# **CHAPTER 4**

# **OPTIMIZATION AND DESIGN**

## Introduction

In the previous chapter, electrical and mechanical design parameters of the selected axial flux permanent magnet generator are expressed. To do that, mathematical design equations and related drawings are represented. These equations are important for this thesis work. Because they are used in the main design code, which is written in MATLAB. Also in the previous chapter, verification of the given analytical equations of the some important design parameters is given by means of finite element analysis for a sample design. For this purpose, comparison of the design equations and the finite element analysis is made in terms of airgap flux density and induced emf. It’s concluded that the results show good agreement. Therefore, these equations can be used in the optimization algorithm with high accuracy. In this chapter, optimization process of the given design will be summarized and optimum design parameters of the proposed 5 MW AFPM generator will be determined. First, evolutionary algorithms (EA) will be reviewed including the selected genetic algorithm (GA). Then, process of the genetic algorithm based optimization method, which is used in this thesis study, will be explained in detail. Optimization of the proposed generator is constructed with MATLAB optimization toolbox. Also in this chapter, a brief information of this toolbox and used parameters in the optimization algorithm will be covered. Finally, optimized design parameters of the proposed 5 MW 12 rpm AFPM generator will be given. These design parameters will be used for the finite element modelling and analysis in the following chapter.

## Evolutionary Algorithms (EA) and Genetic Algorithm (GA)

There exist different mathematical search algorithms and conventional methods for modern world engineering problems. However, multi-variable nonlinear problems require new methods to avoid from getting stuck into local minimums during optimization process [1]. Main motivation in the Evolutionary Algorithms (EA) is to mimic the nature to find optimum solutions to these problems. EA can be evaluated as direct, stochastic and population-based search algorithm. There are three main rules of biological processes which inspire the EA based search algorithms. These processes can be summarized as follows :

* **Continuous evolution process** , which occurs at the most basic level of “source-code” of living beings, i.e. chromosomes
* **Natural Selection mechanism** , in which the fittest individuals in a society can have more chance to survive and have more robust offspring than those who are not fit at all.
* **Evolutionary process at reproduction** , which is done by the reproduction operators such as cross-over and mutation.

EA mimics the natural selection of living beings. Fittest one in the group has more chance to survive and to be selected. Individuals correspond to encoded solutions of the given problem. Every individual has a fitness value which is calculated by objective function of problem. Algorithm itself evaluates the “adaptive skills” of every indivudials according to this fitness value. Least “fit” individuals are eliminated from the population, hence more adapted and robust individuals replace the old generations. Fitness value is the only required quantitive information about the individual in EAs , contrary to other search techniques such as gradient based optimization methods, in which derivative information is needed [2]. Another advantage of evolutionary search algorithm is population based evaluation, which is a big computational load advantage over the conventional search algorithms which sample one individual at a time. This population leveled optimization is more advantageous especially when working with large search spaces [3], [4]. In Fig. 4-1, a classification table of search techniques is given.



Fig. 4-1. Classification of search techniques[1], [4]

The most popular search technique among other techniques in the EA family is the genetic algorithms (GA). In this algorithm, individuals are generally represented as fixed-length bit strings as shown in Fig. 4-2 and Fig. 4-3 . Different cell positions in these strings contains information which corresponds to different properties of the individual they represent [4]. Two frequently used operators during the reproduction stage of GA are cross-over and mutation operators. Various “species” or various “solutions” can be obtained during the optimization process by using these two operators. Working principles of cross-over and mutation operators are depicted in Fig. 4-2 and Fig. 4-3, respectively. In cross-over, data interchanges between parents around the crossover point which determined in the reproduction stage. However, in mutation, random new data is written to randomly selected locus on the selected “chromosome” or “individual”.



Fig. 4-2. Bit string cross-over operation between parent individuals [4]



Fig. 4-3. Bit string mutation operation [4]

Evolutionary algorithms start with the initial population where values of the initial variables are selected randomly by selection operators based on stochastic methods. Successive generations are created based on the selection and the reproduction principles. Population size is preserved throughout the generations. Algorithm stops when termination criterions are satisfied [3], [4]. These criterions can be different conditions such as predetermined fitness value, predetermined number of successive generation or limited time.

Every problem can be solved by using EA as long as it is expressed with a proper fitness function. User should define a fitness function such that generations could converge to optimal solution. Therefore, every necessary parameter and penalty coefficient corresponding to it should exist in the fitness function maybe not equally but in a weighted form [2]. Penalty coefficients and related definitions will be covered in the following sections. Another advantage of EAs is that it can be combined with other conventional search techniques. EAs can be utilized in a parallel fashion in order to evaluate the fitness among the candidate solutions, as mentioned before. Possibility of converging local minimum is decreased due to this parallel process. Because of the high computational burden related to larger search spaces and hybridization processes, distributed computing gaining attention [4]. Also, evolutionary algorithms can easily adapt to changing environment conditions. Therefore, it’s not necessary to restart the algorithm in case of sudden changes, contrary to as it was in conventional search methods[1].

To sum up, evolutionary algorithms gaining popularity especially in the last two decades due to advantages aforementioned above although first attempts of evolutionary techniques in optimization problems were made in nearly 60 years ago [3], [4]. There are two biggest key aspects of this search technique. One of is that the similarity between the nature during selection and variation stages. The other one is that it is not necessary to provide mathematical information except fitness function in order to evaluate generations of individuals [5]. Additionally, there exist a large application area of this algorithm from medical treatments to advanced engineering problems [6]. This application area seems to enlarge due to new explorations of evolutionary genetics science in biology and increased computer capacities.

## Genetic algorithms based optimization

Genetic algorithms (GA) are stochastic search techniques and exist on the subgroup of evolutionary algorithms. GA was first proposed by John Holland in 1975 with the aim of investigating the usage of natural evolutions on optimization principles [3]. The most salient feature of the GA among the other search techniques is that it doesn’t need derivative information of related search space. This feature helps GA to avoid trapping at local minimums [6], [7]. Algorithm itself based on the three operators namely selection, crossover and mutation [3], [8]. As it was in the evolutionary algorithm case, GAs can also explore the search space in a parallel fashion. Another advantage of GA is that optimization procedure can converge to global minimum solution regardless of the starting point. Crossover and mutation definitions are same for the GA as it is mentioned in previous section. For an effective optimization, options of the GA such as population size, cross-over and mutation possibilities and termination criterion, should be suitably configured [9].

General flowchart of a GA is given in Fig. 4-4. However, it is useful to describe some of the technical terms about GA before continue with the flowchart.

* *Gene* is a parameter which defines the specific trait of the considered solution. For example; stator outer diameter, axial length, airgap flux density. This parameter is encoded in the related locus of fixed-length chromosome.
* *Chromosome* is the combined form of genes, thus representing a complete “individual”.
* *Locus* is the specific position of encoded data exist in each string of individual or solution.
* *Fitness* is the measure of how suitable a generated solution is. This numeric value is used by the GA when evaluating and selecting the best individuals from the candidate solutions. Because of this reason GA optimizations are usually mentioned with the term “survival of the fittest”. Fitness function of the optimization procedure should be constructed carefully in order to achieve the optimum design parameters of the selected AFPM.
* *Selection* used in GA is mainly based on stochastic processes and natural similarities. However, there are different selection methods for application such as roulette wheel selection and tournament selection.
* *Population* can be considered as a group of individuals in one generation. Large sizes of population leads to longer solution times but larger search spaces.
* *Generation* is the set of individuals employed in one cycle of optimization. As the evaluated number of generations are increased, more fit solution candidates will be created by the GA.
* *Elitism* is related to best individuals which are preserved and directly pass to next generation without manipulation. If number of elite is too much generations don’t change much and diversity decreases. If number of elites is low then optimization lasts longer to converge global minimum because of large diversity.
* *Independent variable* is an optimization parameter which is changed by the GA at every iteration. For example in this thesis work, there are 15 different independent variables in the optimization process of the proposed AFPM.
* *Penalty function* is a concept that is used to convert a constrained optimization to an unconstrained optimization problem. Main idea in this concept is that to “penalize” the individuals with additional higher fitness values, whose solution parameters violate the limits of predetermined constraints.



Fig. 4-4. General flowchart of a GA optimization [9]

## MATLAB GA Toolbox Implementation

## MATLAB Toolbox and Configurations

In this thesis study, required optimization procedure is handled by MATLAB optimization toolbox. For this purpose, three different codes, which include the necessary design equations described in the previous chapter, are written and tested in the MATLAB environment. These codes are mainly performs; “optimization main handling and saving performance parameters”, “iterative loop for required multi-speed operation calculation” and finally “main design calculation of the generator for a given set of variables” actions. A sample view of aforementioned toolbox GUI is given in Fig. 4-5.

In MATLAB optimization toolbox “ga-genetic algorithm” solver is utilized in order to find the optimum solution to objective function of the given problem. In our design, objective function is constructed based on the cost of the proposed generator. Therefore, GA tries to minimize the total mass of the generator since unit costs of the materials taken as constant in the optimization. Details of the objective function and constants will be given in the following subsections. In this subsection, details and the configurations of optimization procedure will be described. Configuration parameters used in the optimization process are given in Table 4-1.

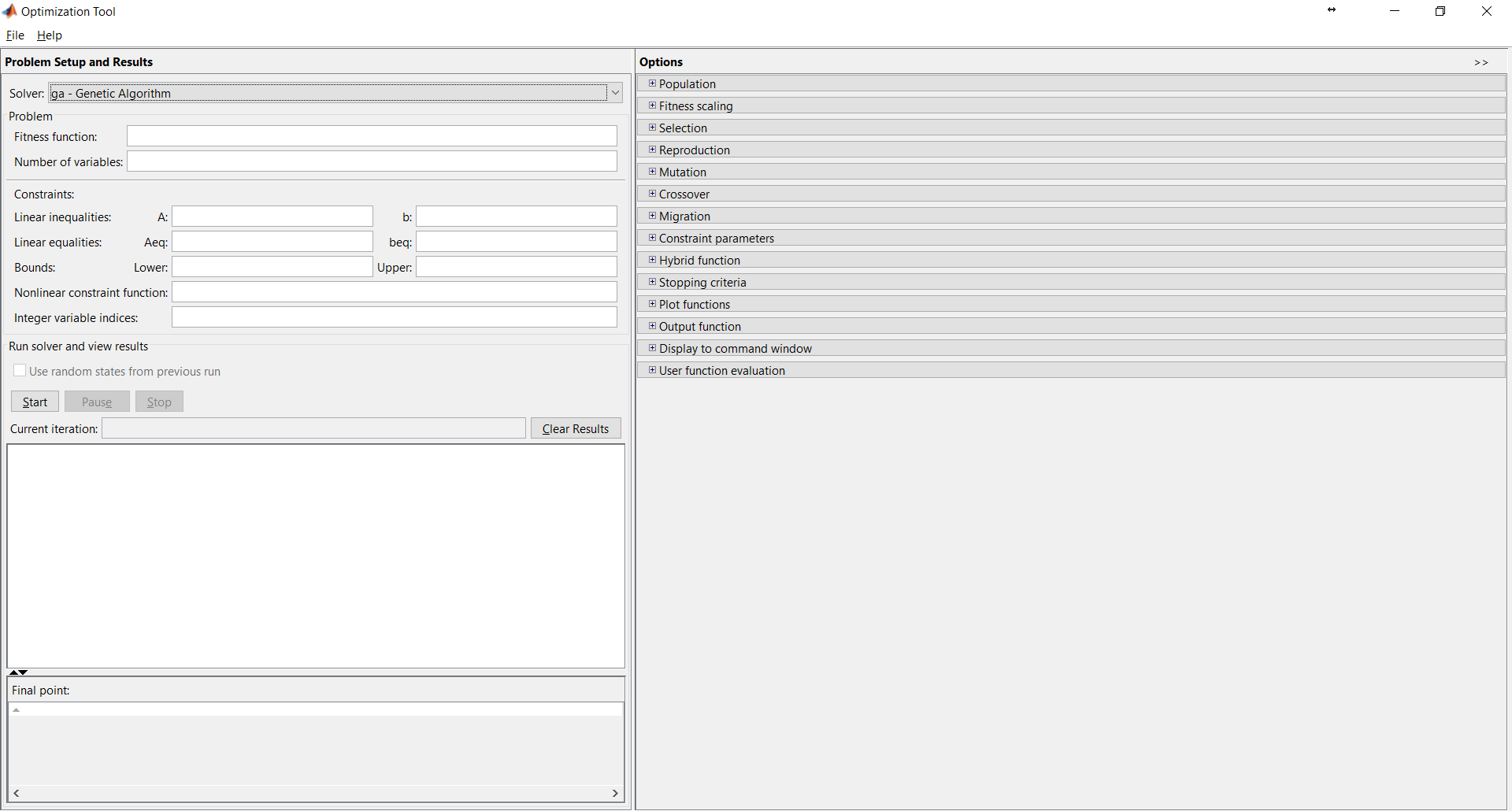


Fig. 4-5. MATLAB Optimization Toolbox GUI

Table 4-1. Configuration parameters of the optimization

|  |  |
| --- | --- |
| Solver | Genetic Algorithm-ga |
| Number of variables | 16 |
| Population Size | 100 |
| Fitness Scaling | Rank |
| Selection Function | Stochastic Uniform |
| Elite Count | 15 |
| Crossover Fraction | 0.9 |
| Mutation | Gaussian with Scale/Shrink |
| Crossover Function | Scattered |
| Number of Generation | 200 |
| Stall Generations | 50 |
| Stall tolerance | 1x10-6 |

As seen from Table 4-1, there are 16 different independent variables used in our design optimization. These variables can be seen in Table 4-2. However, airgap clearance parameter, which exists in the second locus of the variable vector, are used as a constant during the optimization process. It is intendedly placed in the variable vector to see the effect of the airgap change in the resulting design parameters. Individuals in a population evaluated according to their calculated fitness values and then get a rank number inversely proportional to this fitness. In other words, most “fit” individual ranked by 1 while least “fit” individual ranked by 100.

Selection is realized via stochastic uniform function based on the fitness value. In this function, individuals have probability to be selected by the GA inversely proportional to their rank value. Therefore, individuals with lower rank value have more chance to be selected. Cross-over fraction determines the rate of the individual in a population (except the elite ones) which are subjected to cross-over operation during the reproduction stage. Higher rates of this parameters results in higher diversity despite longer solution times. Cross-over is realized via scattered function. In this function first a random vector which consists of random binary numbers. Then this random vector is compared with the selected parent vectors in bit-wise. Variables of the offspring individual created according to this comparison. If binary number is 1 then “gene” is taken from first parent otherwise second parent gives the related gene from its corresponding locus [10].

Mutation is realized via Gaussian Scale/Shrink method. In this method, a random Gaussian distribution number, whose standard deviation is controlled via Scale and Shrink parameters, is added to each gene of the individuals. However, this standard deviation has a decreasing trend through the generations. In this study standard deviation of the added number in the first generation is %1 while it “shrinks” to zero in the last generation. Therefore natural mutation is imitated as the better generations are created [10].

Table 4-2. Configuration parameters of the optimization

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Vector-x** | **Variable Definition** | **Variable Vector-x** | **Variable Definition** |
| x(1) | Mean radius | x(9) | Number of poles *Np* |
| x(2) | Airgap *g* | x(10) | Number of branches |
| x(3) | Current Density *J* | x(11) | Height of the winding |
| x(4) | Outer limb thickness | x(12) | Pitch ratio |
| x(5) | Inner limb thickness | x(13) | Fill factor *kfill* |
| x(6) | Steel web thickness *lc* | x(14) | Height of the magnet *hm* |
| x(7) | Magnet/steel width ratio | x(15) | Length of the magnet *l*m |
| x(8) | Number of turns *Nt* | x(16) | Number of parallel stacks |

In our optimization process two different termination criterions are defined as it can be seen on Table 4-1. Optimization process will stop either when the total number of generation is equal to 200 or when the number of successive generations with average change in fitness function is less than “Stall tolerance”. GA algorithm in MATLAB searches for the optimum set of parameters in the predetermined lower and upper boundaries. These boundaries of the optimization is given in Table 4-3 with respective units.

Table 4-3. Lower and upper boundaries of the independent variables

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable name** | **Lower boundary** | **Upper boundary** | **Unit** |
|  | 4 | 8 | m |
| *J* | 5 | 7 | A/mm2 |
|  | 0.03 | 0.045 | m |
|  | 0.02 | 0.03 | m |
| *lc* | 0.02 | 0.03 | m |
|  | 0.7 | 0.8 | - |
| *Nt* | 50 | 90 | Turn |
| *Np* | 200 | 260 | Pole |
|  | 4 | 6 | Branch |
|  | 0.03 | 0.045 | m |
|  | 0.3 | 0.4 | - |
| *kfill* | 0.7 | 0.8 | - |
| *hm* | 0.01 | 0.02 | m |
| *l*m | 0.25 | 0.3 | m |
|  | 3 | 6 | - |

As mentioned earlier, airgap clearance *g* is segregated from this table because it is taken as constant. Constant value of this parameters is taken as 7 mm for our proposed generator.

## Constants

## Flowchart and fitness function

As mentioned before, penalty functions are defined and used in order to convert our constrained optimization problem to an unconstrained optimization problem.

## 5MW AFPM generator with optimized design parameters

**References**

[1] Umut Güvengir, “ONLINE APPLICATION OF SHEM TO GRID-CONNECTED INVERTERS WITH VARIABLE DC LINK VOLTAGE BY PARTICLE SWARM OPTIMIZATION,” 2014.

[2] O. Keysan, A. S. McDonald, and M. Mueller, “Integrated Design and Optimization of a Direct Drive Axial Flux Permanent Magnet Generator for a Tidal Turbine,” in *International Conference on Renewable Energies and Power Quality - ICREPQ’10*, 2010.

[3] M. Mitchell, “An introduction to genetic algorithms,” *Comput. Math. with Appl.*, vol. 32, no. 6, p. 133, 1996.

[4] S. N. Sivanandam and S. N. Deepa, *Introduction to genetic algorithms*. 2008.

[5] P. Virtiˇ and M. Vraˇ, “Design of an Axial Flux Permanent Magnet Synchronous Machine Using Analytical Method and Evolutionary Optimization,” *IEEE Trans. Energy Convers.*, vol. 31, no. 1, pp. 150–158, 2016.

[6] G. Papa, B. Korousic-Seljak, B. Benedicic, and T. Kmecl, “Universal motor efficiency improvement using evolutionary optimization,” *IEEE Trans. Ind. Electron.*, vol. 50, no. 3, pp. 602–611, Jun. 2003.

[7] R. Zeinali, “DESIGN AND OPTIMZIATION OF HIGH TORQUE DENSITY GENERATOR,” *MS thesis*, no. September, 2016.

[8] M. Łukaniszyn, M. JagieŁa, and R. Wróbel, “Optimization of permanent magnet shape for minimum cogging torque using a genetic algorithm,” *IEEE Trans. Magn.*, vol. 40, no. 2 II, pp. 1228–1231, 2004.

[9] J. Azzouzi, N. A. Karim, G. Barakat, and B. Dakyo, “Axial flux PM synchronous generator design optimization: robustness test of the genetic algorithm approach,” *2005 Eur. Conf. Power Electron. Appl.*, vol. 9, p. 10 pp.-pp.P.10, 2005.

[10] “MATLAB Documentation.” [Online]. Available: https://www.mathworks.com/help/matlab/. [Accessed: 08-Aug-2017].