first is Magnetic core losses. This type of losses consists of the eddy current and hysteresis losses which also included surface losses, in the magnetic structure of the motor.

Friction and windage losses are caused by the friction in the bearings of the motor and also the loss which occurs on the other moving parts of a motor. This type of losses depend on a few aspects such the bearing size, speed of the motor and also the lubrication of the motor. Most of the windage losses are usually associated with the ventilation fans and the amount of ventilation required to remove heat which is generated by other losses in the other part of the motor such as $I^2 R$ losses, magnetic core losses and also stray load losses.

Stray load losses are basically the residual losses in the motor which is difficult to determine by direct measurement or any kind of calculation. These losses are mostly load related and most of the time assumed to vary as square of the output torque. Some of the aspects that mainly influence this type of losses are the stator

windings design, air gap ratio to the rotor slot openings, the air gap flux density and also the condition of the rotor air gap surface.

In the topic of motor losses we always find that various losses are very much dependent. In the present day the motor design is aimed to balance out all the type of losses to obtain high efficiency but yet it should meet the other performance criteria such as locked rotor torque, locked rotor ampere, breakdown torque and also the power factor.

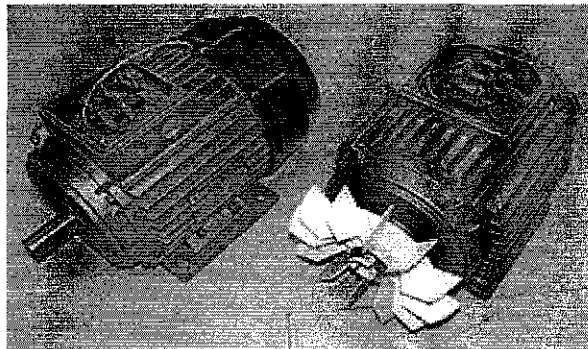


Figure 2.1: Common Induction Motors^[12]

Figure 1 above shows common types of induction motor that are available in the market today.

2.5 Induction Motor Equivalent Circuit

The movement of flux and current can be transformed into a steady state equivalent circuit for an induction motor. For this part of literature review it is assumed that the 3-phase machine model used is Y-connected which in bring to the currents are line current and the voltage values are line-to-neutral values.

Similar to a transformer an induction motor can be transformed and represented by an equivalent circuit as shown in Figure 2 below.

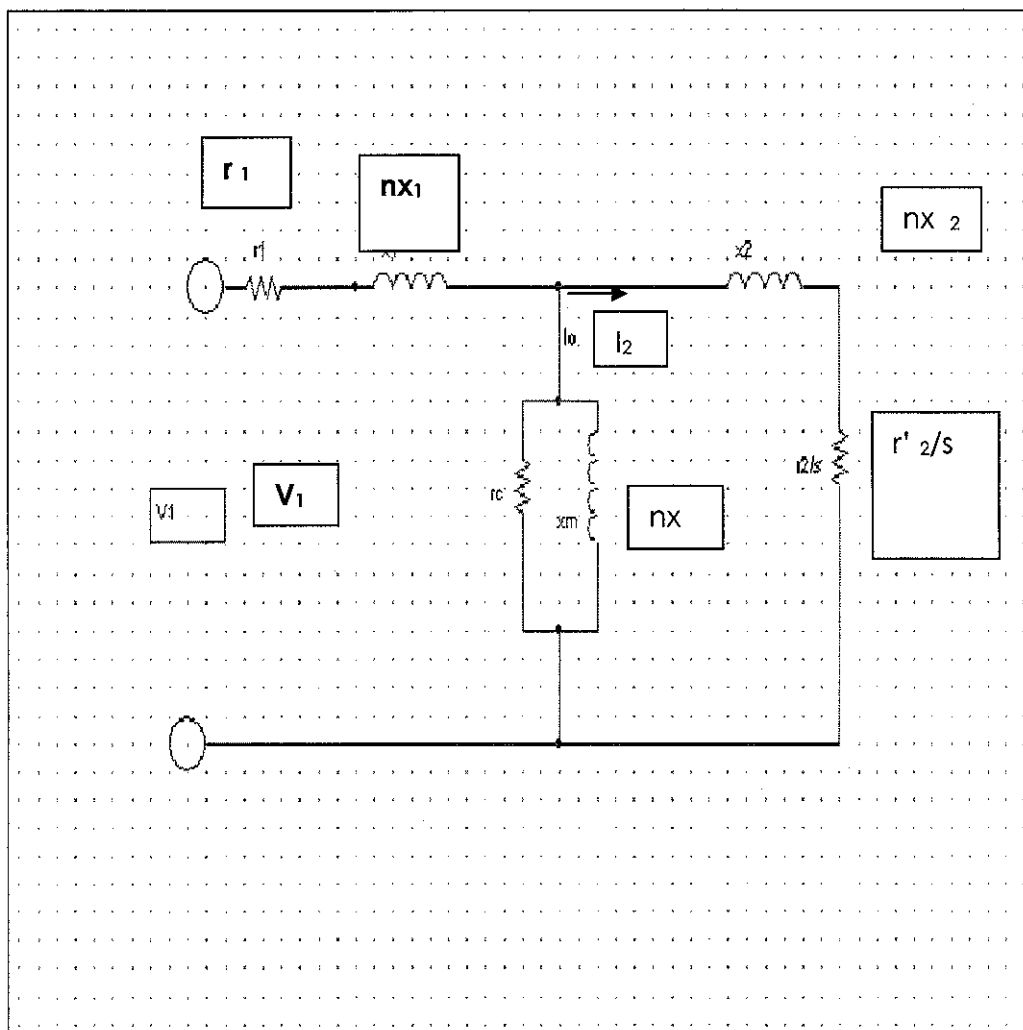


Figure 2.2: Induction Motor equivalent Circuit at fundamental component and corresponding n^{th} harmonic component

This equivalent circuit gives the ability to evaluate the induction motor performance at steady-state conditions. The stator resistor r_1 and leakage reactance x_2 are practically separated from the magnetizing branch and it is assumed that loss currents are flowing through the magnetizing branch of r_c and x_m .

For the stator part of the machine, there is this rotating air-gap flux which produces a balanced 3-phase counter emf in the stator. However the stator terminal voltage has a different value compared to the counter emf generated due to voltage drop in the stator leakage impedance. The equation to relate the phasor relationship can be represented by

$$V_1 = E_1 + I_1 (r_1 + j x_1)$$

Where V_1 = Terminal voltage of the stator

E_1 = Emf produced by the air gap flux

I_1 = Stator current

r_1 = Effective resistance in the stator

x_1 = Stator leakage reactance

2.5 Optimization Technique

Optimization is often referred to as mathematical programming and at present days C-language is used heavily to develop programs to optimize objective functions that represent design of electrical machine [3]. Optimization technique is often related to Non-Linear Programming (NLP). The reason this happens is because most design functions involved especially in electrical machines studies are non-linear equations[3].

Using more than one objective function in a optimizing program is called multi objective optimization which in designing and optimizing an induction motor is inevitable. It can be said that using programming based optimization ,designs of electrical machines can be optimized much easier compared to manual calculation which can only house single objective functions [3].

Optimal design theoretically has the needs to formulate the design problem as Nonlinear Programming (NLP) problem and solving it using a mathematical programming technique, Genetic Algorithm (GA), Artificial Neural Network (ANN) and other available methods. Using the non-linear programming requires objective function, constraints and design variables [10]. The objective function for a synchronous generator or induction motor can be the active material cost, volume or weight. Logically the performance specifications of the generator or motor constitute the constraints. A selected number of variables based on their affect on the objective and constraint functions are considered as design variables. The total number of variables for a motor or generator is normally large in number. However it is not practicable to choose all these variables as design variables as it is not possible.

2.6 Genetic Algorithm (GA)

Genetic Algorithm or commonly known as GA is a evolutionary programming method that optimizes the design of electrical machine which in this case a 3-phase induction motor. This method is a soft approach that uses surface search technique to optimize multi-objective design equations. With a stopping limit inserted the solution is obtained when the limit is satisfied on both side [4]. The literature of Genetic Algorithm is heavily influenced on Darwin's Theory of species evolution which explains the breeding of living organism is a mathematical approach [4].

The process starts with initialization of a few set of solution called population. The next step is initialization of number generation which is a repetition of random generation of sets of chromosomes.

To perform and use genetic algorithm on design equations based program there are few pre-determined steps that need to be followed such as:

Step 1 : Selection of algorithm breed.

- Defining Design equations.
- Creating Objective Function.
- Initializing of optimization calculations.

Step 2: Generation of population

- A new population is selected to start a new breed.
- Generation of population is done at random.
- Objective function is used.

Step 3 : Evaluation

- The population is evaluated using the objective function inserted.

Step 4 : Limit and convergence test

- If the minimum limit is found then program will stop to produce the satisfied values.

Step 5 : New Generation

- Generate more variation of generations through selection, crossover and mutation to produce more viable results ^[4].

Step 6 : Next population test

- Population selection can be increased or decreased according to the necessity of the calculation.

Step 7 : Termination

- Termination process is a repeated step until a suitable value of optimized design value is obtained which satisfies the limit and design construction limit.

Genetic Algorithm (GA) requires a few parameter so that the optimization can take place and good results can be obtained if the following operators are done well[3]:

- Selection - Individual strings selected according to its fitness and the probability selection can be

defined as
$$P_j = \frac{F(x_j)}{\sum_i F(x_i)} \quad [13]$$

- Crossover –Crossover is a process there a point of crossing is selected between the first and last bits of the chromosomes [13].The binary bit on the utmost roght of the crossover point takes the place of the beginning of the second offspring code. This process continues until a probability of (P) that represents the number of individuals involved in the crossover process is achieved. The example is shown as :

$$C_1 = 0010010|101 \quad \text{Offspring}_1 = 0010010|100$$

$$C_2 = 0101011|100 \quad \text{Offspring}_2 = 0101011|101$$

[13]

- Mutation-This process is related to changing the binary numbers during the copying process of the chromosome [13].In other words a binary number is changes to its opposite number randomly to create a search outside the search parameter region.

Design function or objective function in this study is represented by F(x) and in the context of electrical machines it can represent cost of the machine, weight or even the efficiency of the machine. The drawback of using Genetic algorithm to optimize an electrical machines is that they are different margin of constraints for

differ rent part of the machine. As an example a rotor design might have a completely different dimension need compared to the stator dimension of the same machine [4].

Together with the objective function a penalty function is needed which is often represented by $g(x)$.The purpose of this function is to remove constraints periodically and when needed only. It is important that a penalty function is inserted in order to produce a optimized design which is both viable and satisfies the upper and lower limits of a design function [13].

The programming of a Genetic Algorithm based optimization technique can be written in C- language or any other higher level programming language such as C++ and Delphi [4] .

CHAPTER 3

METHODOLOGY/PROJECT WORK

3.1 Project Flow

Figure 3.1 shows the methodology that is used to approach this project systematically. The procedures are planned in guiding to completion of this project. The steps are shown in Figure 3.1. The time line and brief description of each stage is shown in the Gantt chart attached in Appendix B.

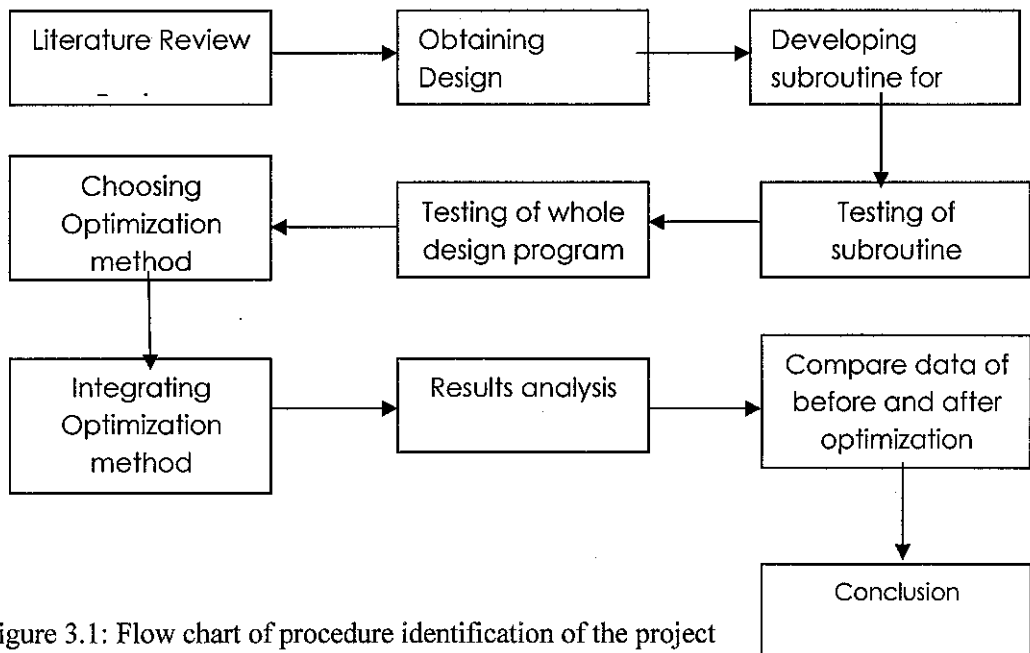


Figure 3.1: Flow chart of procedure identification of the project

The first phase of the project involves obtaining the design variables which are to be optimized. The design variables and constraints on the design variables for the project are as follows:

<u>Design Variables</u>	<u>Units</u>	<u>Constraints</u>
X₁ = Diameter of stator bore	m	0.1 - 0.2
X₂ = Length of stator core	m	0.09 - 0.15
X₃ = Width of stator slot	m	0.006 - 0.01
X₄ = Depth of stator slot	m	0.02 - 0.04
X₅ = Depth of stator core	m	0.02 - 0.03
X₆ = Length of the air gap	m	0.003 - 0.0005
X₇ = Width of the rotor slot	m	0.006 - 0.01
X₈ = Depth of the rotor Slot	m	0.01 - 0.03
X₉ =Depth of rotor core	m	0.03 - 0.05
X₁₀ = Average flux density in the air gap	T	0.5 - 0.7
X₁₁ = Area of cross section of end ring	mm²	120.0 - 140.0
X₁₂= Current Density in Stator	A/mm2	3-5

Table 3.1: List of Design Variables Used

The design variables are obtained from books and journals and so is their starting value as displayed in the Table 3.1. The constraints which are listed in the table are used in the optimization part of the project. Using the design variables necessary design equations are developed and translated into C-language based program.

As mentioned before the program is developed part using function programs by part to minimize error. This is important because running a long program all together will make trouble shooting extremely difficult. The example of a small program for the design of the stator winding is as shown in Figure 3.2.

```

//stator winding design
stwdg(nzs,x3,x4,h3,h4,cph,sigma,sdi,swi,aos,bos,acs,ncw,ncd,c
dty);

cout <<"cdty="<<cdty<<"\n";

nzs      =      ncw*ncd;

ntph    =      npl/2*nspp*nzs;

akp      =      1.0;
//pitch factor - full pitch

akd      =
factor    sin(nspp*pi/gsl/2.0)/(nspp*sin(pi/gsl/2.0));      //distribution

akw1    =      akp*akd;
//winding factor

tos      =      pi*x1/ns1;
// stator slot pitch

wtf      =      tos - x3;

wth      =      pi*(x1+x4)/ns1-x3;

wtb      =      pi*(x1+2.0*x4)/ns1 - x3;

ap1      =      phi*npl/ns1/eli;

bfta    =      ap1/wtf;

```

Figure 3.2: Example of small programs used.

The next phase of the project is to combine the entire design program and executing it. The entire program when run produces the results for the equations which is keyed in based on the design variables used. The result of the executed program is as shown in Figure 3.3.

```

"C:\Documents and Settings\wshnu\Desktop\Debug\IM - eqckt-main_23 Mar 2009.exe"
cib= 249.093, cer= 713.598, cdrb= 5.6102, aer= 120
der= 23, wer= 5.21739, cder= 5.94665, cior= 6.16023, xm= 61.8406
wbb= 0.0606513, clre= 0.01, albr= 0.0711061, clpr= 0.01
rbr= 3.36313e-005, dier= 0.11439, rer= 6.28892e-005, rbr1= 2.8677e-005, rrp= 3.6
0941
wt1=0.0105, wt2=0.00605, d2 =0.14889
bcsa=1.48151, tcas=907.545, alcs =0.0645335, ttra =266.845
alcr =0.0329213, atga =512.755, bcr1 =1.15743, bcsa =1.48151
btca =0.893609, bga =0.8064, btra =1.15106, bcra =14.6437
fesi =1.0049, sesi =0.0183559, aloxi =0.998601
xsi =10.5676, xoi =3.36618, xzi =0.63622
flpt=1, hrfq=50, cpxi=14.5552
fundamental phi=0.00296881
htsi= 0.586195, athsi= 83.7422, pmsi= 0.007
edtsi= 0.0789728, edtsi= 0.4879, gsth= 0.127679
, frwd= 1563.09, rymn= 0.0070996, cjmn= -0.0161706
gscw= 1.37027, gend= 0.750358, gsth= 0.127679, gscr= 7.49844
grth= 0.484538, gcp= 43.2687, gbr= 0.750358, gir= 8.49766, grcr=
aloo= 0.0935619, alov= 0.247124, are1= 0.194529, are2= 0.0518363
pps= 7.85398, area= 0.399148, bedi= 11.361, belt= 31.2428
vber= 5.94861, befr= 1544.93, wind= 18.1572, frwd= 1563.09
csfh= 5.72381, crfh= 0.255054, rmst= 5.72381, rmsh= 0
pscu= 142.724, prcu= 0.704422, pts= 0.566872, pcs= 91.5059, ptr= 3.56938
pcr= 461.408, pss= 1.34427e-008, psr= 1.38933e-007, pszz= 4.85408, psse= 0.01221
15
prre= 0.0122115, pstp= 0.00214064, psbn= 134.524, pskn= 0.00190281
stls= 234.811, tems= 23.5312, rtls= 605.074, arer= 0.0205188
temp= 475.337, shaf= 3.30889, ttls= 2402.98, sln= 0.000133066, pwin=
eff= 0.0340364, pf= 0.362178, pmx= 894268, hpmx= 1198.75
ciky= 1, cckg= 2, chkg= 2, pfsc= 0.250533
cost = 96.5357
Press any key to continue

```

Figure 3.3: Executed results of design program

However the starting value used for the testing purposes are dummy values. The correct values are used in the optimization program and will be explained in the later part of the report. From the design program some the parameter which are calculated are cost, weight of the machine efficiency, magnetizing current and

magnetizing reactance etc. For this project the weight and efficiency are used as results and compared.

The next stage of the project is choosing the suitable optimization technique. Since the induction motor design equation involves many non-linear equations a suitable Non-linear programming optimization technique is chosen. The method chosen in this project report is Genetic Algorithm or commonly known as (GA).

The justification of the suitability of using this technique is the size of the design program. Genetic algorithm is relatively simple to use compared to other optimization technique such as Scatter Search (SS) .Unlike other technique GA does not require the design equation to be transformed to its derivative function. Changing long design equations with many declarations in the C-program will take extremely a lot of time and also make the complete calculation program prone to errors.

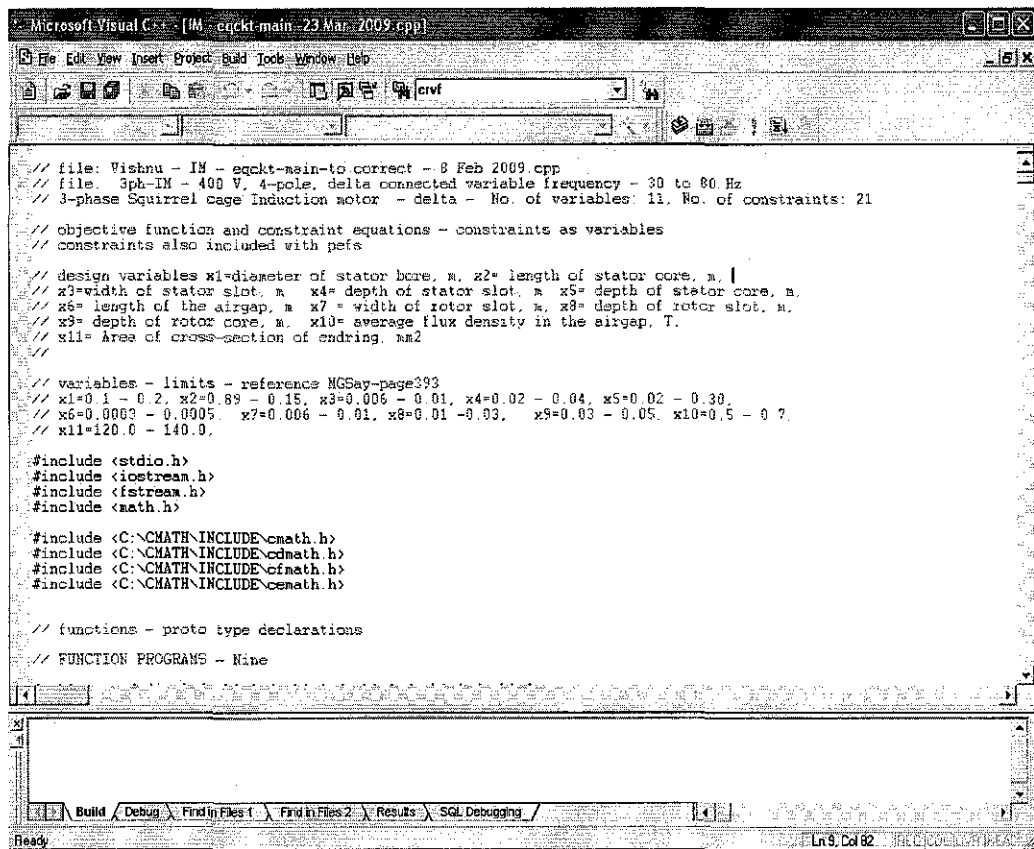
The results obtained after the optimization are then compared with the performance parameters obtained before the optimization is done. Comparison graph is drawn for efficiency and cost parameter. However it is important to always check the data of the design variables obtained in order to ensure the optimized design variables obtained are within the specified limit.

3.2 Software and Tools

The software used for the programming purpose of this project is :

- MICROSOFT VISUAL C++ 6.0 Edition.

This software has the capability to compile C language written programs and also to link required extension files into the main program. This software is relatively simple to use compared to other compiler program such as Visual Studio and Borland C++. The main view of the software is as shown in Figure 3.4.



```
// file: Vishnu - IM - eqckt-main-to correct - 8 Feb 2009.cpp
// file: 3ph-IM - 400 V, 4-pole, delta connected variable frequency - 30 to 80 Hz
// 3-phase Squirrel cage induction motor - delta - No. of variables: 11, No. of constraints: 21
// objective function and constraint equations - constraints as variables
// constraints also included with pefs

// design variables x1=diameter of stator bore, m, x2= length of stator core, m,
// x3=width of stator slot, m, x4= depth of stator slot, m, x5= depth of stator core, m,
// x6= length of the airgap, m, x7 = width of rotor slot, m, x8= depth of rotor slot, m,
// x9= depth of rotor core, m, x10= average flux density in the airgap, T,
// x11= Area of cross-section of endring, m2

// variables - limits - reference NGSay-page393
// x1=0.1 - 0.2, x2=0.89 - 0.15, x3=0.006 - 0.01, x4=0.02 - 0.04, x5=0.02 - 0.30,
// x6=0.0002 - 0.0005, x7=0.006 - 0.01, x8=0.01 -0.03, x9=0.03 - 0.05, x10=0.5 - 0.7,
// x11=120.0 - 140.0.

#include <stdio.h>
#include <iostream.h>
#include <fstream.h>
#include <math.h>

#include <C:\MATH-INCLUDE\cmath.h>
#include <C:\MATH-INCLUDE\cdmath.h>
#include <C:\MATH-INCLUDE\cfmath.h>
#include <C:\MATH-INCLUDE\cenmath.h>

// functions - proto type declarations
// FUNCTION PROGRAMS - Nine
```

Figure 3.4 : Microsoft Visual C++ 6.0

One of the drawbacks of using a high level programming language to write mathematical equations is the ability of the program to understand equations containing complex number. To solve this problem a separated include file is created to give the compiler the knowledge to understand complex number calculation. The performance calculation of the equivalent circuit needs complex number.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Calculations and Results

The data used for the starting value for the design program which is integrated into the Genetic Algorithm (GA) optimizing technique are as follows.

Motor Type	:	3- Phase Squirrel Cage Induction Motor
Frequency	:	Varied between 40 and 60 Hz
Supply	:	400 V
Horse power(h.p)	:	5
Slip	:	4.2 %
Connection	:	Star-Delta
Poles	:	4
Number of slots/pole/phase	:	3
p.f	:	0.84

Design Variables	Units	Constraints	Design Values
X₁ = Diameter of stator bore	m	0.1 - 0.2	0.15
X₂ = Length of stator core	m	0.09 - 0.15	0.09
X₃ = Width of stator slot	m	0.006 - 0.01	0.006
X₄ = Depth of stator slot	m	0.02 - 0.04	0.024
X₅ = Depth of stator core	m	0.02 - 0.03	0.021
X₆ = Length of the air gap	m	0.003 - 0.0005	0.00045
X₇ = Width of the rotor slot	m	0.006 - 0.01	0.0065
X₈ = Depth of the rotor Slot	m	0.01 - 0.03	0.0105
X₉ =Depth of rotor core	m	0.03 - 0.05	0.03
X₁₀ = Average flux density in the air gap	T	0.5 - 0.7	0.5
X₁₁ = Area of cross section of end ring	mm²	120.0 - 140.0	120
X₁₂ = Current Density in Stator	A/mm2	3.0-4.5	

Table 4.1: Starting Value for the Design variable and the Constraints Involved

The Genetic Algorithm program is run using the values in Figure 9. The number of generation is varied from 100-600. The results after optimization are shown in Figure 10 and 11 respectively for a 50 and 60 Hz machine. The C-program used to achieve the results in this section is included in APPENDIX 2.

The other GA parameters are fixed as

- Population size = 100
- Probability of Mutation = 0.044
- Probability of Crossover = 0.8867

The cost is represented in the units as the active material cost of lamination and winding copper are considered as in the ratio of 1:2 units.

Design Variables	Units	Number of Generations					
		100	200	300	400	500	600
X ₁ = Diameter of stator bore	m	0.106482	0.105585	0.112911	0.105295	0.106268	0.104137
X ₂ = Length of stator core	m	0.142589	0.143439	0.13402	0.143793	0.142519	0.145325
X ₃ = Width of stator slot	m	0.009916	0.009985	0.009973	0.00999286	0.00999335	0.00999347
X ₄ = Depth of stator slot	m	0.0202	0.020566	0.020524	0.0201389	0.020166	0.0200211
X ₅ = Depth of stator core	m	0.020207	0.020004	0.020005	0.0200162	0.0200206	0.020025
X ₆ = Length of the air gap	m	0.000492	0.000426	0.000496	0.000495901	0.000484515	0.000498151
X ₇ = Width of the rotor slot	m	0.006059	0.006056	0.006047	0.00602484	0.00600153	0.00600708
X ₈ = Depth of the rotor Slot	m	0.010078	0.010046	0.010005	0.0100061	0.0100003	0.0100027
X ₉ =Depth of rotor core	m	0.030435	0.030031	0.030109	0.0300565	0.0300226	0.0300174
X ₁₀ = Average flux density in the air gap	T	0.698163	0.699554	0.699969	0.699985	0.699719	0.69989
X ₁₁ = Area of cross section of end ring	mm ²	130.308	120.591	121.303	120.004	121.376	120.015
X ₁₂ = Current Density in Stator	A/mm ²	4.22181	4.30861	3.5648	3.98093	4.27844	4.07993
Weight	Kg	31.13512	30.81608	31.23851	30.74945	30.82599	30.66175
Cost	units	50.3974	50.05385	50.06567	49.96511	49.97877	49.93099
Total Losses	Watt	3951.534	3953.654	4003.104	3738.735	3767.33	3674.544
Efficiency		0.805196	0.805288	0.804908	0.8100432	0.8095888	0.8113441
Power Factor		0.758283	0.764557	0.748886	0.7599582	0.7604835	0.7615377

Table 4.2: Optimized Design Parameters for a 60 Hz machine according to number of generations

Design Variables	Units	100	200	300	400	500	600
X ₁ = Diameter of stator bore	m	0.106452	0.106259	0.103136	0.103351	0.101588	0.10645
X ₂ = Length of stator core	m	0.143882	0.142678	0.146828	0.14655	0.149076	0.142173
X ₃ = Width of stator slot	m	0.00997076	0.0099856	0.00999512	0.00999139	0.00999463	0.00999719
X ₄ = Depth of stator slot	m	0.0200479	0.0200082	0.0201279	0.0203473	0.0200845	0.0202905
X ₅ = Depth of stator core	m	0.0200371	0.0200562	0.0200265	0.0200005	0.0200009	0.0200125
X ₆ = Length of the air gap	m	0.000481396	0.000458843	0.000475711	0.000498376	0.000492618	0.00049953
X ₇ = Width of the rotor slot	m	0.00607282	0.00602258	0.00604486	0.00600006	0.00602661	0.00600513
X ₈ = Depth of the rotor Slot	m	0.0101038	0.0101013	0.0100079	0.0100046	0.0100204	0.01
X ₉ =Depth of rotor core	m	0.0300485	0.0303983	0.0300113	0.0300836	0.03	0.030007
X ₁₀ = Average flux density in the air gap	T	0.695273	0.699802	0.699634	0.699789	0.699646	0.699954
X ₁₁ = Area of cross section of end ring	mm ²	127.958	120.315	122.199	124.28	120.033	120.825
X ₁₂ = Current Density in Stator	A/mm ²	3.83868	3.5594	3.88272	3.35809	3.41067	3.34987
Weight	Kg	31.15756	30.91499	30.6275	30.6516	30.50282	30.81061
Cost	per unit	50.51716	50.12559	49.99667	49.99029	49.95902	49.94544
Total Losses	Watt	4288.606	4283.807	4062.839	3968.191	3913.807	4091.911
Efficiency		0.8014504	0.8019292	0.806242	0.8082064	0.8090209	0.8061232
Power Factor		0.774786	0.7771706	0.7798481	0.7778902	0.7787197	0.7746795

Table 4.3: Optimized Design Parameters for a 50 Hz machine according to number of generations

Design Variables	Units	50 Hz	60 Hz	Values Before Optimizing
X ₁ = Diameter of stator bore	m	0.101588	0.104137	0.15
X ₂ = Length of stator core	m	0.149076	0.145325	0.09
X ₃ = Width of stator slot	m	0.00999463	0.00999347	0.006
X ₄ = Depth of stator slot	m	0.0200845	0.0200211	0.024
X ₅ = Depth of stator core	m	0.0200009	0.020025	0.021
X ₆ = Length of the air gap	m	0.000492618	0.000498151	0.00045
X ₇ = Width of the rotor slot	m	0.00602661	0.00600708	0.0065
X ₈ = Depth of the rotor Slot	m	0.0100204	0.0100027	0.0105
X ₉ =Depth of rotor core	m	0.03	0.0300174	0.03
X ₁₀ = Average flux density in the air gap	T	0.699646	0.69989	0.5
X ₁₁ = Area of cross section of end ring	mm ²	120.033	120.015	120
X ₁₂ = Current Density in Stator	A/mm ²	3.41067	4.07993	3.5
Weight	Kg	30.50282	30.66175	33.287
Cost	per unit	49.95902	49.93099	53.2258
Total Losses	Watt	3913.807	3674.544	3875.87
Efficiency		0.8090209	0.8113441	0.8414
Power Factor		0.7787197	0.7615377	0.794723

Table 4.4: Optimized Design which is chosen after 6 (100-600) generations

The data for weight, cost, efficiency and power factor are compared at the end to choose the design that fits the demand of the required industry. Some industry may require different specifications of machine such as lightweight and other may opt for cheaper machine for their application.

The graph is drawn according to the motor type as follows:

- Type 1 : 50 Hz Motor (after optimizing)
- Type 2 : 60 Hz Motor (after optimizing)
- Type 3 : Original design motor before optimizing

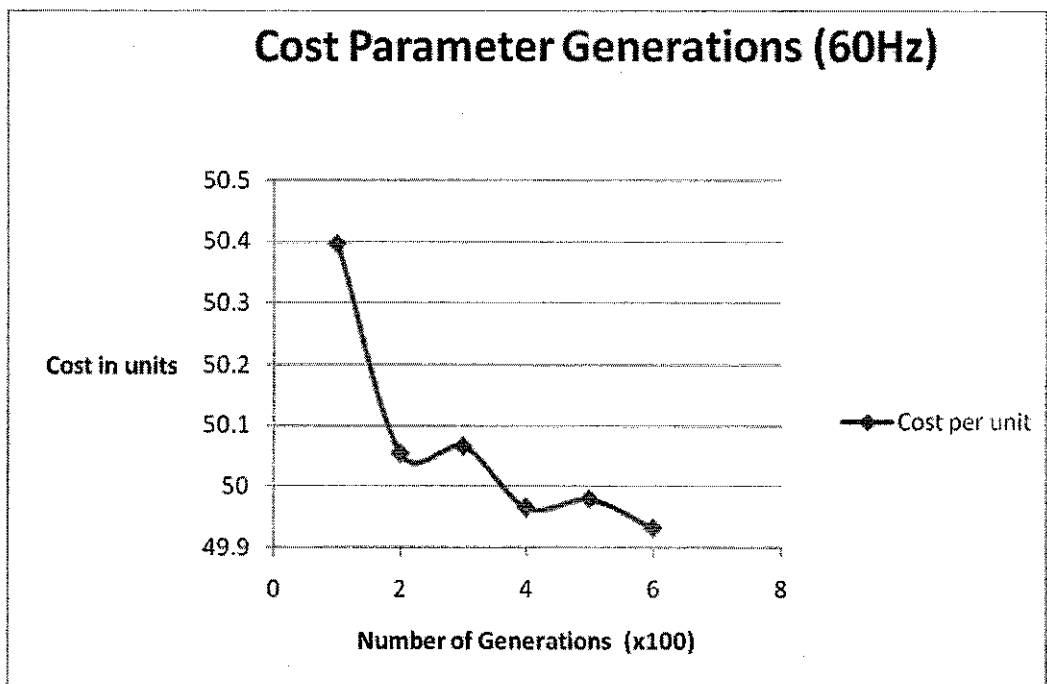


Figure 4.1 : Cost Vs Number of Generations(x100) for 60Hz machine

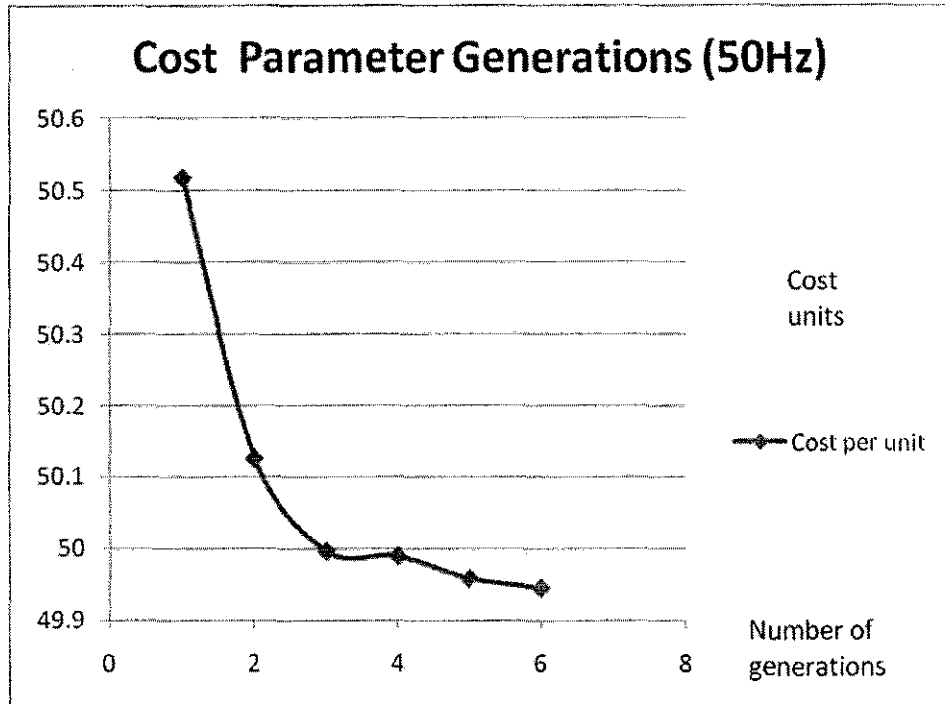


Figure 4.2: : Cost Graph Vs Number of Generations(x100) for 50Hz machine

As Figures 4.1 and 4.2 shows it can be seen that as the number of generations increases the value for cost and weight decreases and reaches a minimum point where further generations just does not reduce the cost parameter anymore without violating the constraints provided. The final value obtained is assumed an optimum value.

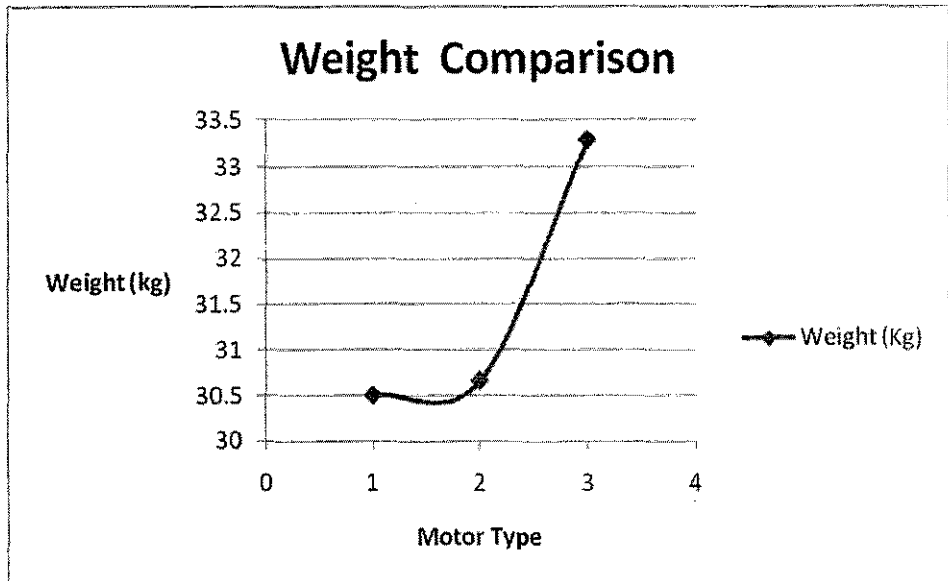


Figure 4.3: Weight Comparison Graph

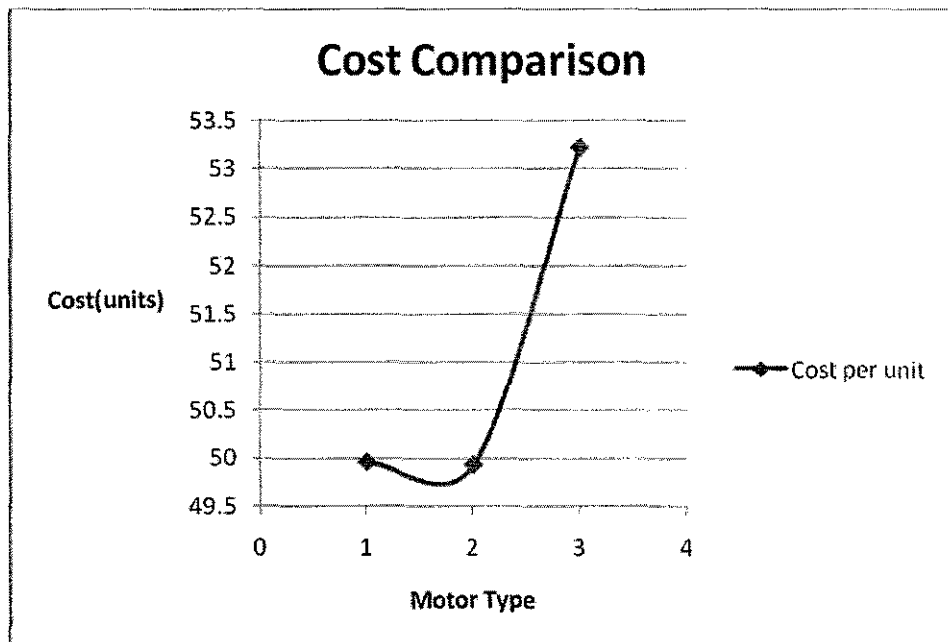


Figure 4.4 : Cost Comparison graph

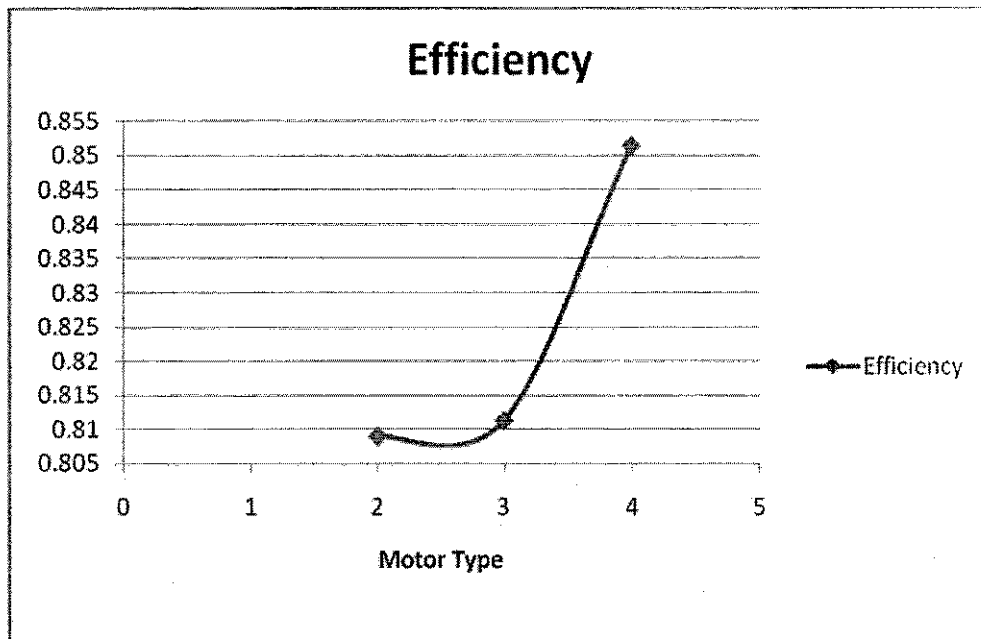


Figure 45: Efficiency Comparison Graph

4.2 Discussion

After obtaining the results a comparison is made on weight, cost and efficiency. As for the graph in Figure 4.3 the weight of the 50 Hz machine design is the lightest compared to the 60 Hz machine and the original design parameters used before optimization

For the cost comparison as shown in figure 4.4, the 60 Hz machine produces the least cost in unit terms compared the other two machine design. While for the efficiency comparison shown in figure 4.5, the original design variables still gives the highest efficiency.

Thus it is observed that to improve the weight and the cost of the machine a slight decrease in efficiency is noted from the results obtained after the optimization. Using the results obtained it is shown that to achieve successful designs with lower cost and weight and yet still being able to satisfy all the constraints given.

As we can see from Table 4.4 some of the dimensions of the design variables have shown variation during optimization by the Genetic Algorithm.

Thus it is demonstrated with an example that Genetic Algorithm (GA) program is highly suitable in order to achieve a lighter and less costly design while sacrificing slightly in the machine efficiency.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Relevancies to Objectives

The first objective, which is to develop a 3-Phase squirrel cage Induction motor using C language based programming. This design program is then successfully optimized using a Non-linear programming (NLP) technique which is able to optimize the design analysis program and to produce optimized design variables which improves the original design .To come up with the optimized design variables, GA or Genetic Algorithm is successfully applied.

After the results of the optimization are obtained the suitable generated results are selected as the final optimal design. In this project variable frequency of the 3-phase induction motor is tested with two different frequencies namely 50Hz and 60 Hz.

The design of the motor are obtained and as discussed in Chapter 4 is less weight and less cost compared to the original design. However to achieve a slight efficiency decrease is noted after the optimization.

5.2 Recommendations

One of the recommendations that can be suggested for this project is the increase of number design variables to further increase the machine design's details. The objective function can be selected as losses to maximize the efficiency .A comparative study of optimal parameters minimizing losses, weight and cost is recommended for different specifications of motor and for a variation of frequency between 20 - 80Hz.

5.3 Conclusion

Optimization using C language based program is a clever way to design and optimize electrical machine design parameters systematically. In this project GA based optimization technique has been used to a 5 hp 3-phase induction motor. The final design is compared with the original parameters before optimization. The result shows that lighter and cheaper induction motor can be produced after optimizing the design program. However the efficiency of the machines is reduced slightly in order to achieve the optimized values of the design variables.

REFERENCES

- [1] A.E.Fitzgerald, Charles Kingsley, Jr.; “*Electrical Machinery*”, 1992, Fifth Edition, Metric Edition ,pp 321-378
- [2] S K Bhattacharya; “*Electrical Machines*”, 1998, Second Edition, Mc Graw Hill, pp 288-397
- [3] Hans-Paul Schwefel, *Optimization*
 , John Wiley & Sons, Great Britain, 1981
- [4] G. F. Uler and O. A. Mohammed: *Utilizing Genetic Algorithms for the Optimal Design of Electromagnetic Devices*, IEEE, 1995
- [5] Stephan J.Chapman, 2005, *Electric Machinery Fundamentals 4th Edition*
- [6] Atanasi Jornet AEG Fabrica de motores, S.A., Test Laboratory, Terassa, Spain , Angel Orille, Alberto Perez and Diego Perez
Electrical Engineering Department, Industrial Engineering Technical High College, Polytechnic University of Catalonia, Barcelona, Spain, 2003

Optimal design of high frequency induction motor with the aid of finite element
- [7] Nicola Bianchi, 2005 *Electrical Machine Analysis Using Finite Elements*, Taylor & Francis Group
- [8] Ned Mohan, Torre M. Undeland, William P. Robbins, 1995, *Power Electronics Converters, Applications and Design*, Wiley, New York.
- [9] Richard Valentine, 1998. *Motor Control Electronic Handbook*, McGraw-Hill, New York.
- [10] M.G.Say, “*The performance and Design of Alternating Current Machines*”, 2002, 5th Editin , CBS.

- [11] Zhong Yiwen. *The Study on Intelligent Optimization Methods and Their Applications[D]*. Zhejiang University.
- [12] “Induction motor,photo” www.wikipedia.com
- [13] Mehmet Cunkas and Ramazan Akkaya, *Design Optimization Of Induction Motor By Genetic Algorithm and Comparison With Existing Motor*.
Selcuk University,Konya,Turkey

APPENDICES

APPENDIX (1)

Project Milestone for Final Year Project (FYP)

Table A: Milestone for First Semester

No.	Detail/ Week	1	2	3	4	5	6	7	8	9	10	Mid-semester break	11	12	13	14	15		
1	Selection of Project Topic	█	█																
	- Discussion with supervisor	█																	
	- Submission of project title		█																
2	Literature Review		█	█	█														
	Understanding the theory		█	█	█														
	Refer to journals and papers for current development		█	█	█														
3	Preliminary Report Submission				█														
4	Seminar 1 (Optional)							█											
5	Data Collection / Tabulation													█	█	█	█		
6	Preliminary Design Study							█	█	█	█								

Appendix (2)

Project Milestone for Final Year Project (FYP)

Table B.2: Milestone for Second Semester

No.	Detail/ Week	1	2	3	4	5	6	7	8	9		10	11	12	13	14	
1	Optimization Technique Studies										Mid-semester						
2	Progress Report 1																
3	Developing Optimization Program																
	- Testing																
	- Troubleshooting																
4	Progress Report 2																
5	Seminar																
7	Poster Exhibition																
8	Dissertation Report (Soft Bound)																
9	Oral Presentation																
10	Dissertation Report (Hard Bound)																

APPENDIX 2

Genetic Algorithm Program Developed For Optimizing and Integrated with the Induction Motor Design

/ Declare variables for the GA parameters and set them to some default values.

```
int popsize = 100;
```

```
int ngen = 100;
```

```
float pmut = 0.044;
```

```
float pcross = 0.8867;
```

```
//float pmut = 0.02;
```

```
//float pcross = 0.5;
```

```
// Create a phenotype for n variables. The number of bits you can use to
```

```
// represent any number is limited by the type of computer you are using. In
```

```
// this case, we use 16 bits to represent a floating point number whose value
```

```
// can range from -5 to 5, inclusive. The bounds on x1 ...xn can be applied
```

```
// here and/or in the objective function.(constraints as variables)
```

```
//
```

```
GABin2DecPhenotype map;
```

```
map.add(16,0.1,0.2);
```

```
map.add(16,0.09,0.15);
```

```
map.add(16,0.006,0.01);
```

```
map.add(16,0.02,0.04);
```

```
map.add(16,0.02,0.03);
```

```
map.add(16,0.3e-03,0.5e-03);
```

```
map.add(16,0.006,0.01);
```

```
map.add(16,0.01,0.03);
```

```
map.add(16,0.03,0.05);
```

```

map.add(16,0.5,0.7);
map.add(16,120.0,140.0);
map.add(16,3.0,4.5);
//next:
// Create the template genome using the phenotype map we just made.
GABin2DecGenome genome(map, objective);
// Now create the GA using the genome and run it. We'll use sigma truncation
// scaling so that we can handle negative objective scores.
GASteadyStateGA ga(genome);
GASigmaTruncationScaling scaling;
ga.minimaxi(-1);
ga.populationSize(popsiz);
ga.nGenerations(nngen);
//next:
ga.pMutation(pmut);
ga.pCrossover(pcross);
ga.scaling(scaling);
ga.scoreFilename("trans.doc");
ga.scoreFrequency(1);
ga.flushFrequency(50);
ga.selectScores(GAStatistics::Minimum);
ga.evolve(seed);

// Dump the results of the GA to the screen.
genome = ga.statistics().bestIndividual();
cout.precision(7);

// effect of pef
/*

```

```

    pefc=5;
    pefw=3;
    pefv=0.001;
*/

//cout << "Design of 3-ph Rectifier transformer - results";
cout << "\npopsize=" << popsize << "- " << "ngen=" << ngen;
cout << "\n x1 = ";
cout << genome.phenotype(0) << " m\n";
cout << " x2 = ";
cout << genome.phenotype(1) << " m\n";
cout << " x3 = ";
cout << genome.phenotype(2) << " m\n";
cout << " x4 = ";
cout << genome.phenotype(3) << " m\n";
cout << " x5 = ";
cout << genome.phenotype(4) << " m\n";
cout << " x6 = ";
cout << genome.phenotype(5) << " m\n";

cout << " x7 = ";
cout << genome.phenotype(6) << " m\n";
cout << " x8 = ";
cout << genome.phenotype(7) << " m\n";
cout << " x9 = ";
cout << genome.phenotype(8) << " m\n";
cout << " x10 = ";
cout << genome.phenotype(9) << " m\n";
cout << " x11 = ";

```

```

cout << genome.phenotype(10) << " mm\n";
cout << " x12 = ";
cout << genome.phenotype(11) << " A/mm2\n";

```

```

double x1 = genome.phenotype(0),x2 = genome.phenotype(1),x3 = genome.phenotype(2);
double x4 = genome.phenotype(3),x5 = genome.phenotype(4),x6 = genome.phenotype(5);

```

```

double x7 = genome.phenotype(6),x8 = genome.phenotype(7),x9 = genome.phenotype(8);
double x10 = genome.phenotype(9),x11 = genome.phenotype(10),x12 =
genome.phenotype(11);

```

```

// design equations

```

```

// design variables x1=diameter of stator bore, m, x2= length of stator core, m,
// x3=width of stator slot, m x4= depth of stator slot, m x5= depth of stator core, m,
// x6= length of the airgap, m x7 = width of rotor slot, m, x8= depth of rotor slot, m,
// x9= depth of rotor core, m, x10= average flux density in the airgap, T,
// x11= Area of cross-section of endring, mm2, x12 = current density -stator , A/mm2

```

```

// motor data - 1

```

```

hp=5.0;freq=55.0;v = 400.0;npl=6; efy = 0.87;pfa = 0.85;nspp = 3;
//nspp=slots/pole/ph
nfn=10; slip = 0.04;

```

```

// constants

wduc = 0.01; h3 = 0.5; h4 = 1.0; hw= 1.0; ho=1.0; w1= 0.003; w2 = 0.001;

clre = 0.01; clpr = 0.01; cikg = 1.0; cckg = 2.0; cbkg = 2.0;

pi      =      3.1415926;

crvf    =      v/frq;                //v/f ratio

//      for(i=1;i<=10;i++)
//      {

v        =      crvf*frq;            //volts/ph -
Fundamental comp - rms voltage

edc      =      v* pow(2.0,0.5);     // dc voltage -

//

Fundamental, a1 = 1.0 Edc, Vph = a1/pow(2.0,0.5)

rpm      =      120.0*frq/npl;

cva      =      hp*746.0/(efy*pfa);  // voltamps

eph      =      v/1.05;              // back emf

pmu      =      4.0*pi*1.0e-07;

nduc     =      x2/0.05;

eli      =      0.9*(x2-wduc*nduc);  // effective iron

length

twp      =      pi*x1/npl;           // stator pole

pitch

phi      =      x10*twp*x2;         // flux = B Y L

cph      =      cva/(3.0*eph);

tph      =      eph/(4.44*0.955*frq*phi); // turns/ph

nzs      =      tph/(npl/2*nspp)/2*2; // stator conds/slot

ntph     =      npl/2*nspp*nzs;     // stator turns/phase

stator slots

ns1      =      3*nspp*npl;         // number of

```

```

slots/pole    gs1      =      ns1/npl;                                // stator

//stator winding design

stwdg(nzs,x3,x4,x12,h3,h4,cph,sigma,sdi,swi,aos,bos,acs,ncw,ncd,cdty);

nzs          =      ncw*ncd;
ntph   =      npl/2*nspp*nzs;
akp          =      1.0;                                //pitch
factor - full pitch
akd          =      sin(nspp*pi/gsl/2.0)/(nspp*sin(pi/gsl/2.0));
//distribution factor
akw1   =      akp*akd;
//winding factor
stator slot pitch    tos      =      pi*x1/ns1;                                //

wtt          =      tos - x3;
wth          =      pi*(x1+x4)/ns1-x3;
wtb          =      pi*(x1+2.0*x4)/ns1 - x3;
ap1          =      phi*npl/ns1/eli;
bfta   =      ap1/wtt;
btha   =      ap1/wth;
btba   =      ap1/wtb;
sks          =      tos*x2/eli;

curat(bfta,atft,perm);
curat(btha,atth,perm);
curat(btba,atfb,perm);

```



```

btf      =      bfta-pmu*atft*(sks/wft-1.0);
bth      =      btha-pmu*atth*(sks/wth-1.0);
btb      =      bfta-pmu*atfb*(sks/wtb-1.0);
bt1      =      (btf+4.0*bth+btb)/6.0;

```

```

curat(bt1,at1,perm);

```

```

atf      =      at1*x4;
parc     =      0.7*twp;
dex      =      x1+2.0*x4+2.0*x5;
bcr1     =      phi/(2.0*x5*eli);

```

```

curat(bcr1,acr1,perm);

```

```

atc      =      acr1*pi*(x1+2.0*x4+x5)/(2.0*npl);
ewdl     =      1.15*twp+0.12;
ctrs     =      0.021*ntph*2.0/acs;
rsc      =      ctrs*x2;           //stator winding
resistane(slots)/phase
rse      =      ctrs*ewdl;       //stator end winding
resistance/phase
rs       =      rsc+rse;        //stator winding resistance

```

```

// squirrel-cage rotor design

```

```

nr2      =      ns1+npl/2;       //number of rotor slots

```

```

goto lb15;

```

```
lb122: nr2      =      nr2+1;
```

```
lb15:  if((nr2-ns1) != npl || (nr2-ns1) != npl/2) goto lb111;  
      goto lb122;
```

```
lb111: if(nr2 != 10*npl || nr2 != 8*npl) goto lb13;
```

```
lb13:  ns2      =      nr2;
```

```
      d2      =      x1-2.0*x6;  
pitch  tor     =      pi*d2/ns2;                // rotor slot
```

```
      el2     =      pow((x2*x2+tor*tor),0.5);
```

```
      // gs2  =      float(ns2)/float(npl);
```

```
      gs2     =      float(ns2/npl);
```

```
      aor     =      x8*1000.0 -(hw+ho) - 0.5;
```

```
      bor     =      x7*1000.0 - 0.5;
```

```
      acr     =      aor*bor;
```

```
      cr2p    =      cph*pfa;
```

```
      cib     =      akw1*gs1*nzs/gs2*cr2p;
```

```
      cer     =      cib*gs2/pi;
```

```
      cdrb    =      cib/acr;
```

```
// aer      =      x11*1.0e+02 ;                // area of c.s.of end ring
```

```
      aer     =      x11;
```

```
      der     =      aor+5.0+10.0 ;           // depth of end ring
```

```
      wer     =      aer/der;                // width of end ring
```

```

cder = cer/aer; // current density of end ring
wbbr = el2+2.0*clre*0.001; //clre= clearance, m
albr = wbbr+2.0*(wer+clpr)*0.001 ; //clpr= clearance, m
of bar
rbr = 0.021*albr/acr; // resistance

dier = d2-2.0*der*0.001+0.5*der*0.001;
ring
rer = 0.021*pi*dier/aer; // resistance of end

rbr1 = 0.021*el2/acr; // resistance of bar
rrp = 4.0*3.0*akw1*akw1*ntph*ntph/ns2*(rbr+rer*gs2/
end ring
(pi*pi*npl/2.0)); // eq resistance of rotor bars and

akp2 = 1.0;
akd2n = sin(ns2*pi/(3*npl*gs2*2.0));
akd2d = ns2*sin(pi*gs2/2.0)/(3*npl);
akd2 = akd2n/akd2d;
akw2 = akp2*akd2;
wtrt = tor-x7;
wtrh = pi*(d2 - x8)/ns2 - x7;
wtrb = pi*(d2 - x8)/ns2 - x7;

ap2 = phi*npl/ns2/el2;

btr1 = ap2/wtrt;
btr2 = ap2/wtrh;
btr3 = ap2/wtrb;

curat(btr1,atr1,perm);
curat(btr2,atr2,perm);

```

```
curat(btr3,atr3,perm);
```

```
skr      =      tor*x2/el2;  
bt2t    =      btr1 - pmu*atr1*(skr/wtrt-1.0);  
bt2h    =      btr2 - pmu*atr2*(skr/wtrh-1.0);  
bt2b    =      btr3 - pmu*atr3*(skr/wtrb-1.0);  
bt2      =      (bt2t + 4.0*bt2h + bt2b)/6.0;
```

```
curat(bt2,at2,perm);
```

```
attr    =      at2*x8;  
dshf    =      (d2-2.0*x8 - 2.0*x9)*100.0;      //diameter of shaft  
bcr2    =      phi/(2.0*x9*el2);
```

```
curat(bcr2,acr2,perm);
```

```
atcr    =      acr2*pi*(d2-2.0*x8-x9)/(2.0*npl);  
opening rko1    =      w1/x6;                      //stator slot  
  
opening rko2    =      w2/x6;                      //rotor slot
```

```
cartc(rko1,ako1);
```

```
cartc(rko2,ako2);
```

```
akg1    =      tos/(tos-w1*ako1);  
akg2    =      tor/(tor-w1*ako2);  
akg      =      akg1*akg2;  
elg      =      x6*akg;
```

```

ag      =      twp*x2;

//      wt1      =      pi*(x1+2.0/3.0*x4)/ns1-x3;
//      wt1      =      0.5*((pi*x1/ns1-x3)+(pi*(x1+2.0*x4)/ns1-x3));

wt1     =      x3;
sta     =      gs1*el1* wt1 ;
wt2     =      x7;

//      wt2     =      pi*(d2-2.0/3.0*x8)/ns2 - x7;
//      wt2     =      0.5*((pi*d2/nr2-x7)+(pi*(d2-2.0*x8)/nr2 - x7));

rta     =      gs2*el2*wt2;
alcs    =      pi*(x1+2.0*x4+x5)/(2.0*npl);
alcr    =      pi*(d2-2.0*x8-x9)/(2.0*npl);
bcsa    =      1.28*bcr1;

btsa    =      1.28*phi/sta;
bga     =      1.28*phi/ag;
btra    =      1.28*phi/rta;
bcra    =      1.28*bcr2;

curat(bcsa,tcsa,pecs);
curat(btsa,ttsa,pets);
curat(btra,ttra,petr);
curat(bcra,tcra,pecr);

atga    =      bga*elg/pmu;
ata     =      tcsa*alcs+ttsa*x4+ttra*x8+tcra*alcr+atga;

```

```

cior = ata*npl/(2.0*ntph*1.17*akw1);
xm = eph/cior;

ptap = ((ncd/2)*aos/(3.0*x3)+(sdi+ncd/2*aos)/x3+2.0*h3/(x3+w1)+h4/w1)/1000.0;
pbtm = ((ncd/2)*aos/(3.0*x3)+2.0*h3/(x3+w1)+h4/w1)/1000.0;
pmtb = ((ncd/2)*aos/(2.0*x3)+2.0*h3/(x3+w1)+h4/w1)/1000.0;
ps1 = ptap+pbtm+2.0*pmtb;

xkf = 15.8e-06*frq*ntph*ntph/(npl/2*nspp);
xs1 = xkf*x2*ps1;
hang = 0.7*ewdl-0.4*pi*(x1+x4)/npl;
xo1 = xkf * hang;
xz1 = 5.0 * xm /(6.0 * gs1 * gs1);

wo = w2;
ps2 = (aor/(3.0*x7)+1.0/x7+2.0*hw/(x7+wo)+ho/wo)/1000.0;
xs2 = xkf*x2*ps2;
xo2 = 2.0*pi*frq*pmu*(ns2*ns2)/(3*npl)*2.0/3.0*(wbbr-
x2+0.18*pi*dier/npl);
xz2 = 5.0/6.0*xm/(gs2*gs2);
x2t = xs2+xo2+xz2;
x2tp = x2t*akw1*akw1*ns1/ns2/(akw2*akw2);

gscw = (x2+ewdl)*ntph*2.0*3.0*acs*1.0e-06*8.93*1.0e+03;
grcw = el2*ns2*acr*1.0e-06*8.93*1.0e+03;
gend = 2.0*pi*dier*aer*1.0e-06*8.7*1.0e+03;
gsth = ns1*eli*(wtt+wtb)/2.0*x4*7.55*1.0e+03;
gscr = pi*(x1+2.0*x4+x5)*x5*eli*7.55*1.0e+03;

```

```

grth = ns1*el2*(wtrf+wtrb)/2.0*x8*7.55*1.0e+03;

grcr = pi*(d2-2.0*x8-x9)*x9*el2*7.55*1.0e+03;
gcp = gscw+grcw;

gbr = gend;
gir = gsth+gscr+grth+grcr;
aloa = 0.07+0.3*twp;
alov = x2+2.0*aloa;
are1 = pi*x1*alov+2.0*pi*(x1+2.0*x4)*(dex-(x1+2.0*x4));
are2 = pi*dex*x2;
pps = pi*x1*rpm/60.0;
area = are1*(1.0+0.1*pps)+are2;
bedi = dshf-1.0;
belt = 2.75*bedi;
vber = pi*bedi*0.01*rpm/60.0;
befr = 0.3*bedi*belt*pow(vber,1.5);
wind = 0.8*d2*100.0*alov*100.0*pps*pps*1.0e-03;
frwd = befr+wind;

```

```
// tanya - 41 parameters
```

```

// tanya(x1,x2,x3,x4,x5,x6,x7,x9,csfh,crfh,rmst,rmsh,rmsv,pf,vlf,csck,cnlf,pfsc,
// pscu,prcu,pts,pcs,ptr,pcr,pss,psr,pszz,psse,prre,pstp,psbn,pskn,
// hp,rpm,frwd,cioh,sscp,sscl,tqst,tqmax,sln);

```

```
// Eq ckt (tanya) included in main
```

```
m = 1;
```

```

prcu   = 0.0;
prcu   = 0.0;
pts     = 0.0;
pcs     = 0.0;
ptr     = 0.0;
pcr     = 0.0;
pi      = 3.14159265358;
wms     = 2.0*pi*rpm/60.0;

```

```
//do 25 i=1, nh, 2
```

```
  for(i=1;i<=nh;i=i+2)
```

```
{
```

```
  it     = i/3*3;
```

```
  if (i == it) goto lb25;
```

```
  if (i == 1) goto lb26;
```

```
  ifw    =      i;
```

```
  ibw    =      i;
```

```
  if (ifw == (6*m+1)) goto lb27;
```

```
  if (ibw == (6*m-1)) goto lb28;
```

```
lb27:  sl[i] = ((i-1) + slip)/i;
```

```
      m     = m+1;
```

```
      goto lb26;
```

```
lb28:  sl[i] = ((i+1) - slip)/i;
```

```
      m     = m+1;
```

```
      goto lb26;
```



```

lb26: flpt = i;
      hrfq = frq*flpt;
      if(i==1) sl[i] = slip;
      sli = sl[i];
      akps[i] = 1.0;
//      akds[i] = sin(flpt*nspp*pi/gsl/2.0)/(nspp*sin(flpt*pi/gsl/2.0));
      akds1[i] = sin(nspp*flpt*pi/(gsl*2.0));
      akds2[i] = nspp*sin(flpt*pi/(gsl*2.0));
      akds[i] = akds1[i]/akds2[i];
      akws[i] = akps[i]*akds[i];
      akpr[i] = 1.0;
      rspp = ns2/(3*npl);
      akdr[i] = sin(flpt*rspp*pi/gs2/2.0)/(rspp*sin(flpt*pi/gs2/2.0));
      akwr[i] = akpr[i]*akdr[i];

      ges[i] = aos*pow(pi*pmu*bos*ncw*hrfq/(x3*1000.0*0.021),0.5);
      gesi = ges[i];

//      zess(ges[i],fes[i],ses[i],akx1[i]);
      zess(gesi,fesi,sesi,akx1i);
      fes[i] = fesi;
      ses[i] = sesi;
      akx1[i] = akx1i;
      akr1[i] = fes[i] + ses[i] *(ncd*ncd - 1.0)/3.0;
      r1[i] = akr1[i]*rsc+rse;
      x1ta[i] = akx1[i]*xs1+xo1+xz1;
      ger[i] = aor*pow(pi*pmu*bor*sl[i]*hrfq/(x7*1000.0*0.021),0.5);
      geri = ger[i];

```

```

//      zess(ger[i],fer[i],ser[i],akx2[i]);
      zess(geri,feri,seri,akx2i);

      fer[i] =      fer;
      ser[i] =      seri;
      akx2[i] =     akx2i;
      akr2[i] =     fer[i];

//      e22[i] =     12.0*akws[i]*akws[i]*ntph*ntph/(ns2*akwr[i]);

      e22[i] =     12.0*akws[i]*akws[i]*ntph*ntph/float(ns2);

      r2p[i] =     e22[i]*(akr2[i]*rbr1+0.021*(albr-el2)/acr+rer*gs2/(pi*pi*npl/2));

//      x2p[i] =     akx2[i]*xs2*akws[i]*akws[i]*ns1/(akwr[i]*akwr[i]*ns2)+xo2+xz2;

//      x2p[i] =     akx2[i]*xs2*akws[i]*akws[i]*ntph*ntph/ns2+xo2+xz2;

      x2p[i] =     akx2[i]*xs2*akws[i]*akws[i]*ns1/ns2+xo2+xz2;

//      vn[i] =     3.0*edc/(pow(2,0.5)*pi*flpt);

      vn[i] =     edc/pow(2,0,0.5);

      cpx1 =     flpt*x1ta[i];

```

```

// Complex impedances -----
// zsn[i] = stator complex impedance.
zsn[i].Re    =    r1 [i];
zsn[i].Im    =    cpx1;

xmm[i]       =    xm;
cpxm         =    flpt*xmm[i];

// zmn[i] = Magnetizing branch complex impedance - rm neglected..
zmn[i].Re    =    0.0;
zmn[i].Im    =    cpxm;

rzn         =    r2p[i]/sl[i];
cjr         =    flpt*x2p[i];

// zrn[i] = Rotor complex impedance
zrn[i].Re    =    rzn;
zrn[i].Im    =    cjr;
cjse        =    flpt*(x2p[i]+xmm[i]);

//zsen[i] = sum of (magnetizing branch + rotor) complex impedance..
zsen[i].Re    =    rzn;
zsen[i].Im    =    cjse;

// ztn[i] =    zsn[i]+zmn[i]*zrn[i]/zsen[i];

```

```

// conjugate of zsen[i]

conzsen[i] = conj(zsen[i]);

// ratio zrn[i]/zsen[i] = nzrn[i]/dzrn[i] = ztn1 [i]
nzrn[i] = zrn[i] * conzsen[i];
dzrn[i] = zsen[i]* conzsen[i];

// modulus value of dzrn[i]

mdzrn[i] = dzrn[i].Re;

// ratio of impedances zrn[i]/zsen [i]

ztn1 [i] = (1.0/mdzrn[i]) * nzrn[i];
ztn2[i] = zrn[i] * ztn1 [i];
ztn[i] = zsn[i] + ztn2[i]; // complex quantity, ztn[i]

//absolute value of ztn[i]
//zzn[i] = cabs(ztn[i]);

conztn[i] = conj(ztn[i]);
zzn[i] = ztn[i] * conztn[i];
zznisq[i] = zzn[i].Re; //
absolute value of ztn[i] = zzn[i]

zzni[i] = pow(zznisq[i],0.5);
cisin[i] = vn[i]/zzni[i]; //
Absolute value of stator current

// vnv[i] = complex ..

```

```

vnv[i].Re    =    vn[i];
vnv[i].Im    =    0.0;

// complex motor total current, cztn[i] = vnv[i]/ztn[i]=ncztn[i]/dcztn[i]

ncztn[i]=    vnv[i]*conztn[i];
dcztn[i]    =    ztn[i]*conztn[i];
mdcztn[i]   =    dcztn[i].Re;

cztn[i]     =    (1.0/mdcztn[i])*ncztn[i];

// Stator impedance drop

vzsn[i]     =    cztn[i]*zsn[i];

// Stator emf, evn[i] = vnv[i] - vzsn[i]

evn[i]      =    vnv[i]-vzsn[i];

//en[i] =    cabs(evn[i]);

conevn[i]   =    conj(evn[i]);
en[i]       =    evn[i] * conevn[i];
enresq[i]   =    en[i].Re;
enre[i]     =    pow(enresq[i],0.5);
enim[i]     =    en[i].Im;

phs[i]      =    enre[i]/(4.44*akws[i]*hrfq*ntph);    //Harmonic flux

```

```

bts[i] = phs[i]/(wt1*gs1*eli);

// Define subscripted variables as nonsubscripted for all functions

//      btsi = bts[i]; athsi = aths[i]; pmsi = pms[i]

//      curat(bts[i],aths[i],pms[i]);

      btsi = bts[i];
      curat(btsi,athsi,pmsi);
      aths[i] = athsi;
      pms[i] = pmsi;

//      rict(sigh,bts[i],hrfq,pms[i],gsth,hts[i],edts[i]);
      sigh = 3.0; //for
0.35 mm thick laminations

      rict(sigh,btsi,hrfq,pmsi,gsth,htsi,edtsi); // Hys and eddy current loss - Richter's

//      hts[i] = htsi;
//      edts[i] = edtsi;
//      wts[i] = hts[i]+edts[i];
      bcs[i] = phs[i]/(2.0*x5*eli);

//      curat(bcs[i],acoas[i],pmc[i]);

      bcsi = bcs[i];
      curat(bcsi,acoasi,pmci);
      acoas[i]= acoasi;
      pmc[i] = pmci;

//      rict(sigh,bcs[i],hrfq,pmc[i],gscr,hcs[i],edcs[i]);

```

```

    rict(sigh,bcsi,hrfq,pmci,gscr,hcsi,edcsi);
    hcs[i] = hcsi;
    edcs[i] = edcsi;

    wcs[i] = hcs[i]+edcs[i];
    btr[i] = phs[i]/(wt2*gs2*el2);

//    curat(btr[i],athr[i],pmr[i]);

    btri = btr[i];
    curat(btri,athri,pmri);
    athr[i] = athri;
    pmr[i] = pmri;

//    rict(sigh,btr[i],hrfq,pmr[i],grth,htr[i],edtr[i]);

    rict(sigh,btri,hrfq,pmri,grth,htri,edtri);
//    htr[i] = htri;
//    edtr[i] = edtri;
//    wtr[i] = htr[i]+edtr[i];
    bcr[i] = phs[i]/(2.0*x9*el2);

//    curat(bcr[i],acor[i],pmcr[i]);

    bcri = bcr[i];
    curat(bcri,acori,pmcri);

```