OPTIMAL DESIGN OF A BLDC MOTOR BY GENETIC ALGORITHM

By

AZRUL HISHAM BIN OTHMAN

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

OPTIMAL DESIGN OF A BLDC MOTOR BY GENETIC ALGORITHM

by

AZRUL HISHAM BIN OTHMAN

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:

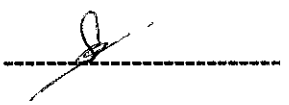
Assoc. Prof. Dr. K. S. Rama Rao
Project Main Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2007

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 reference and acknowledgement, and that the original work contained herein have not been undertaken or done by unspecified sources or persons



AZRUL HISHAM BIN OTHMAN

ABSTRACT

Brushless DC (BLDC) motors are widely used in many applications in the industry and as such, the design of the motor and its control circuits are very important. BLDC motor is a permanent magnet motor with electronic commutation. The design procedures of these motors are much different from that of traditional motors. The project report describes an optimal design of Brushless DC (BLDC) motor using Genetic Algorithm (GA) and Simulated Annealing (SA). A constrained optimization on the objective function is performed and optimal parameters are derived. The resulting effects of varying GA parameters such as population size, number of generations, and the amount of mutation and crossover fraction, are also presented for single and multi-objective functions. In the case of SA technique, the effect of varying number of iterations on the objective function is analyzed. The design and analysis of the motor are performed using software tools within the C/C++ programming environment. The optimal design parameters of the motor obtained by GA are compared with those obtained by SA technique.

Keyword: Brushless DC motor, genetic algorithm and simulated annealing.

ACKNOWLEDGEMENT

Thanks to merciful **Allah s.w.t for** His blessings of giving me a chance to finish my Final Year Project (FYP). Also, thanks to all who gave me support, encouragement, concerns and help during the entire project period :

- To loving parents, **Aini Binti Amasasi** and **Azlinda Idayu Binti Othman** for their never ending love, concern, support, motivation and care ensured performing the best for my FYP.
- A biggest appreciation and compliments to **Associate Professor Dr. K.S.Rama Rao** for being my supervisor. All valuable knowledge taught, project advised, supervised; kindness and cooperation given were really appreciated.
- To all helpful, kind, supportive and knowledgeable UTP staffs especially from Electrical and Electronics Engineering department. Especially to **Puan Siti Hawa Binti Haji Mohd Tahir** for assisting me throughout my FYP.
- Moreover, thanks also to my colleagues **Mr. Muhammad Ariff, Mr. Muhammad Firdaus and Ms. Mayarisma** who were so much friendly, helpful, funny, supportive and cooperative during my entire Final Year Project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

BLDC	Brushless Direct Current Motor
GA	Genetic Algorithm
SA	Simulated Annealing
PM	Permanent Magnet
SWG	Standard Wire Gauge
Nd-Fe-B	Neodymium Iron Boron

CHAPTER 1
INTRODUCTION

1.1 Background of Study

Brushless DC (BLDC) motor is a rotating electric machine where the stator is a classic three-phase stator like that of an induction motor and the rotor has permanent magnets. In this respect, the BLDC motor is equivalent to a reversed DC commutator motor, in which the magnet rotates while the conductors remain stationary. In the DC commutator motor, the current polarity is altered by the commutator and brushes. The Brushless DC motor (BLDC) is also referred to as an electronically commuted motor. The commutation is performed electronically at certain rotor positions. The stator is usually made from magnetic steel sheets [1].

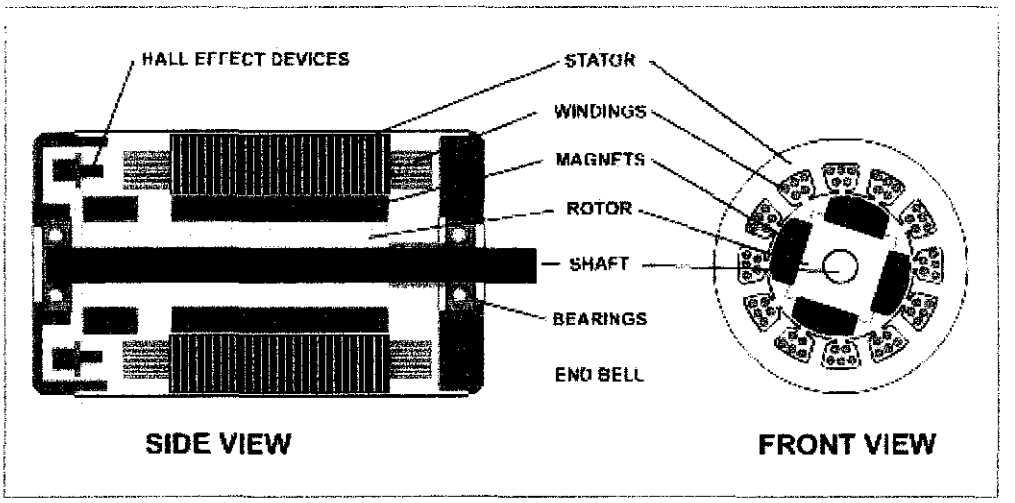


Figure 1: Cross Section of BLDC motor.

1.2 Problem Statement/Problem Identification

BLDC motor is widely used when high torque-to-volume and torque-to-inertia ratios are required, as well as high starting torque and regular running under speed and position control. A further peculiar characteristic of the BLDC motor is its high flexibility to be designed in several forms.

This project utilized a procedure for the design of a BLDC motor tightly combined with the thermal analysis. A steady state analysis of a BLDC motor supports the design procedure. It gives the expression of the main electrical, magnetically, mechanical and thermal quantities as a function of the machine dimensions and working condition. Thus the motor design is reached by solving the non linear equations, derived from the aforementioned working analysis, where the motor performance, the material stress limits, and other constraint are imposed. Taking advantage of the completely analytical design procedure, an optimization procedure to individuate the best design of the motor has been easily developed.

1.3 Objectives and Scope of Study

The main objective of this project is to create and develop an optimal design of BLDC motor using genetic algorithm. Besides, the knowledge of C++ or C programming is essential in order to develop a design procedure for this project. In addition to GA, another optimization technique is also used for BLDC motor design. The design analysis is compared by both the methods (Simulated Annealing and Genetic Algorithm).

CHAPTER 2
LITERATURE REVIEW

2.1 Genetic Algorithm

The genetic algorithm is one of the artificial intelligence techniques. It is a search procedure emulating the mechanism of natural selection and natural genetics. When applied to motor design optimization, the Genetic Algorithm (GA) explores the motor design variable space by means of the mechanisms of reproduction, crossover and mutation, with the aim of producing the best motor design [2]. To apply the GA approach, a fitness function *F(X)* has to be defined to evaluate how good each motor design is. This fitness function is, of course, closely related to the objective function and often coincides with it. Strings of binary digits (representing sets of values of the motor design variables) are manipulated by GA that measure the strength of each string by a fitness value given by the value of the fitness function, possibly modified by the penalty terms. The fittest strings advance and mate with other strings to produce off springs.

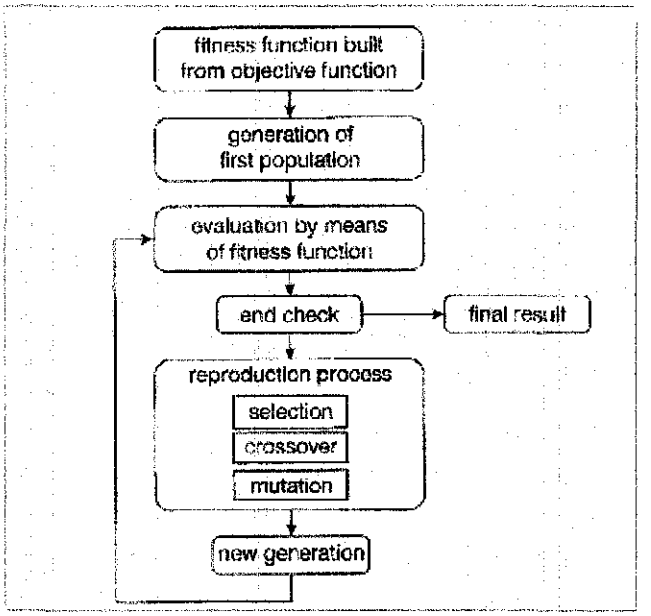


Figure 2: Main Steps of Genetic Algorithm Technique [7]

The main steps of the genetic algorithms are reported as follows [5]:

1) The first step lies in building fitness function from the objective function.

2) A population of N individuals (N different motor designs) is randomly generated, each of them characterized by a string of digits, zeros and ones. The string is composed by some concatenated substrings, describing the design variables X_i, with a suitable accuracy which determines the substring length.

3) All the individuals of the population are evaluated by means of the fitness function $f(X_i)$. The best fitness F_{best} is calculated. The average fitness F_{av} of the population as well as the global fitness F_{gl} is also often evaluated.

4) At this stage, the rules of the genetic algorithms apply in order to generate a new population of N individuals. The reproduction process is composed by the following three steps:

2.1.1 Selection

Individuals of the old population are selected and put in the new one, according to a rule that favors those with higher fitness. The selection can be a stochastic sampling (by which the best individual can be selected several times, while the worst one can be excluded, according to a selection probability expressed as $p_i=f(x_i)/\sum_i f(x_i))$ or a deterministic sampling (by which the best individuals are selected and the worst ones excluded).

2.1.2 Crossover

Two randomly selected strings, among those selected in the previous step, are mated. A position along one string is again randomly selected and all binary digits following this position are swapped with those of the second string. Then the two entirely new strings move on to the new generation. It is worth to note that the

Cold Work: When a metal is physically deformed at temperatures that are relatively low compare to its melting point, it is said to be cold worked. A rough rule of thumb is to assume the plastic deformation corresponds to cold working is carried out at temperatures lower than one half of the melting point at absolute scale. Most of the energy expanded in cold work appears in the form of heat. However, a finite fraction is stored in the metal as strain energy.

Recovery: At this stage, not much change in microstructure. However, atomic mobility is sufficient to diminish the concentration of point defects within grains and in some cases, allow dislocations to move to lower energy state.

Recrystallization: The temperature at which atomic mobility is sufficient to affect mechanical properties is approximately one third to one half times the absolute melting point. New stress free grains nucleate at high stress regions in the cold worked microstructure. These grains grow until they constitute the entire microstructure. The decrease in hardness is substantial during recrystallization process.

Grain Growth: The microstructure developed during recrystallization occurred spontaneously. It is stable compared with original cold worked structure. However, recrystallized microstructure contains a large concentration of grain boundaries. The reduction of these high-energy interfaces is a method of stabilizing a system.

2.2.3 Boltzmann Distribution

After looking at physical annealing process of a metal, we further describe the simulated annealing process in probabilistic and statistical aspects.

Simulated annealing process originated from the analogy between 2 problems. The first problem is finding the ground state of a solid and the second problem is finding a globally minimal configuration in a combinatorial optimization problem. In solid-state physics, annealing implies a physical process by which, if carried out slow enough, the ground state of the solid can be obtained. The simulated annealing

algorithm takes the name because of the fact that it is based on an algorithm to simulate the annealing process.

At each given temperature, the solid is allowed to reach thermal equilibrium. In thermal equilibrium, the probability of occurrence of a state with energy E is given by the Boltzmann Distribution:

$$\Pr(\overline{E} = E) = \frac{1}{Z(T)} \exp\left(\frac{-E}{K_B T}\right) \qquad (2.1)$$

where $Z(T)$ is the partition function, K_B is the Boltzmann constant, and $\exp\left(\frac{-E}{K_B T}\right)$ is the Boltzmann factor. As the temperature decreases, the Boltzmann distribution concentrates on the low energy states. When the temperature approaches zero, only the minimum energy states have a non-zero probability of occurrence.

It is important to know that if the cooling is taken place too fast, the solid will not reach thermal equilibrium. This phenomenon is called quenching. Particles will freeze and form metastable amorphous structures. There is similarity between a solid and an optimization problem. Both of the cases, there are many degrees of freedom, which are the positions of particles in a solid and the configuration in an optimization problem. Both cases also have some global quantities that have to be minimized, that are the energy of the solid and the cost function in optimization respectively.

2.2.4 Monte Carlo Method (Metropolis Algorithm)

Given a current state of a solid, characterized by the positions of its particles, a randomly generated perturbation is applied. This corresponds to a small displacement of a randomly chosen particle. If the perturbation results in a lower energy state, the process is continued to a new state. If $\Delta E \geq 0$, the probability of acceptance of the

perturbed state is given by $\exp\left(\frac{-E}{K_B T}\right)$. This rule of accepting new states is referred to as the Metropolis criterion. The solid will eventually reaches thermal equilibrium after a large number of perturbations. The probability distribution of states will approach the Boltzmann Distribution. This Monte Carlo method is known as Metropolis algorithm.

The Metropolis algorithm can be used to generate configurations of optimization problem. The configuration assumes the role of the states of a solid while the control parameter, T_k and the cost function, F assume the roles of the temperature and energy respectively.

Simulated annealing can be shown as a sequence of Metropolis algorithms evaluated at decreasing values of control parameter. The control parameter is given a large value initially and a sequence of trials is generated. In each trial, a configuration j is generated by choosing at random an element from the neighborhood of the current configuration i. This corresponds to the small perturbation in Metropolis algorithm. Let

$$\Delta F_{ij} = F(j) - F(i) \qquad (2.2)$$

The probability of configuration j being the next configuration in the sequence is 1, if $\Delta F_{ij} < 0$. And, the probability is $\exp\left(\frac{-F(i)}{T_k}\right)$, if $\Delta F_{ij} > 0$. This is the Metropolis criterion. Thus, there is a non-zero probability of continuing with a configuration with higher cost than the current configuration. This sequence of trials is continued until equilibrium is reached, that is when the probability distribution of the configurations approaches the Boltzmann distribution,

$$\Pr(configuration = i) = q_i(T_k) = \frac{1}{Q(T_k)} \exp\left(\frac{-F(i)}{T_k}\right) \qquad (2.3)$$

where $Q(T_k)$ is a normalization constant depending on the control parameter T_k. The value of control parameter is reduced in steps until it approaches zero. However, the system is still allowed to reach equilibrium at each step by generating a sequence of trial as discussed previously. After termination, the final configuration is taken as the solution to the problem.

2.2.5 Simulated Annealing (SA) - The Algorithm

After looking at the statistical viewpoint of the annealing process, we now look at how these are implemented in computer programs in order to perform some optimization process. Figure 4 depicts the flow chart of the algorithm.

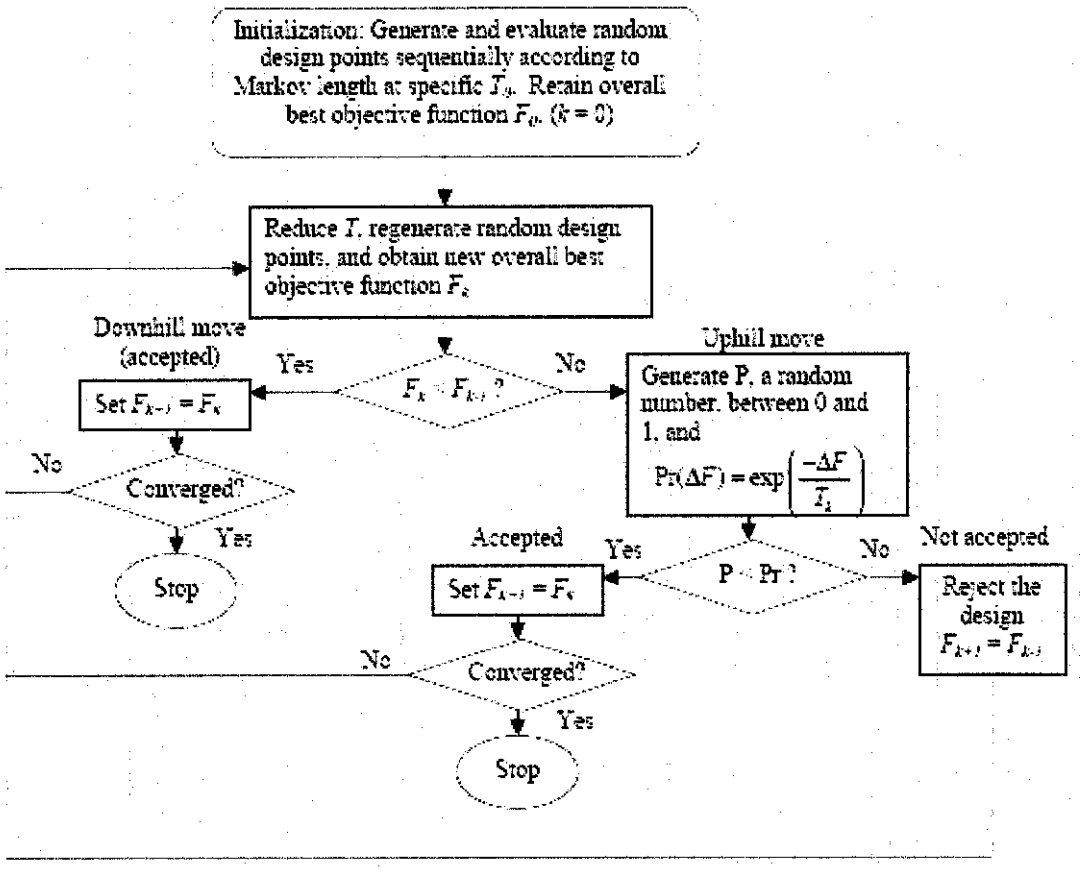


Figure 4: Flowchart of a Simulated Annealing Algorithm

In the algorithm, the following steps should be carried out:

Step One:

An initial temperature, T_o is set. Then, at T_o, a set of random points is generated sequentially, and the number of points generated is according to the length of the Markov chain, m. Markov chain is a string of "bits" which represent the cost functions of each random point.

Step Two:

Each point is evaluated and the point that gives the overall best objective function value for that temperature is set to be the objective function value ${}_k F$ for the particular iteration of temperature ($k = 0$).

Step Three:

At the next iteration ($k = 1$), the temperature is reduced according to a defined reduction ratio, and another set of random points are again generated sequentially, and evaluated.

Step Four:

The overall best objective function value in the Markov chain, F_k is compared to the objective function value of the previous temperature, F_{k-1}.

Step Five:

If F_k is less that F_{k-1}, then F_k replaces F_{k-1} as the new current minimum. If F_k is greater than F_{k-1}, which is an uphill move, a probabilistic selection process is used to determine the move should be accepted or not by determining Pr. Pr is defined as:

$$\Pr(\Delta F) = \exp\left(-\frac{\Delta F}{T_k}\right) \qquad (2.4)$$

where ΔF is the difference in the objective function. A random number, between 0 and 1, P is generated here. If $P < Pr$, then the design point of an uphill move is accepted to be the new minimum. If $P > Pr$, the design point will be rejected and remain the

original design point. This is where the Boltzman Distribution and Monte Carlo Algorithm applied in the process.

Step Six:

This process is repeated until convergence is reached, or if the maximum number if iterations is reached. The convergence check can be done in several ways. One of them is to determine if the objective function value does not change significantly after several iterations. Another way is to see if the objective function value does not change significantly within a Markov chain.

2.2.6 The Control Parameters

In SA, there are a few control parameters that require user to define. The control parameters are: initial temperature, scaling factor of temperature at each iteration, maximum number of iterations, scaling factor of neighborhood size, and the length of Markov chain. The control parameters that can be input are:

1. **Initial Temperature:** Initialize the temperature of the first iteration
2. **Scaling Factor for Temperature:** Factor of decrement of temperature at each iteration. The value has to be in between 0 and 1.
3. **Maximum Number of iterations:** User can set the maximum number of iteration, so that it will not take a long time to converge. Although the test problems were tested to converge in less than 200 iterations, this option is useful when the user wishes the optimization process to stop at a specified number of iterations. The maximum number of iterations is set to be 600.
4. **Scaling Factor of Neighborhood Size:** User can set the factor of decrement of the boundary of the design points generated during each iteration.
5. **Length of Markov Chain:** Specifying the number of values generated in a Markov chain.

CHAPTER 3
BRUSHLESS DC MOTOR DESIGN

3.1 GEOMETRICAL PROPERTIES: DIMENSIONS OF THE MOTORS

The geometrical parameters of the different motor configurations are presented in this section.

3.1.1 Inner-rotor Brushless DC (BLDC) motor

For these motors, the permanent magnets are placed on the rotor surface, as shown in Figure 5. This is the most commonly used configuration [6]. The main advantage is its simplicity and consequently its lower construction cost compared to other PM machines. The main drawback is the exposition of the permanent magnets to demagnetisation fields. Furthermore, the magnets are subject to centrifugal forces that can cause their detachment from the rotor. However, as these forces increase with the rotational speed, they are low in the studied low-speed applications.

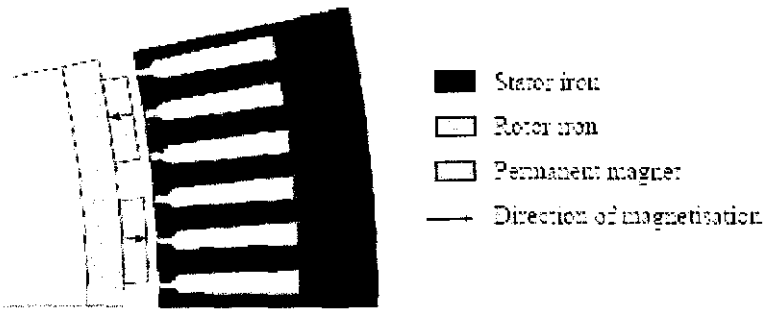


Figure 5: Cross Section of BLDC motor with inner rotor

Figure 6 shows the geometry of a BLDC motor [6] including the parameters of the geometrical dimensions. These dimensions are expressed in equations (3.1) to (3.6), where Q_s is the number of stator slots. The parameter k_{open} is the ratio of the stator slot opening to the slot width (3.6). The teeth are straight, which means that the tooth width b_{ts} is constant all along the tooth.

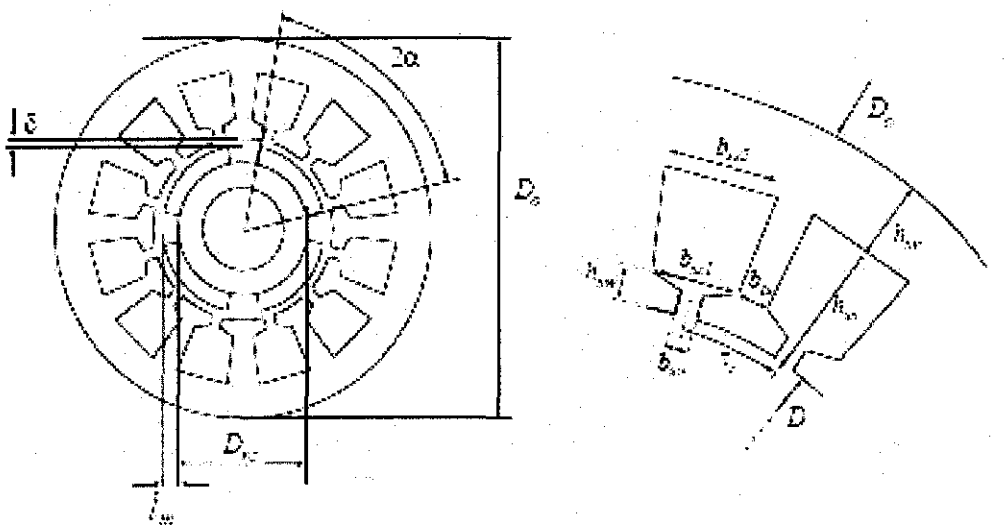


Figure 6: Geometrical parameters of the inner rotor BLDC motor

$$D = D_{rc} + 2l_m + 2\delta \qquad (3.1)$$

$$\tau_s = \pi \frac{D}{Qs} \qquad (3.2)$$

$$b_{ss1} = \pi \frac{D + 2h_{sw}}{Q_s} - b_{ts} \qquad (3.3)$$

$$h_{sy} = \frac{1}{2}\left(D_o - D - 2h_{ss}\right) \qquad (3.4)$$

$$b_{ss2} = \pi \frac{D + 2h_{ss}}{Q_s} - b_{ts} \qquad (3.5)$$

$$k_{open} = b_{so} / b_{ss1} \qquad (3.6)$$

The inner stator diameter D is very large compared to the slot pitch τ_s. Therefore, b_{ss1}, b_{ss2} and b_{ts}, which actually are arcs of circle, are approximated as straight lines in equations (3.3) and (3.6). The slot area A_{sl} is given by equation (3.7).

$$A_{sl} = \frac{1}{2}(b_{ss1} + b_{ss2}) * (h_{ss} - h_{sw}) \qquad (3.7)$$

According to the equations, the two-dimensional geometrical structure of the stator can be described entirely with the following parameters: D_{rc}, l_m, δ, h_{sw}, b_{ts}, D_o, h_{ss}, k_{open} and Q_s. Adding the number of poles p, the half pole angle α and the active length L, the whole three-dimensional geometry can be described (without considering the end-windings).

3.2 MAGNETIC PROPERTIES

For this project, the permanent magnet used is Neodymium Iron Boron (Nd-Fe-B).

3.2.1 Analytical calculation of the flux density in the air gap

The amplitude of the fundamental air gap flux density $B\hat{}_{\delta}$, has to be calculated with accuracy since the design procedure relies on it. For the BLDC motor design, the air gap flux density is assumed to have a rectangular shape as wide as the permanent magnet width and a maximum value B_m.

B_m is calculated as follows:

$$B_m = \frac{B_r k_{leak}}{1 + \frac{\mu_r \delta k_c}{l_m}} \qquad (3.8)$$

where Br is the remanence flux density of the magnet, μ_r the relative magnet permeability and k_C the Carter factor.

$$k_c = \frac{\tau_s}{\tau_s - \frac{(k_{open} b_{ss1})^2}{b_{ss1} k_{open} + 5\delta}} \qquad (3.9)$$

The factor k_{leak} *takes* the magnetic leakage between two neighboring permanent magnets into account.

3.2.2 Analytical calculation of the flux density in the teeth

It is important to obtain a correct analytical value of the flux density created by the permanent magnets in the teeth. If the value turns out to be higher than expected, the teeth can be saturated which means a high magnetic leakage.

The flux density in the teeth B_{ts} is calculated from the permanent magnet flux passing through the air gap, the width of a tooth b_{ts}. The number of teeth through which the flux is passing (2 for q = 1). Equation (3.10) gives the flux density in a tooth for a BLDC motor. The factor $k_{leaktooth}$ is used to take into account the part of leakage flux, passing through the tooth shoe only.

$$B_{ts} = \frac{B_m 2\alpha \frac{2}{p}\left(\frac{D}{2} - \delta\right) * \left(1 - k_{leaktooth}\right)}{2b_{ts}} \tag{3.10}$$

$$k_{leaktooth} = \frac{17p/56 - 13/14}{100} \tag{3.11}$$

3.3 ELECTRICAL PROPERTIES

3.3.1 Inductance

For a non-salient synchronous or BLDC motor the d- and q- axis synchronous inductances are equal and

$$L_d = L_q = L_l + L_{md} = L_l + L_{mq} \qquad (3.12)$$

where L_l is the leakage inductance and L_{md} and L_{mq} are the d- and q- axis magnetizing inductances respectively.

$$L_l = pqn_s^2 L\mu_o\lambda_1 \qquad (3.13)$$

$$L_{md} = \frac{3}{\pi}(qn_s k_{w1})^2 \frac{\mu_o}{\delta k_c + \frac{l_m}{\mu_r}}(D-\delta)L \qquad (3.14)$$

where λ_1 is the specific permeance coefficient of the slot opening and depends on the slot geometry , q is the number of slots per pole per phase and k_{w1} is the fundamental winding factor. If the number of slots per pole per phase is equal to 1, then the winding factor k_{w1} is 1.

3.3.2 Resistance of one phase of the stator winding

It is assumed that all the coils in one phase are coupled in series. The phase resistance is calculated as:

$$R = p_{cu} \frac{(pL + (D + h_{ss})\pi kcoil) n_s^2 q}{f_s A_{sl}} \qquad (3.15)$$

where f_s is the slot fill factor, p_{cu} is copper resistivity and k_{coil} is coil factor.

3.3.3 Induced voltage

The induced phase voltage E is deduced from Faraday's law:

$$E(t) = Ns \frac{d\phi m}{dt} \qquad (3.16)$$

The maximum fundamental of the magnet flux $\hat{}\varphi m$ linked to one turn of the coil is:

$$\hat{\phi}_m = \frac{2}{\pi} \hat{B}_\delta L(D - \delta) \frac{\pi}{p} \qquad (3.17)$$

Ns is the number of turns per phase which are in series:

$$N_s = \frac{p}{2} q n_s \qquad (3.18)$$

Finally the rms-value of the induced voltage can be calculated as:

$$E = \frac{1}{\sqrt{2}} \omega k_{w1} q n_s \hat{B}_\delta L(D - \delta) \qquad (3.19)$$

3.3.4 Current loading

The peak value of the fundamental current loading $\hat{S}_1$ is calculated from the torque equation:

$$\hat{S}_1 = \frac{4T}{\pi(D-\delta)2L\hat{B}_\delta k_{w1} k_{cor} \sin\beta} \qquad (3.20)$$

where β is $\pi/2$ for BLDC motor, k_{cor} is correction factor and T is rated torque.

The correction factor k_{cor} is used to compensate the losses and leakages that are not analytically calculated, such as the flux leakage through the slots.

3.3.5 Ampere turns and current density

From the fundamental peak current loading and the ampere-turns (3.21), the current density J can be calculated.

$$n_s\hat{I} = \hat{S}_1\tau_s \qquad (3.21)$$

$$J = \frac{n_s I}{A_{sl}} \qquad (3.22)$$

3.3.6 Number of conductors per slot

The number of conductors per slot can be calculated as :

$$n_s = \frac{2E_{max}}{pqk_{w1}\hat{B}_\delta LD\omega} \qquad (3.23)$$

Where E_{max} is a maximum emf voltage and ω is a motor speed in radian per second.

3.4 PERFORMANCE

To compute the efficiency, it is necessary to compute the winding copper losses and iron losses. Of these, the core or iron loss is the most difficult to compute accurately. The permanent magnet and rotor back iron experience little variation in flux and therefore do not generate significant core loss. On the other hand, the stator teeth and stator back iron experience flux reversal on the order of B_{max} (maximum steel flux density) at the fundamental electrical frequency. With knowledge of B_{max} and electrical frequency f_e, the core loss of the stator can be roughly approximated.

3.4.1 Winding Copper Losses

The winding copper losses can be calculated as:

$$Pcopper = J^2 A_{sl} p_{cu} L_{endw} Q_s \qquad (3.24)$$

Where p_{cu} is the copper resistivity , Q_s is number of slots and L_{endw} is the average end winding length.

3.4.2 Core or Iron Losses

The iron losses can be calculated as:

$$P_{iron} = \rho_{bi} V_{st} \Gamma(B_{max}, f_e) \qquad (3.25)$$

Where ρ_{bi} , kg/m^3, is the mass density of the back iron, V_{st} is the stator volume, and $\Gamma(B_{max}, f_e)$, W/kg, is the core loss density of the stator material at the flux density B_{max} and frequency f_e.

The efficiency of the BLDC motor producing torque at rated speed is:

$$\eta = \frac{T\omega}{T\omega + Pcopper + Piron + Ps} x100\% \qquad (3.26)$$

Where P_s is consists of the stray loss, composed of windage, friction and other loss components. Depending on motor speed and construction, P_s typically decreases the efficiency on the order of several percent. If desired, the loss incurred in driving the motor can be included in (3.26), giving more realistic total system efficiency.

3.5 COST, VOLUME AND WEIGHT OF ACTIVE MATERIAL

The active material of BLDC motor consists of wire winding copper, stator lamination and permanent magnets.

3.5.1 Cost of active material

Table 1 shows the summary of material cost of winding copper per 250 gram which will be based on Standard Wire Gauge (SWG). Besides, the summary of stator lamination (stainless steel) cost also reported at Table 2.

Table 1: Winding Copper Cost per 250 gram

DIAMETER (mm)	SWG	PRICE(RM)
1.6	16	30.64
1.25	18	30.64
0.9	20	30.64
0.71	22	30.64

Table 2: Cost of stator lamination and permanent magnets (Neodymium iron boron) per kg

MATERIAL	PRICE(RM)/kg
Stator Lamination (Stainless Steel)	7.70
Neodymium Iron Boron	77.00

3.5.2 Weight and volume of active material

Before considering total weight and volume of active material, it is interesting to consider density of each material. The density of each material is reported at table 3.

Table 3: Density of active material

DESCRIPTION	Density
Magnet density	7500 *kg/m3*
Stator lamination density	7850 *kg/m3*
Copper density	8920 *kg/m3*

3.6 OPTIMIZATION PROCEDURE

3.6.1 Objective function and augmented objective function

The main focus in this report is to derive optimal design parameters minimizing a single objective function such as volume, weight or cost of the active material BLDC motor and a multi-objective function of the above [9]. The cost function for the BLDC motor is expressed as:

$$F_c(X) = C_1 + C_2 + C_3 \qquad (3.27)$$

Where $X = (x_1, x_2 \ldots\ldots\ldots x_n)$, vector of design variables; C_1,C_2 and C_3 – cost of winding copper, stator/rotor lamination (stainless steel) and permanent magnets(Neodymium Iron Boron). Similarly the weight and volume functions are defined as:

$$F_w(X) = G_1 + G_2 + G_3 \qquad (3.28)$$

$$F_v(X) = V_1 + V_2 + V_3 \qquad (3.29)$$

Where G_1,G_2 and G_3- weight of winding copper, stator lamination and permanent magnets. Besides, V_1,V_2 and V_3 represent volume of winding copper, stator lamination and permanent magnets. The multi-objective function for this project is:

$$F(X) = F_c(X) + F_w(X) + F_v(X) \qquad (3.30)$$

Where total $F(X)$ is approximately to be 1. So when doing comparison, as an example GA gives 0.967 and SA gives 0.98, the GA will be the optimum value.

In addition to the constraints the design variables are constrained with upper and lower bounds to satisfy performance restrictions. The BLDC mathematical model posed as a nonlinear programming problem is stated as:

Find x such that $F(X)$ is minimum, subject to $g_j(X)\{\leq=\}0, j=1,2,....m$ with $X \geq 0$ being a non-negative solution. $F(X)$ and $g_j(X)$ are the nonlinear objective and constraint functions. Using exterior penalty function method, an augmented objective function, P is formulated as:

$$P(X,r) = F(X) + r\sum_{j=1}^{m}\left[g_j(X)\right]^2, r \geq 0 \qquad (3.31)$$

where r is the penalty factor and set to be 1000 for both GA and SA techniques.

In GA method, each design variable is coded as a 16 bit binary string. The objective and constraint functions are developed in terms of the specified design variables.

3.6.2 The design procedure of BLDC motor

The geometry of a BLDC motor is completely described with 12 design parameters. The optimal design parameters are obtained by solving an optimization problem, applying Genetic Algorithm (GA) and Simulated Annealing (SA). The goal of the optimization is to minimize a single or the multi objective function of the motor and fulfill the requirements and the constraints that guarantee the required mechanical, thermal, and magnetic behaviors. The objective function along with the variables and constraints are explained in the following sections.

Table 4: Rated Motor Data

DESCRIPTION	VALUE
Phase	3
Rated speed	7500 rpm (785 rad/s)
Motor Rated torque	0.4Nm
Rated voltage	24 V
Rated power	314 W
Remanence flux density	1.08T
Relative permeability	1.03
Iron Stacking factor	1
Stator slot fill factor *fs*	0.45
Magnet density	7500 *kg/m*3
Iron density	7750 *kg/m*3
Copper density	8920 *kg/m*3
Copper resistivity at 20 ◦ C	$1.72e-8\ \Omega$/m
Flux density in the rotor yoke	1.4T

Table 5: Constraints Motor Data.

DESCRIPTION	CONSTRAINT
Stator yoke height at least half the slot height	$h_{sy} \geq h_{ss}/2$
Slot width between 0.15 and 0.5 times slot height	$0.15h_{ss} \leq \tau_s - b_{ts} \leq 0.5h_{ss}$
Tooth width at least 30% of the slot pitch	$b_{ts} \geq 0.3\tau_s$
Slot opening width at least 2 mm	$b_{ss1}k_{open} \geq 2mm$
Flux density in stator teeth under 1.6T	$B_{ts} \leq 1.6T$
Efficiency more than 90 percent	$\eta \geq 90\%$
Weight Material less than 0.5 kg	TotalWeight $\leq 0.5kg$

Table 6: Design variables with lower and upper limits.

Variables	Description	Lower Limit	Upper Limit	Unit
X1	Number of poles	4	8	-
X2	Slot per pole per phase	0.5	1	-
X3	Rotor Diameter	20	25	mm
X4	Air Gap Length	0.1	0.5	mm
X5	Magnet Thickness	1	2	mm
X6	Half Pole Angle	0.88	1.1	radian
X7	Outer Stator Diameter	50	55	mm
X8	Stator Tooth Width	2.5	3	mm
X9	Stator Slot Height	8	10	mm
X10	Open Factor	0.45	0.5	-
X11	Slot Wedge Height	2	3	mm
X12	Motor Length	40	50	mm

Some of the combinations of the above variables can yield to unacceptable design; in fact they can generate some geometrical discrepancies. A case is when the gap and permanent magnet length are not compatible with the rotor space fixed by the inner stator diameter. In the design procedure, a control on geometrical feasibility is introduced. If it is not passed, the control produces a penalty effect on the objective function. A single and multi objective function already reported on section 3.6.2.

CHAPTER 4

METHODOLOGY

4.1 Procedure Identification

The procedure involved for the project design development is shown in Figure 7:

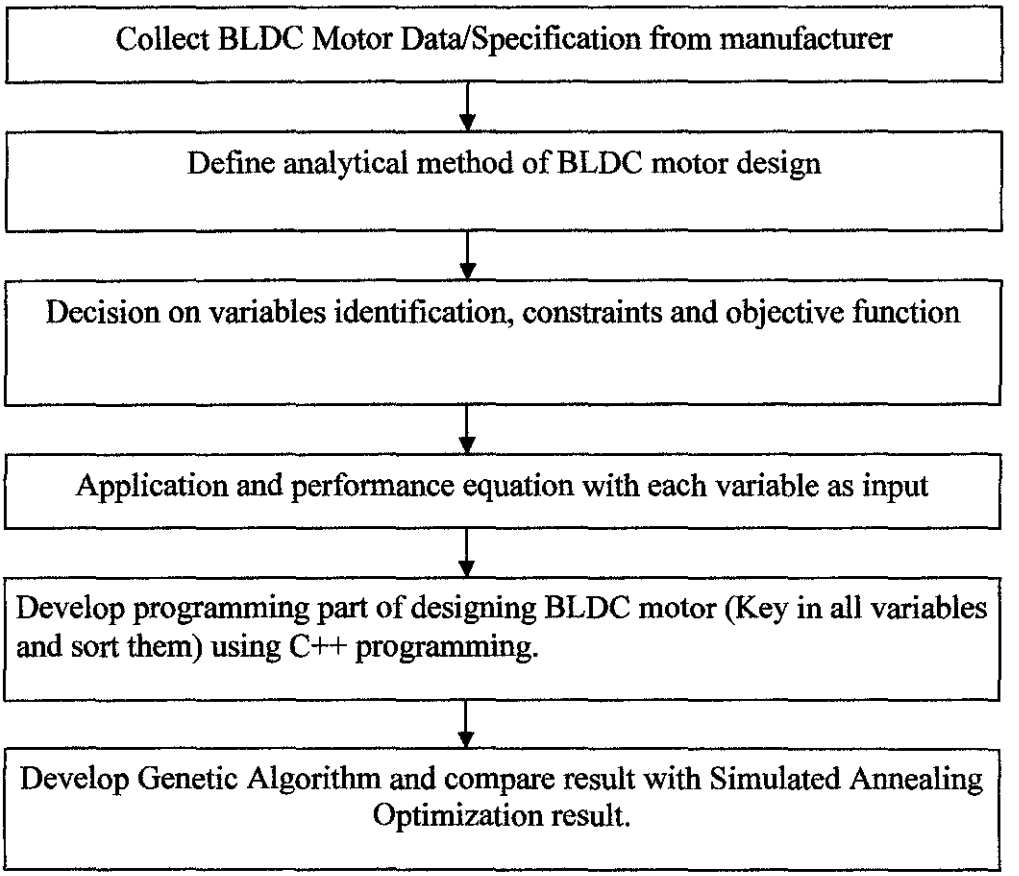


Figure 7: Procedures for Optimization of BLDC motor.

4.2 Tools required

- Microsoft Visual Studio C++ version 6.0
- Borland C++ version 5.02

CHAPTER 5
RESULT AND DISCUSSION

5.1 OPTIMIZATION BY GENETIC ALGORITHM (GA) TECHNIQUE

The genetic algorithm technique has been tested by several simulations: number of different generations (N=50 until N=600) have been investigated. The size of the generations greatly affects the quality of the result and computation time. In addition, different probabilities of crossover p_c and mutation p_m have been tested. The crossover operator searches in new parts of solution space: a low pc forbids a profitable search, while a high pc includes in the new population a lot of successive generations. The algorithm has been tested with $p_c = 0.25$, $p_c = 0.5$ and $p_c = 0.75$.

The mutation operator explores new zones out of the solution spaces. Value of $p_m = 0.01$ has been examined. For all simulations, 600 generations have been considered. With this value a good convergence has been obtained. The result, reported in Table 7, confirm the robustness of the algorithm.

Table 7: Best F(x) with fixed number of generations, p_c and p_m

Generations	**Crossover probability p_c**	**Mutation probability p_m**	**Best F(x)**
50	0.25	0.01	0.962165
	0.50		0.962163
	0.75		0.962168
100	0.25	0.01	0.962163
	0.50		0.962163
	0.75		0.962167
200	0.25	0.01	0.962163
	0.50		0.962163
	0.75		0.962167
300	0.25	0.01	0.962163
	0.50		0.962163
	0.75		0.962167
400	0.25	0.01	0.962163
	0.50		0.962163
	0.75		0.962165
500	0.25	0.01	0.962163
	0.50		0.962163
	0.75		0.962164
600	0.25	0.01	0.962163
	0.50		**0.962162**
	0.75		**0.962162**

Table 7 shows that the lower value of multi objective function is reach with number of generations = 600, p_c = 0.5 and 0.75 and p_m = 0.01. The optimal solution is achieved after 600 generations.

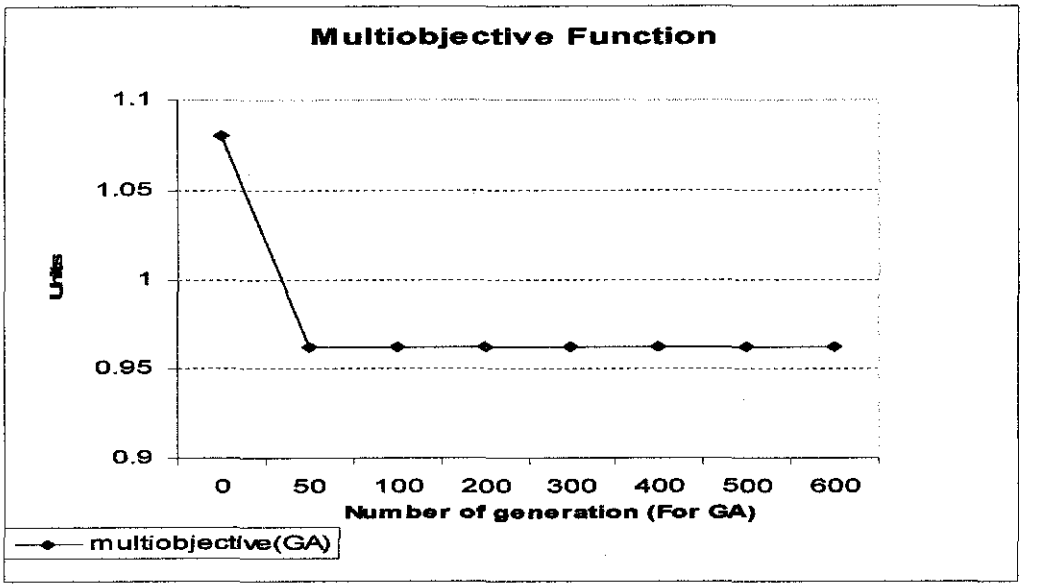


Figure 8: Multi objective optimization using Genetic Algorithm

From Figure 8, as the number of generation increase, the value of multi objective function will be constantly optimum. The final value from GA is **0.962162.**

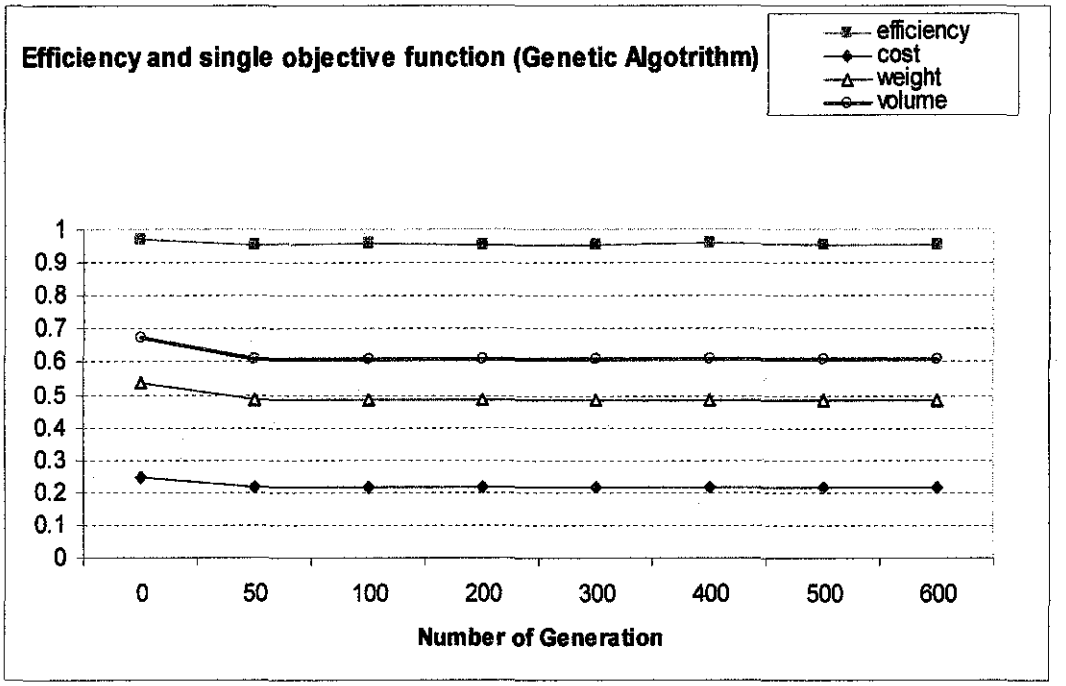


Figure 9: Efficiency, Cost, Weight and Volume of materials trend during GA

From Figure 9, as the number of generations increased until 600 generations, the values converge to optimum. The final values obtained from the graph are 95.3% for efficiency, RM 21.91 for material cost, 0.486 kg for material weight and 60600 mm^3 for material volume.

Table 8 below shows the final result of each design variables obtained from GA optimization.

Table 8: Design variables from GA optimization

Variables	**Description**	**Value GA**	**Unit**
X1	Number of poles	4	-
X2	Slot per pole per phase	1	-
X3	Rotor Diameter	20	mm
X4	Air Gap Length	0.1	mm
X5	Magnet Thickness	1	mm
X6	Half Pole Angle	0.88	radian
X7	Outer Stator Diameter	50	mm
X8	Stator Tooth Width	2.984	mm
X9	Stator Slot Height	8	mm
X10	Open Factor	0.475	-
X11	Slot Wedge Height	2.635	mm
X12	Motor Length	40	mm

5.2 OPTIMIZATION BY SIMULATED ANNEALING (SA) TECHNIQUE

With simulated annealing, some input parameters must be considered which are:

Initial Temperature, Tk = 125^0C.

Scaling factor temperature, Tk scale = 0.75.

Maximum number of iteration, Max Iter = 300.

Scaling factor for neighbourhood size, NS scale = 0.95.

Length of markov chain, Markov Lg = 80.

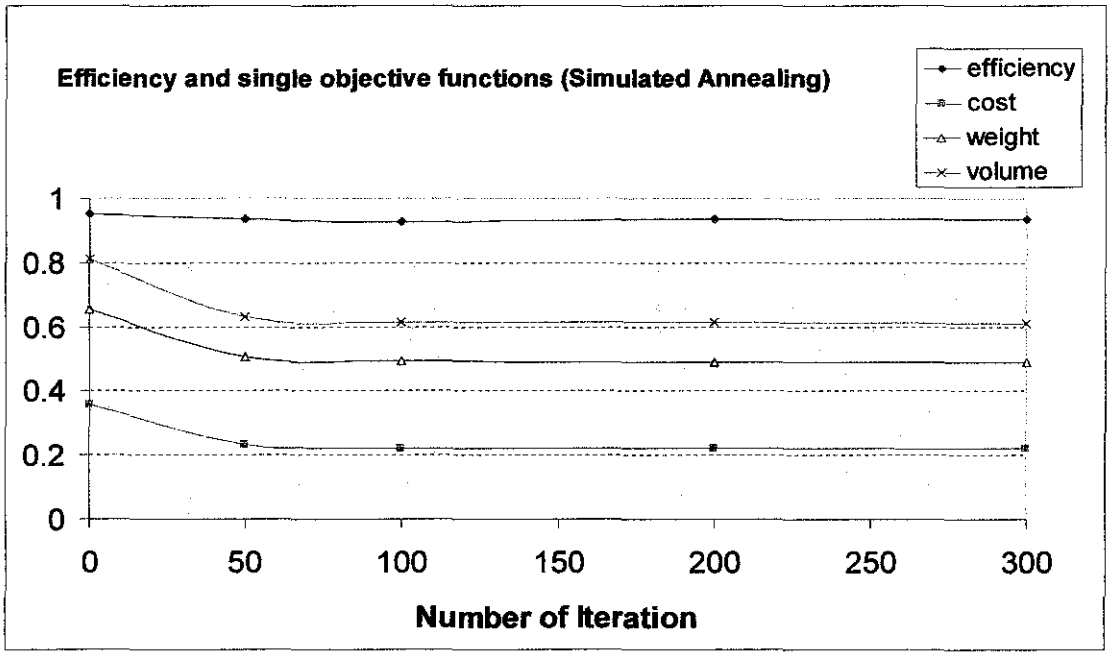


Figure 10: Efficiency, Cost, Weight and Volume of materials trend during SA

From Figure 10, as the number of iterations increased until 300 iterations, the value will be constantly optimum. The final values obtained from the Figure 10 are 93.4 % for efficiency, RM 22.05 for material cost, 0.492 kg for material weight and 61382 mm^3 for material volume.

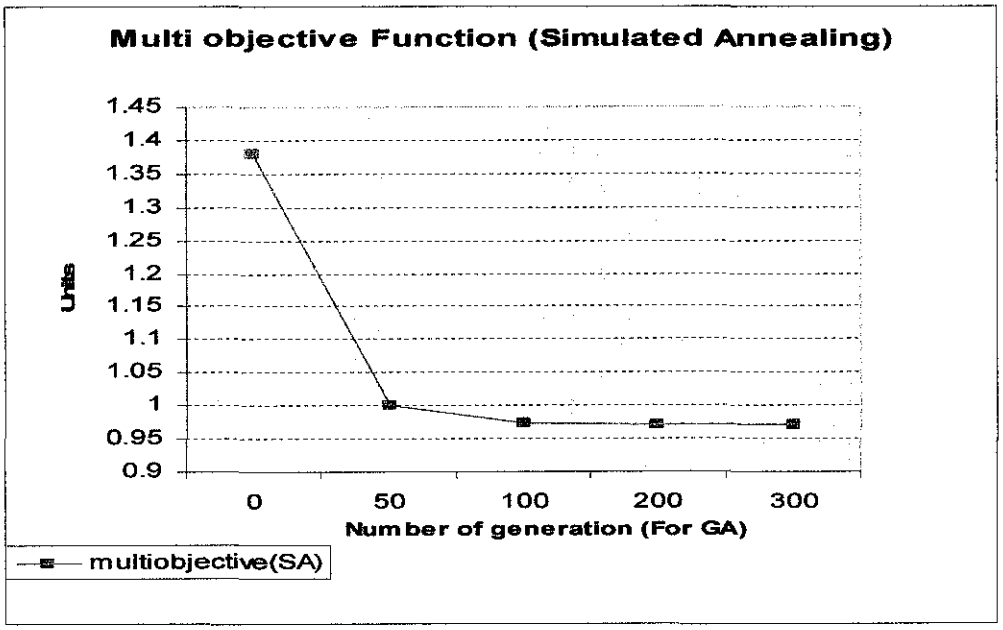


Figure 11: Multi Objective Function optimization for Simulated Annealing (SA)

From Figure 11, as the number of iterations for SA increased the value of multi objective function converges to optimum value. The final value from SA is 0.971 which is more than GA value.

Table 9: Design variables result from SA

Variables	Description	Value SA	Unit
X1	Number of poles	4	-
X2	Slot per pole per phase	1	-
X3	Rotor Diameter	20	mm
X4	Air Gap Length	0.4	mm
X5	Magnet Thickness	1.028	mm
X6	Half Pole Angle	0.90	radian
X7	Outer Stator Diameter	50	mm
X8	Stator Tooth Width	2.597	mm
X9	Stator Slot Height	8.02	mm
X10	Open Factor	0.46	-
X11	Slot Wedge Height	3	mm
X12	Motor Length	40	mm

5.3 RESULT COMPARISON BETWEEN GENETIC ALGORITHM AND SIMULATED ANNEALING

Table 10 shows the comparison of optimized motor data between Genetic Algorithm (GA) and Simulated Annealing (SA).

Table 10: Optimized Motor Data

	Unit	SA	GA
Bore Diameter	mm	22.4	22
Length of core	mm	40	40
Air gap length	mm	0.2	0.1
Magnet thickness	mm	1	1
External Diameter	mm	50	50
Electrical Load	A/m	29851	30122
Number of slots		12	12
Number of poles		4	4
Air gap flux density	Tesla	0.911	0.911
Rotor yoke flux density	Tesla	1.4	1.4
Tooth flux density	Tesla	1.57	1.6
Current density	A/mm^2	8.626	7.810
Magnet Weight	Kg	0.0456	0.049
Core Weight	Kg	0.311	0.298
Copper Weight	Kg	0.135	0.139
Total Material Weight	Kg	0.492	0.486
Volume of material	mm^3	61362	60600
Iron losses	Watt	1.95	1.90
Copper losses	Watt	15.91	14.29
Stray losses	Watt	3.14	3.14
Winding Temperature	Celcius	61.4	55.5
Efficiency	%	93.4	95.3
Material Cost	RM	22.05	21.91
Multi objective		0.971	0.962
Execution Time	Second	5.250	7.340

The solution achieved by GA is slightly often more advantageous than that reached by SA method. The multi objective function value for Genetic Algorithm is more optimally minimum (0.962) than Simulated Annealing (0.971). Execution time of GA is slower than SA since GA is dealing with higher number of generation and also population in order to find most optimum value.

CHAPTER 6
CONCLUSION AND RECOMMENDATION

This project report describes the optimal design of BLDC motor using GA and SA. The methodology covers the selection of suitable design variables, constraints and objective functions. All the variables are determined from the configuration of BLDC motor and equations related are derived as an input during programming part. The optimal design of the BLDC motor is obtained by GA and SA techniques. The results obtained by GA technique are compared with SA which also used C/C++ coding. From the simulation results, it is observed that GA performed better than SA for BLDC motor design.

It is recommended to compare GA with other optimization techniques such as Tabu Search, Neural Network and so on for future works. Different theoretical designs have been found to be very promising for the application. Thus, the future work could concentrate further on one particular design for the purpose of building a prototype.The constraints applied during the design procedure should be improved, notably the constraints on the rigidity of the structure. They have to be adapted to the application. If a prototype is built, a lot of work would be required for designing the mechanical elements, testing the prototype and verifying if the machine fulfills the expectations.

REFERENCE

[1] Freescale Semiconductor, *Brushless DC motor.* Retrieved from http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=02nQXGrrlPY7r8hv0V

[2] Lance D.Chambers (1999). Genetic Algorithm .*Complex Coding Systems.* Volume 3: CRC Press.

[3] Jung Leng Foo, Chia Ken Leong, Chin Pei Tang, "A Web-Based Simulated Annealing Application Tool ", Final Project 2002, State University of New York at Buffalo.

[4] Ki-Jin Han, Han-Sam Cho, Dong-Hyeok Cho (2000). Optimal Core Shape Design for Cogging Torque Reduction of Brushless DC motor using GA. Retrieve from http://ieeexplore.ieee.org/iel5/20/19005/00877824.pdf?arnumber=877824&htry=2

[5] D.T.Pham and D.Karaboga (1998], Intelligent Optimisation Techniques: Springer.

[6] F.Libert, J.Soulard "Design study of a Direct-Driven Surface Mounted Permanent Magnet Motor for Low Speed Application", Doctoral Thesis, Royal Institute of Technology, Sweden, 2004.

[7N.Bianchi, S.bolognani, "Design optimization of electric motors by genetic algorithm", IEE Proceeding Electric Power Application, vol. 145, no. 6, September 1998.

[8] Th. Koch, A. Binder, "Permanent Magnet Machines with Fractional Slot Winding for Electric Traction", Proceedings of ICEM 2002, Brugge, Belgien.

[9] Associate Professor Dr. K.S Rama Rao, UTP, 2007, Private communication.

APPENDICES

APPENDIX A: GLOSSARY OF SYMBOLS

APPENDIX B: EQUATIONS FOR COST, WEIGHT AND VOLUME OF ACTIVE MATERIALS.

APPENDIX C: C CODE FOR BLDC MOTOR EQUATIONS FROM GENETIC ALGORITHM

APPENDIX D: C++ CODE FOR BLDC MOTOR EQUATIONS FROM SIMULATED ANNEALING

APPENDIX A

GLOSSARY OF SYMBOLS

A_{sl}	Slot area	[m^2]
$\hat{B}_{\delta}$	Amplitude of the fundamental air gap flux density	[T]
B_m	Maximum Value of the air gap flux density	[T]
B_r	Remanence flux density of the permanent magnets	[T]
B_{ry}	Flux density in the rotor yoke	[T]
b_{so}	Stator slot opening	[m]
b_{ss1}	Inner stator slot width	[m]
b_{ss2}	Outer stator slot width	[m]
b_{ts}	Stator tooth width	[m]
B_{ts}	Flux density in stator tooth	[m]
D	Inner stator diameter	[m]
D_{rc}	Outer rotor diameter	[m]
D_i	Inner rotor diameter	[m]
D_o	Outer stator diameter	[m]
E	Fundamental of the induced voltage RMS	[V]
f_s	Slot fill factor	-
hry	Rotor yoke height	[m]
hss	Stator slot height	[m]
hsw	Slot wedge height	[m]
I	Line current	[A]
kc	Carter factor	-
$kcoil$	End-winding coefficient	-
k_{cor}	Correction factor for current loading calculation	-
$k_{leaktooth}$	Correction factor for flux density in teeth	-
k_{open}	Ratio of the slot opening over the slot width	-
k_{w1}	Fundamental winding factor	-
L	Active length	[m]

l_m	Length of permanent magnet	[m]
J	current density	[A/m^2]
N_s	number of turns per phase	-
n_s	number of conductors per slot	-
p	number of poles	-
q	number of slots per pole per phase	-
Q_s	number of stator slots	-
R	resistance of one phase of the stator winding	Ω
S_1	current loading	[A/m]
T	torque	[Nm]
α	half pole angle	[rad]
δ	air gap length	[m]

APPENDIX B

EQUATIONS FOR COST, WEIGHT AND VOLUME OF ACTIVE MATERIALS.

WINDING COPPER

Volume

$$V_{copper} = \pi f_s h_{ss} \left(\frac{D}{2} + h_{ss}\right)\left(L + \frac{2\pi \frac{D}{2} K_{leak}}{2p}\right)$$

Weight

$$G_{copper} = V_{copper} \rho_{copper}$$

Cost

$$C_{copper} = G_{copper} * Cost\ of\ winding\ copper\ per\ kg$$

PERMANENT MAGNET (NEODYMIUM IRON BORON)

Volume

$$V_{magnet} = 2l_m(D_{rc} + l_m)(\alpha)Lp$$

Weight

$$G_{magnet} = V_{magnet} \rho_{magnet}$$

Cost

$$Cmagnet = G_{magnet} * Cost\ of\ permanent\ magnets\ per\ kg$$

STAINLESS STEEL (STATOR/ROTOR LAMINATION AND SHAFT)

Volume shaft

$$V_{shaft} = \frac{\pi D_i^2 L}{4}$$

Weight Shaft

$$G_{shaft} = V_{shaft} \rho_{stainless\ steel}$$

Cost Shaft

$$C_{shaft} = G_{shaft} * Cost\ of\ stainless\ steel\ per\ kg$$

Volume Stator Lamination

$$V_{stator} = L(\pi \frac{D}{2} h_{ss} + \pi h_{sy}(\frac{D}{2} + h_{ss}))$$

Weight Stator Lamination

$$G_{stator} = V_{stator} \rho_{stainless\ steel}$$

Cost Stator Lamination

$$C_{stator} = G_{stator} * Cost\ of\ stainless\ steel\ per\ kg$$

Volume rotor stacking

$$V_{rotor} = \frac{\pi(D_{rc}^2 - D_i^2)L}{4}$$

Weight rotor stacking

$$G_{rotor} = V_{rotor} \rho_{stainless\ steel}$$

Cost rotor stacking

$$C_{rotor} = G_{rotor} * Cost\ of\ stainless\ steel\ per\ kg$$

APPENDIX C

C CODE FOR BLDC MOTOR EQUATIONS FROM GENETIC ALGORITHM

```
/*==========================================================================
OBJECTIVE FUNCTION ( Supposed to be minimized) :
Change it for different applications
==========================================================================*/
void objective(indv)
INDIVIDUAL *indv;
{
int i;
double g[MAXCONSTR], gsum, x[2*MAXVECSIZE],penalty_coef=1000;
int Nph=3,Qs=12,ns,poles,q;
double T=0.4;//torque
double speedw=785,Lendw,ls,VolumeM,Lqprime,Emax=24;
double D,ts,bss1,hsy,bss2,bso,Asl,kc,Bg=0.911,pi=3.14159,Drc,gap,lm,angle,Do,bts,hss,kopen,hsw,L;
double Kleak=0.95,Bm,Br=1.08,mur=1.03,Kleaktooth,Bts;
double Lq,muo=0.0000012566,permeancecoff=2,kw1=1,pcu=0.0000000172;
double kcoil,fs=0.65,V,ma=0.5,Vd=400;
double fluxm,Ns,Erms,S1,kcor=0.95,J,Di,Bry=1.4,Iphase,Pr,R,Ps,Pcu,Rn,Sins=0.00064,Rf,Eff,WindT,Cost;
double Eprime,Rprime;
double VolumeS,VolumeST,VolumeRT,VolumeCu,TotalVolume;
double WeightS,WeightM,WeightST,WeightRT,WeightCu,TotalWeight;
double Costiron,CostM,CostCu,Piron,WireArea;
double your_func;

#ifdef yours // define `yours' in the beginning of the code

MINM = 1; // use -1 for maximization

//-------------------------------------------------------------------------------------------------------------
//Variables definition
poles=x[0];//pole
q=x[1] ; //slot per pole per phase
Drc=x[2];//diameter rotor core
gap=x[3];//air gap length
lm=x[4];//magnet thickness
angle=x[5];//half pole angle in electrical degree
Do=x[6];//diameter stator outor
bts=x[7];//stator tooth width
hss=x[8];//stator slot height
kopen=x[9];//ratio of stator slot opening over slot width
hsw=x[10];//slot wedge height
L=x[11];//machine length
```

```
//BLDC EQUATION
//1-geometrical constraint
D=x[2] + 2*x[4] +2*x[3];
ts=pi*D/Qs;
bss1=(pi*(D+2*x[10])/Qs)-x[7];
hsy=0.5*(x[6]-D-2*x[8]);
bss2=(pi*(D+2*x[8])/Qs)-x[7];
bso=x[9]*bss1;
Asl=0.5*(bss1+bss2)*(x[8]-x[10]);

//2-Magnet properties
kc=ts/(ts-(pow(x[9]*bss1,2)/(bss1*x[9]+5*x[3])));
Bm=(Br*Kleak)/(1+(mur*x[3]*kc/x[4]));

//3-flux density
Kleaktooth=(17*x[0]/56-13/14)/100;
Bts=Bm*2*x[5]*(2/x[0])*(0.5*D-x[3])*(1-Kleaktooth)/(2*x[7]);
//-----------------------------------------------------------------------------
//ELECTRICAL PROPERTIES
//-----------------------------------------------------------------------------

//E'
Eprime=0.7071*speedw*kw1*x[1]*Bg*x[11]*(D-x[3]);

//kcoil
kcoil=1.6*Qs/x[0];

//R'
Rprime=pcu*(((x[0]*x[11])+(D+x[8])*pi*kcoil)*x[1])/(fs*Asl);

//Lq'
Lqprime=(x[0]*x[1]*x[11]*muo*permeancecoff)+((3/pi)*pow(x[1]*kw1,2)*(muo/(x[3]*kc*+x[4]/mur))*(D-x[3])*x[11]);

//external voltage(per phase)
V=0.612*ma*Vd/1.732;

//current loading
S1=(4*T)/(pi*(D-x[3])*x[11]*Bg*kw1*kcor*1*(D-x[3]));

//no of conductor per slot
ns=Emax/(x[0]*kw1*Bg*x[11]*0.5*D*x[1]*speedw);

//inductor estimation
Lq=pow(ns,2)*Lqprime;

//resistance per phase
R=pow(ns,2)*Rprime;
```

```
//induced voltage
fluxm=(2/pi)*Bg*x[11]*(D-x[3])*(pi/x[0]);

//no of turn per phase
Ns=(x[1]*x[0]*ns)/2;

//rms value of induced voltage
Erms=ns*Eprime;
//ampere turn per slot
//ampere turns and current density

Is= T/(x[0]*kw1*Bg*x[11]*0.5*D*x[1]);      //from Dr rama
Iphase=Is/(Nph*ns);                        //from Dr rama
J=S1*ts/(fs*Asl);

//inner rotor diameter(shaft diameter)
Di=x[2]-(Bm*2*x[5]*(2/x[0]))*(D-x[3])/Bry;

//wire size
WireArea=0.8165*Iphase/J;//after get value convert to AWG reference.

//-------------------------------------------------------------
// ACTIVE MATERIAL VOLUME
//-------------------------------------------------------------


//Volume Shaft
VolumeS=pi*pow(Di,2)*x[11]/4;

//Volume magnet
VolumeM= x[4]*(x[2]+x[4])*2*x[5]*x[11]*x[0];

//Volume stator
VolumeST=x[11]*(pi*0.5*D*x[8]+pi*hsy*(0.5*D+x[8]));

//Volume iron rotor stacking
VolumeRT=0.25*x[11]*pi*(pow(x[2],2)-pow(Di,2));

//Volume copper
VolumeCu=pi*fs*x[8]*(0.5*D+x[8])*(x[11]+2*pi*0.5*D*Kleak/2/x[0]);

//total volume
TotalVolume=VolumeS+VolumeM+VolumeST+VolumeRT+VolumeCu;

//-------------------------------------------------------------
// ACTIVE MATERIAL WEIGHT
//-------------------------------------------------------------
```

```
//weight shaft(stainless steel)
WeightS=VolumeS*7850;

//weight magnet(nedy iron boron)
WeightM=VolumeM*7500;

//weight stator
WeightST=VolumeST*7750;

//Weight rotor stacking
WeightRT=VolumeRT*7750;

//Weight wire copper
WeightCu=VolumeCu*8920;

//Total Weight
TotalWeight=WeightS+WeightM+WeightST+WeightRT+WeightCu;


//-----------------------------------------------------------
//ACTIVE MATERIAL COST
//-----------------------------------------------------------

//cost of iron
Costiron=7.7*(WeightRT+WeightST+WeightS);

//cost of magnet
CostM=WeightM*77;

//cost of copper   after get AWG, refer cost per kg.
CostCu=120*WeightCu;

//total cost
Cost=Costiron+CostM+CostCu;

//-----------------------------------------------------------
//POWER LOSSES
//-----------------------------------------------------------

//Pr=Nph*(Iphase*Iphase)*R;  //ohmic loss
Ps=0.01*T*speedw;  //stray losses
Lendw=kcoil*pi*(D+x[8]/2)/Qs; //length of winding
Rn=1/(pi*D*x[11]*(bss1*x[9]/ts + (1+2*pi/Qs)*x[8]/ts))*(Sins/43 + 1/10); //thermal resistance between slot copper and stator iron
Rf=0.02*(1/(pi*D*x[11]*(1+2*pi*x[8]/Qs/ts  + pi*Bg/2/(2*x[0])/1.4)));//thermal resistace between iron and external air
Pcu=pow(J,2)*Asl*pcu*Lendw*Qs;//copper losses
Piron=7750*VolumeST*10;//iron losses
```

```
//winding temperature
WindT=Pcu*(Rf+Rn);

//Efficiency
Eff=T*speedw/(T*speedw +Piron+Pcu+Ps)*100;

//-----------------------------------------
/*
printf("D = %f\n",D*1000);//unit mm
printf("ts= %f\n ",ts*1000);//unit mm
printf("bss1= %f\n",bss1*1000);//unit mm
printf("bss2= %f \n",bss2*1000);//unit mm
printf("hsy= %f \n",hsy*1000);//unit mm
printf("Bg= %f \n",Bg);
printf("Bry= %f \n",Bry);
printf("Asl= %f \n",Asl);
printf("ns= %d \n",ns);
printf("Ns= %f\n",Ns);
printf("S1= %f\n",S1);
printf("J= %f\n",J/1000);
printf("Ps = %f\n",Ps);
printf("Pcu = %f\n",Pcu);
printf("Piron = %f\n",Piron);
printf("Eff = %f\n",Eff);
printf("Is = %f\n",Is);
printf("WindT = %f\n",WindT);
printf("WireArea = %f\n",WireArea*1000000);//to get mm square by time 1000000
printf("Volume = %f\n",TotalVolume*1000000000);  //to get mm cubic
printf("Weight = %f\n",TotalWeight);
printf("Cost = %f\n",Cost);
printf("WeightM = %f\n",WeightM);
printf("Weightcopper = %f\n",WeightCu);
printf("Weightirn = %f\n",WeightS+WeightST+WeightRT);*/

//-----------------------------------------------------------------------
//Objective Function
  your_func=(Cost/24+ 2*TotalWeight+16500*TotalVolume)/3; // Put your function here
  //your_func=Cost;
  //your_func=TotalWeight;
  //your_func=TotalVolume;

  nc =8;
  // Put your constraints here

g[0]=hsy-0.5*x[8];
g[1]=x[8]-0.3*ts;
g[2]=bss1*x[9]-0.002;
```

```
g[3]=ts-x[7]-0.15*x[8];
g[4]=x[7]+0.5*x[8]-ts;
g[5]=1.6-Bts;
g[6]=Eff-90;
g[7]=0.5-TotalWeight;

  indv->obj = your_func;

#endif

  indv->obj = your_func;
  for (i=0, gsum=0.0; i<nc; i++)
   {
    indv->cons[i] = g[i];
    if (g[i] < 0.0) gsum +=pow(1000*g[i],2);
            your_func=your_func+ penalty_coef*pow(g[i],2);

   }
   indv->penalty = gsum;
}

/********************** END OF FILE **************************/
```

APPENDIX D

C++ CODE FOR BLDC MOTOR EQUATIONS FROM SIMULATED ANNEALING

```
//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
double function(double *x)
//Function calculates the objective function values given a design point.
//returns objective function value.
{
//declaring local variables
//int i;
int Nph=3,Qs=12,ns,poles;
double f, gtotal, result;
double g[8];
double T=0.4;
double speedw=785,Lendw,Is,VolumeM,Lqprime,Emax=24;
double D,ts,bss1,hsy,bss2,bso,Asl,kc,Bg=0.911,pi=3.14159,q,Drc,gap,lm,angle,Do,bts,hss,kopen,hsw,L;
double Kleak=0.95,Bm,Br=1.08,mur=1.03,Kleaktooth,Bts;
double Lq,muo=0.0000012566,permeancecoff=2,kw1=1,pcu=0.0000000172;
double kcoil,fs=0.65,V,ma=0.5,Vd=400;
double fluxm,Ns,Erms,S1,kcor=0.95,J,Di,Bry=1.4,Iphase,Pr,R,Ps,Pcu,Rn,Sins=0.00064,Rf,Eff,WindT,Cost;
double Eprime,Rprime;
double VolumeS,VolumeST,VolumeRT,VolumeCu,TotalVolume;
double WeightS,WeightM,WeightST,WeightRT,WeightCu,TotalWeight;
double Costiron,CostM,CostCu,Piron,WireArea;
gtotal = 0;

//Variables definition

poles=x[0];
q=x[1] ; //slot per pole per phase
Drc=x[2];//diameter rotor core
gap=x[3];//air gap length
lm=x[4];//magnet thickness
angle=x[5];//half pole angle in electrical degree
Do=x[6];//diameter stator outor
bts=x[7];//stator tooth width
hss=x[8];//stator slot height
kopen=x[9];//ratio of stator slot opening over slot width
hsw=x[10];//slot wedge height
L=x[11];//machine length
```

```
//BLDC EQUATION
//1-geometrical constraint
D=x[2] + 2*x[4] +2*x[3];
ts=pi*D/Qs;
bss1=(pi*(D+2*x[10])/Qs)-x[7];
hsy=0.5*(x[6]-D-2*x[8]);
bss2=(pi*(D+2*x[8])/Qs)-x[7];
bso=x[9]*bss1;
Asl=0.5*(bss1+bss2)*(x[8]-x[10]);

//2-Magnet properties

kc=ts/(ts-(pow(x[9]*bss1,2)/(bss1*x[9]+5*x[3])));
Bm=(Br*Kleak)/(1+(mur*x[3]*kc/x[4]));

//3-flux density

Kleaktooth=(17*x[0]/56-13/14)/100;
Bts=Bm*2*x[5]*(2/x[0])*(0.5*D-x[3])*(1-Kleaktooth)/(2*x[7]);
//-----------------------------------------------------------------------------------------------------------------------
//ELECTRICAL PROPERTIES
//-----------------------------------------------------------------------------------------------------------------------
//E'
Eprime=0.7071*speedw*kw1*x[1]*Bg*x[11]*(D-x[3]);
//kcoil
kcoil=1.6*Qs/x[0];
//R'
Rprime=pcu*(((x[0]*x[11])+(D+x[8])*pi*kcoil)*x[1])/(fs*Asl);
//Lq'
Lqprime=(x[0]*x[1]*x[11]*muo*permeancecoff)+((3/pi)*pow(x[1]*kw1,2)*(muo/(x[3]*kc*+x[4]/mur))*(D-x[3])*x[11]);
//external voltage(per phase)
V=0.612*ma*Vd/1.732;
//current loading
S1=(4*T)/(pi*(D-x[3])*x[11]*Bg*kw1*kcor*1*(D-x[3]));
//no of conductor per slot
ns=Emax/(x[0]*kw1*Bg*x[11]*0.5*D*x[1]*speedw);
//inductor estimation
Lq=pow(ns,2)*Lqprime;

//resistance per phase
R=pow(ns,2)*Rprime;

//induced voltage
fluxm=(2/pi)*Bg*x[11]*(D-x[3])*(pi/x[0]);
Ns=(x[1]*x[0]*ns)/2;
```

```
//rms value of induced voltage
Erms=ns*Eprime;

//ampere turn per slot
//ampere turns and current density

Is= T/(x[0]*kw1*Bg*x[11]*0.5*D*x[1]);      //from Dr rama
Iphase=Is/(Nph*ns);                        //from Dr rama
J=S1*ts/(fs*Asl);
//inner rotor diameter
Di=x[2]-(Bm*2*x[5]*(2/x[0]))*(D-x[3])/Bry;
//wire size
WireArea=0.8165*Iphase/J;
//-------------------------------------------------------------------------------
//Volume Shaft
VolumeS=pi*pow(Di,2)*x[11]/4;
//Volume magnet
VolumeM= x[4]*(x[2]+x[4])*2*x[5]*x[11]*x[0];
//Volume stator
VolumeST=x[11]*(pi*0.5*D*x[8]+pi*hsy*(0.5*D+x[8]));
//Volume rotor stacking
VolumeRT=0.25*x[11]*pi*(pow(x[2],2)-pow(Di,2));
//Volume copper
VolumeCu=pi*fs*x[8]*(0.5*D+x[8])*(x[11]+2*pi*0.5*D*Kleak/2/x[0]);
//total volume
TotalVolume=VolumeS+VolumeM+VolumeST+VolumeRT+VolumeCu;
//-------------------------------------------------------------------------------
// Weight
//-------------------------------------------------------------------------------
//weight shaft
WeightS=VolumeS*7850;
//weight magnet
WeightM=7500*VolumeM;
//weight stator
WeightST=VolumeST*7750;
//Weight rotor stacking
WeightRT=VolumeRT*7750;
//Weight wire copper
WeightCu=VolumeCu*8920;
//Total Weight
TotalWeight=WeightS+WeightM+WeightST+WeightRT+WeightCu;
//-------------------------------------------------------------------------------
//Cost
//-------------------------------------------------------------------------------

//cost of iron
Costiron=7.7*(WeightRT+WeightST+WeightS);
```

```
//cost of magnet
CostM=WeightM*77;

//cost of copper
CostCu=120*WeightCu;

//total cost
Cost=Costiron+CostM+CostCu;

//-------------------------------------------------------------------
//Power losses
//-------------------------------------------------------------------
Pr=Nph*(Iphase*Iphase)*R;  //ohmic loss
Ps=0.01*T*speedw;  //stray losses
Lendw=kcoil*pi*(D+x[8]/2)/Qs; //length of winding
Rn=1/(pi*D*x[11]*(bss1*x[9]/ts + (1+2*pi/Qs)*x[8]/ts))*(Sins/43 + 1/10); //thermal resistance between slot copper and stator iron
Rf=0.02*(1/(pi*D*x[11]*(1+2*pi*x[8]/Qs/ts  + pi*Bg/2/(2*x[0])/1.4)));//thermal resistace between iron and external air
Pcu=pow(J,2)*Asl*pcu*Lendw*Qs;//copper losses
Piron=7750*VolumeST*10;//iron losses

//winding temperature
WindT=Pcu*(Rf+Rn);


//Efficiency

Eff=T*speedw/(T*speedw +4+Pcu+Ps)*100;

//-------------------------------------
//calculating objective function value
f=(Cost/24+ 2*TotalWeight+16500*TotalVolume)/3;

// Put your constraints here

        g[0]=0.5*x[8]-hsy;
        g[1]=0.3*ts-x[8];
        g[2]=0.002-bss1*x[9];
        g[3]=0.15*x[8]+x[7]-ts;
        g[4]=ts-0.5*x[8]-x[7];
        g[5]=Bts-1.6;
        g[6]=90-Eff;
        g[7]='TotalWeight-0.5;

char *file1;
file1 = "output2.txt";
ofstream a_file(file1);
file1 = (char *)malloc(32*sizeof(char));
```

```
a_file<<"-------------------------------------------------"<<endl;
a_file<<"OPTIMIZED MOTOR DATA USING SIMULATED ANNEALING (SA)"<<endl;
a_file<<"-------------------------------------------------\n"<<endl;

a_file<<"Bore Diameter = "<<D<<endl;
a_file<<"Motor Length = "<<x[11]<<endl;
a_file<<"Air Gap Length = "<<x[3]<<endl;
a_file<<"Magnet Length =  "<<x[4]<<endl;
a_file<<"External Diameter = "<<x[6]<<endl;
a_file<<"Slot Number = "<<Qs<<endl;
a_file<<"Pole Number = "<<x[0]<<endl;
a_file<<"Air Gap Flux density = "<<Bg<<endl;
a_file<<"Rotor Yoke Flux Density = "<<Bry<<endl;
a_file<<"Tooth Flux Density = "<<Bts<<endl;
a_file<<"Current Density = "<<J<<endl;
a_file<<"Magnet Weight = "<<WeightM<<endl;
a_file<<"Iron Weight = "<<WeightS+WeightST+WeightRT<<endl;
a_file<<"Copper Weight = "<<WeightCu<<endl;
a_file<<"Total Material Weight = "<<TotalWeight<<endl;
a_file<<"Total Material Volume = "<<TotalVolume<<endl;
a_file<<"Iron Losses = "<<Piron<<endl;
a_file<<"Copper losses = "<<Pcu<<endl;
a_file<<"Stray Losses = "<<Ps<<endl;
a_file<<"Winding Temperature = "<<WindT<<endl;
a_file<<"Efficiency = "<<Eff<<endl;
a_file<<"Material Cost = "<<Cost<<endl;
a_file<<"Multi Objective = "<<f<<endl;
a_file<<"\n-------------------------------------------------"<<endl;


//evaluating constraints. setting constraints that are less than0 to 0.
//Internal penalty method used to account for constraints

if(g[0] <= 0)
g[0] = 0;
gtotal = gtotal + pow(1000*g[0],2);
//summing objective function value and active constraints
result = f + rp*gtotal;
funcEval++;
return result;
}
```