2002

Parameter identification of induction motor using a genetic algorithm

Edina Bajrektarevic
West Virginia University

Parameter Identification of Induction Motor Using a Genetic Algorithm

by

Edina Bajrektarević

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Wils L. Cooley, Ph.D.
Ronald L. Klein, Ph.D.
Muhammad A. Choudhry, Ph.D., chair

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2002

Keywords:
induction motor, genetic algorithm, parameter identification, nonlinear modeling, lab view software

Abstract

Parameter Identification of Induction Motor Using a Genetic Algorithm

by

Edina Bajrektarević
Master of Science in Electrical Engineering

West Virginia University

Muhammad A. Choudhry, Ph.D., Chair

High performance variable-speed machines incorporate a model for the system in either the controller or state estimation stages. The accuracy and general robustness of the machine is dependant on this model. Therefore, it must accurately represent both the electrical and electromagnetic interactions within the machine and associated mechanical systems. Recently, some new technologies have been tested in the field of electro mechanics like neural networks, fuzzy logic, simulated annealing and genetic algorithms. These methods are increasingly being utilized in solving electric machine problems.

In this thesis, a genetic algorithm (GA) – a form of artificial intelligence – is employed to identify the electric parameters of induction motors. The variables used to calculate the electric parameters are the measured stator currents, stator voltages and rotor speed. The variables are acquired by using Data Acquisition System and Lab VIEW Software. Free acceleration test is performed on 7.5 hp induction motor, using a constant frequency power supply. The performance of the identification scheme is demonstrated with simulated and measured data, and electric parameters obtained using this method are compared with parameters obtained from IEEE standard tests. Based on the results, the method proved to be worth considering in optimizing induction machines and can be applied to a variety of induction motor parameter estimation problems.

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to my advisor Dr. Muhammad A. Choudhry whose guidance and encouragement was instrumental in the successful completion of my work.

I would also like to thank my committee members Dr. Wils L. Cooley and Dr. Ronald L. Klein for supportive advice and discussion. Furthermore, I am very thankful to professors, my students and staff at Lane Department of Computer Science and Electrical Engineering (LDCSEE) for making my work such an enjoyable and memorable experience. I extend the appreciation to my friend Kourosh Sedghisigarchi for providing a great contribution to this research.

Special thanks goes to my family for their inspiration and encouragement during this work. None of my accomplishments would have been possible without their endless love.

Last but not the least I would like to acknowledge the support of LDCSEE to present my research results at 2002 IEEE Winter Power Engineering Society Meeting, which resulted in a third place prize for the Student Poster – Paper Contest.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The significance of the AC induction motors in speed and position controlled drives has grown drastically in the recent decade. In the United States, electric motors consume nearly two thirds of the electricity produced, where the induction motors are the most important energy consumers. In order to achieve high efficiency of the technology, many non – controlled AC drives are being renovated by adding frequency converters. They are now used as speed controlled drives. To make perfect static and dynamic qualities of these drives, control engineers need to have more information available on the controlling object. Thus, engineers should be able to determine accurate motor parameters and to properly select a model of the machine. In the case of incorrect parameter values used in the controller, instantaneous error will appear in both torque and flux resulting in a change in dynamics. There is another concern regarding the motors in the plant, which originates from different manufacturers, where the parameters are not known prior to start – up of the drive. To overcome these problems, a lot of work has been done towards determining the parameters of induction motor prior to the starting and selection of the appropriate models for study.

Furthermore, induction motors constitute a theoretically challenging control problem, since the dynamical system is highly nonlinear, the electric rotor variables are

not measurable, so the electromagnetic parameters are most often not precisely known. Also, the skin effect in the rotor winding, and the iron core saturation lead to even bigger complications in the modeling process of the machine. Thus, the source for the determination of such parameters is usually based on indirect measurement methods where the input is a generic, such as input stator voltage or rotational speed of the motor and the output of the model is compared with the measured one.

Conventional optimization techniques have been applied to this type of problem with limited success, although with the inherent problem of convergence to a local minimum instead of global one. The optimum parameter values determined by these techniques depend heavily on the initial guess of the parameter, with the possibility of a slightly different initial value causing the algorithm to converge to an entirely different solution. Also, one important parameter required by the algorithm is the derivative of the function, which is not always available or may be difficult to calculate. These problems have encouraged the different authors to investigate alternative techniques of solution.

Recently, the use of Artificial Neural Networks (ANNs) and Genetic Algorithms (GAs) has been proposed to identify and control nonlinear dynamic systems including induction machines. In this process the selected optimization method is based on a genetic algorithm. Identification of the induction motor is performed using data acquired during experimental tests consisting of transients varying from standstill to a certain speed. Data is acquired by using a Data Acquisition Card (DAQ) and Lab VIEW Software. The method is based on the idea of determining the unknown five electromagnetic parameters representing the model of the induction machine. The genetic design is described with genes, and instead of one single design an entire population of designs is studied. This enables the use of statistical means for studying the results and improving the reliability of the results. Several different machines are studied, including theoretical models of the induction machine and one physical model, with the objective of minimizing the error between the estimated and measured outputs.

1.1. Thesis Outline

This chapter summarizes some of the existing work relevant to the identification of the parameters of induction motor and reviews some of the essential issues in parameter identification. Two basic schemes are addressed related to on line and off line identification algorithms. Different methods applied to the identification of the parameters of induction motors are discussed. Basics on GAs are given in Chapter 2. Genetic operators are discussed along with an example, which illustrates the work of GAs. Implementation of GAs in the optimization of electric machines is presented. Chapter 3 provides some basic information on the nonlinear induction motor model used in this study [6]. It describes the startup behavior of the induction machine and the dynamics associated with the transients of the machine itself. Transformation to an arbitrary, rotating reference frame is introduced to interpret the induction motor model. The computer simulation of the induction machine is solved in Simulink/Matlab, and can be found in Appendix B. In Chapter 4 an experimental test to acquire the input and output variables of induction motor is described. The procedure starts by explaining the hardware instruments used for *Data Acquisition System* (DAQ). The process configures as a *Virtual Instrument* (VI), is designed in Lab VIEW Software to acquire and read different output signals from machine. Appendix A gives more information on hardware experimental setup. Instructions as how to start VI applications in Lab VIEW Software can also be found here. Chapter 5 describes a GAs implementation to optimize the parameters of the induction motor. Source code for the procedure can be found in Appendix C. Parameter estimation results for the different approaches are evaluated and compared, using both actual and simulated data. Conclusion of this work and recommendations for future research are given in Chapter 6.

1.2. Literature Survey

Accurate and reliable parameter estimation techniques for induction machine are critical for the design and development of high – performance drive systems in which the

parameter estimates are used in the field orientation, motion control, self – sensing, and other advanced algorithms. There are two basic approaches to this problem:

- *On - line identification methods* – utilize state observer theory (e.g. Kalman filter) and Least Square based techniques.

- *Off - line techniques* - rely on statistical curve fitting to the measured data under specific conditions.

Apart from the standard IEEE test procedure [36] recently some methods of parameter identification appeared in the literature. The standard technique of using the short – circuit, no – load and blocked – rotor tests seems to be inaccurate and, consequently, not suitable for the synthesis of high dynamic performance systems. This mainly comes from the assumption that the parameters are constant, regardless of the operating conditions. The following are summaries of some related work in this area, which gives a comprehensive review of the literature.

Reference [37] suggested using an extended *Kalman filter* for parameter estimation. The machine's response to load perturbations was measured and the resulting changes in current, voltage, speed and torque were used to estimate the values of H, mechanical damping and X_m, magnetizing reactance. Since torque measurements are of the prime importance of their method, it may be difficult to implement this approach for larger machines. Also, the method does not yield values for X_1, X_2, R_s or R_r. In this case, the machine is modeled by the linear equations, and the parameters for the specific operating condition are estimated.

In reference [11] a parameter identification method is applied to identify all electric parameters simultaneously. The method assumes that the motor can be described by a time – varying linear model. This method is based on filtering the current, voltage, and speed signals of the motor so that the set of acquired signals are related by simple linear equations. Two estimation methods are applied: General Total Least Squares (TLS)

method and Constrained General TLS. First method proved to be sensitive to the noise. A second method, which uses TLS algorithm with extra constraints, proved to be well defined and performed better under high – noise conditions. However, some interesting phenomena may occur here like pole – zero cancellations and non – persistency of excitation.

The induction machine parameters change far more drastically during starting conditions than at steady state conditions due to a large inrush of current at startup. Therefore, most of the research in this field is done during this interval since the free – acceleration response data gives more information on machine nonlinearity and is therefore more suitable for parameter identification. The work in this thesis also considers this time duration.

Reference [3] describes three methods for estimating the model parameters of an induction motor using free acceleration data. A three – phase balanced induction motor is studied. Measurements of the stator currents and voltages are required for the identification procedure, but no measurements from the motor shaft are needed. The first method applies simple induction motor models with limited temporal domains of validity and obtains parameter estimates by extrapolating the model error bias to zero. Thus, the philosophy of the method is to decompose transients described by a complicated model into smaller domains described by simple, easy to identify models. This method does not minimize any specific error criterion and is presented as a means of finding a good initial guess for a conventional iterative maximum – likelihood or least squares estimator. The second method minimizes equation errors in the induction motor model in the least – square sense using a Levenburg – Marquardt iteration. Presumably, excitation V_s and stator currents I_s are known and measured, so the objective is to minimize the loss function defined as a function of parameters, V_s and I_s. The third method is the continuation of the Levenburg – Marquardt method, motivated by observed properties of some pathological loss functions. The third method is applied to fit two different mechanical load models and it proves to be uniformly good. The methods described can

be applied as "parameter acquisition" tests to determine parameters or even initial guesses required for the state estimator.

Conventional gradient decent techniques for finding the minima of functions are fast and efficient from the computational point of view, but they have a problem of limited value when several local minima exist. Such multi - mode behavior is often associated with functions that involve noisy experimental data such as is found in the case of induction motors. Also, a common thread among these induction machine parameter and state estimation problems is the need for either a good initial guess or accurate knowledge of machine parameters. The idea of employing new different algorithms techniques for solving this problem has been recognized for some time. Genetic algorithms (GAs) and Artificial Neural Networks (ANNs) have been gaining popularity as an optimization technique because of their inherent robustness. Also, they may locate a global minimum, and hence be used for curve fitting with experimental data, even when many local minima exist. The recent work reviewed in the following paragraphs covers a broad range of induction motor parameter estimation by applying these two techniques, presented from various perspectives.

Reference [24] presents a new approach to identify the nonlinear model of an induction machine. The same nonlinear model as in [6] is used for the simulation study. The measurements of the stator voltages, stator currents, and rotor angular velocity are obtained by applying the three – phase AC power to a 5 hp induction motor while it is in standstill condition without mechanical load. For simulation purposes, the measured stator voltages and rotor angular speed are treated as input, and the stator currents are simulated as output. Feed – Forward Neural Networks (FNNs) models are utilized to represent the nonlinear parameters as functions of operating conditions. The FNN estimates the nonlinear functional relationship between input and output patterns by learning from the training data. The input pattern of the FNN is composed of the magnitudes of the d – and q – axis currents and the rotor angular velocity. Each machine parameter is considered as the output pattern. The whole free acceleration test data set is

divided into subsets (time intervals of 0.05 s) which contain short time interval test data, and the parameter values in a subset are considered to be a constant. As the operating condition changes, the input pattern changes as well and the FNN model determines the parameter values, which correspond to the operating condition. The maximum likelihood (ML) algorithm is used to estimate the parameter values from each subset of data. These estimated parameter sets represent the nonlinear model parameter values for different operating conditions. The author proves that parameters are nonlinear in nature and can be represented by the FNNs.

Another interesting research in the area of Artificial Neural Networks (ANNs) reference [15] describes a technique to identify and control induction machine. Two systems are presented here: a system to adaptively control the stator currents via identification of the electrical dynamics, and a system to adaptively control the rotor speed via identification of the mechanical and current – fed system dynamics. For both methods, ANN is introduced and observable forms of the models are described. Performances of these controlling schemes are compared with the standard vector control scheme.

One of the pioneers in the applicability of genetic algorithms (beginning of 1990) to the problem of the motor parameter determination are Richard R. Bishop and Gill G. Richards. They used a simplified steady state model from [36] of induction machine for parameter identification. The problem is formulated as to find several parameters simultaneously while stator resistance and rotor reactance are obtained directly from available manufacturer's data. The algorithm was tested using data from a laboratory ½ hp three-phase induction motor. The motor load tests consist of measuring the motor terminal complex impedance, Z_m, at rated voltage for at least two different values of slip, s_1 and s_2. A measurement of electrical torque is also made at each of the test speeds. Then GA is employed to find a set of parameters such as to minimize the error function at the slip (Note: error is the function of Z_m and T_m). The authors were more concerned in proving that GA can be used for parameter identification. Hence, convergence of the

parameters under different generations sets was analyzed. GA proved to be well suited to the parameter identification problems.

In reference [21] motor parameter identification problem is defined by using steady state models of approximate equivalent circuits, exact equivalent circuits, and a deep bar circuits. The fitness function is defined as the reciprocal of the error between the input motor torques and the calculated torques. The input torques are full load, locked rotor, and breakdown torques obtained from real manufacturer's data. Torques are calculated using the algebraic circuit equations. The fitness is defined as to minimize errors between the input motor torques and the calculated torques. GA is then employed to determine suitably accurate parameters. Here, the motor parameters are assumed to be constant, mainly because the intention was to use the parameters for system level studies where extreme precision is unnecessary. This approach indicates that performance of GA can be affected by numerical values of constants needed in implementation, e.g. mutation or crossover rate. Results show that the approximate equivalent circuit prevents the GA from determining the motor parameters accurately since in the case of 5 hp machine error between the full load torque and calculated one was 20% while in the case of deep bar models it reduces to 3% and 6%. The deep bar model gave better results thus reaffirming the need to use a more accurate model representation in the motor parameter determination, particularly covering a wide speed range. This became a very important issue in modeling and controlling the induction machine and much investigation in selecting the accurate model of induction machine has been done.

With regard to this, references [17] and [18] consider a nonlinear dynamic model of an induction machine for solving the problem of parameter identification. The model is referred to *qdo* axes fixed with the stator and is used for matching input – output behavior of the machine. The problem is formulated to find four electrical parameters (assuming $L_r = L_s$) and also mechanical system parameters such as inertia, Coulomb's friction of an inverter – fed IM. An experimental test carried out consisting of a transient regime of 1 KW two – pole induction motor from standstill to certain speed and mathematical model

is implemented with the aim of simulation of the experimental test. A cost function, defined for GAs, is computed as a weighted sum of either square or absolute differences of the output stator currents acquired experimentally and those computed by simulation at the same instant. Analytical Least Squared Technique is applied to identify the parameter and afterward is compared with GAs with the aim to select the method of identification, which gives the best results. In conclusion, the quality of the identified parameters largely depends on the excitation of the dynamics of the system and on the accuracy of the experimental test equipments used in the testing machine.

In reference [20] GAs are applied to parameter identification of induction motor. The identification was undertaken using the starting performance with four different levels of measurement noise. For the purpose of variable speed application, the motor's general mathematical model based upon Kron's voltage equations is employed to estimate the parameters, and the motor's start-up performance is used as the measurement during the identification process. For comparison, the results of a Simple Random Search (SRS) method under the same condition are given. The results show that the performance of the GAs is much better than that of the SRS technique.

Reference [19] presents a parameter identification method for induction machines based on enhanced genetic algorithm that operates on real – valued parameter sets. The induction motor model used is similar to the nonlinear model in [6]. This model is discretized and the inputs to the model are stator voltages and speed of the rotor. Fourth order Runge Kutta method is employed to determine the states of induction motor model i.e., rotor flux linkages and stator currents. It is recommended, since simulating the machine in the whole transient regime (over a long period of time) is computationally expensive, to study the effects of smaller time intervals for the given machine excitation so that computational cost of calculating fitness is reduced while still maintaining acceptable parameter estimates. The GA is run several times for each interval and the best parameter sets obtained are averaged, thus assuming the same parameters during whole transient regime of machine. In conclusion, some time intervals yield better

estimation results (while comparing estimated stator currents and currents measured by exciting the induction motor under no load) suggesting that the conditions right after machine startup (i.e. t(0) = 0.02 s) are favorable for parameter identification since at steady state, the errors are always relatively high as the current values are small and thus there is insufficient information to estimate the electric parameters.

Idea for the selection of the performance index for GAs implementation in this work comes from the references [17] and [18]. However, authors of the papers consider parameters during the whole transient regime to be constant. Another interesting reference [24] where the problem about nonlinearities of the parameters during different operating conditions is addressed is also considered in this study. The following chapters will provide more details in this. The main contributions of the work in this thesis are:

- Selection of an appropriate nonlinear model of induction machine.
- Experimental measurements of the three phase stator voltages, three phase stator currents and rotor speed of electric machines by designing Lab VIEW Software interface.
- Application of the GAs to the identification of the parameters of induction motors.
- Validation of the identified nonlinear model and comparison of the results with the tradional short – circuit, no – load and blocked – rotor test.

The GAs method is tested on several theoretical cases of induction motors for validation of the proposed approach to different sizes and also on a physical 7.5 hp induction motor.

Chapter 2

Genetic Algorithms

A genetic algorithm (GA) is an optimization technique based on the ideas of genetics and natural selection. Pioneered more than 30 years ago by computer scientist John Holland, GAs proved to be worth considering when little data is available for the algorithm from the optimization process or when the size of the problem becomes large. These problems and how to resolve them will be addressed in the thesis.

Here, GA uses a direct analogy of such a natural evolution. It presumes that a potential solution of a problem is an individual and can be represented by a set of parameters. These parameters, regarded as the genes of a *chromosome*, can be structured by a string of values in binary form. A positive value, known as fitness value, is used to reflect the degree of "goodness" of the chromosome, which is generally correlated with the objective function of the problem.

In the very first genetic algorithm proposed by Holland [13] the variables used in programs are described as a string of bits {0, 1} called *chromosomes*. For each variable to be optimized, a feasible interval is defined. The upper boundary of the interval is described with a string of ones "1 1 1 … 1", while the lower boundary consists of zeros "000…0'. However, string – based representation schemes could not be applicable to many problems, so the need for more powerful representations has been recognized for some time (De Jong [13]). The approach to simulate a genetic system in a computer using

genetic operators was proposed and examined by Fraser [13]. He simulated the evolution of 15 – bit binary string generations and calculated the percentage of individuals with acceptable phenotypes with successive generations. Holland [13] with his students studied the mathematical basis for these techniques. His original genetic algorithm sometime called as "Simple Genetic Algorithm" explores the usage of generational update schemes where each generation produces the next and dies off, so that an individual in e.g. generation *g* never has a chance to breed with one in generation *g+1*.

Starting in the 1980s, many groups began experimenting with multiple populations rather than single one proposed by Holland, more because of the availability of parallel computing scheme. Lawrence Davis [7] emphasized the importance of evaluation function as the link between the genetic algorithm and the problem to be solved. The interaction of an individual with its environment provides a measure of its fitness, and the interaction of a chromosome with an evaluation function provides a measure of fitness that the genetic algorithm uses when carrying out reproduction. Based on this theory he suggested the following to be a standard genetic algorithm:

1. Initialize a *population* of *chromosomes* .
2. Create new *chromosomes* by mating current *chromosomes* .
3. Apply *mutation* and *reproduction* as the parent *chromosomes* mate .
4. Evaluate the new *chromosomes* and insert them into the population .
5. If time is up, stop and return the best chromosome. If not, go to step three .

During the optimization process, the algorithm alters the *chromosomes*, the *genome* of the population members using *genetic operators*. They can be divided into two main categories, namely *mutation* and *crossover* operators. The *mutation* operators change the value of one or more genes in a *chromosome* while the *crossover* operators imitate breeding where two parents give birth to new offspring. The aim is to find the right genes for a population member to be able to "live" most successively in the environment described by the objective function and the constraints. It is expected from this process that a "better" *chromosome* will create a larger number of offspring, which in the end, ensures a higher chance of surviving in the subsequent generation. The cycle of evolution

is repeated until a desired termination criterion is reached. This criterion can be set by the number of evolution cycles (computational runs), or the amount of variation of individuals between different generations, or a pre-defined value of fitness.

2.1 Genetic Algorithms and Advantages

GAs have been applied to many optimization and design problems, from Power Engineering applications (Ding [13]) to a filter design (Yang [13]). One of the reasons for the popularity of this method is its reliability to work in different environments. Natural selection favors genes that increase the reproductiveness of their carriers, or in the case of the problem formulated in this thesis, the optimization produces population members that can live more successively in the environment defined by the objective function and the optimization constraints.

Compared to conventional techniques, the GAs advantages are:

- GAs are domain independent and therefore can be applied to various problems.
- GAs do not require problem specific auxiliary knowledge such as derivatives and good initial guess. This can be beneficial especially when dealing with measurement data.
- GAs can readily enforce constraints on the control variables. In contrast, enforcing constraints using conventional techniques can result in an intractable set of partial differential equations (such as those resulting from setting partial derivatives of the Langrangian equal to zero).
- GAs can simultaneously explore many points in the search space and combine knowledge, which implies that they are less likely to get stuck at a local optimum.

GA has proved to be robust to the selection of optimization parameters and noise. Also, it is not necessary to give specific rules for parameter settings; the number of mutation and crossover operators could be selected freely. De Jong [13] suggests that the real motivation of genetic algorithms is not to find the global optimum, but to have a method for allocating trails in a noisy, time – varying decision process. In GAs it is possible to make backward steps to an older design as the genes of the population contain information about the previous generations. The following table summarizes differences between GAs and the conventional optimization techniques.

Table 2.1. A comparison between conventional optimization techniques and GAs.

Property	Conventional Optimization Techniques	Genetic Algorithms
Applicability	Applicable to a specific problem domain	Applicable to variety of problem domains.
Number of points being simultaneously searched	Single Point	Multiple points
Transition from current point to the next	Deterministic	Probabilistic
Prerequisites	Auxiliary knowledge such as gradient vectors.	An objective function to be optimized
Initial guess	Provided by the user.	Automatically generated by the algorithms
Flow of control	Mostly serial	Mostly parallel
Required CPU time	Small amount	Large amount
Quality of the result	Local optimum, depends on initial guess	Global optimum is more feasible

In addition, one of the advantages of the genetic algorithms lies in handling different types of constraints:

- Domain constraints:

$$l \leq x \leq u, \text{ where } l = [l_1,...,l_n] \text{ and } u = [u_1,...,u_n]. \quad (2.1)$$

- Linear equalities:

$$Ax = b, \text{ where } A=(a_{ij}),\ b=[b_1,...,b_q],\ 1 \leq i \leq q,\ 1 \leq j \leq n$$

$$\text{and } q \text{ is the number of equalities.} \quad (2.2)$$

- Linear inequalities:

$$Cx < d, \text{ where } C=(c_{ij}),\ d=[d_1,...,d_m],\ 1 \leq i \leq m,\ 1 \leq j \leq n$$

$$\text{and } m \text{ is the number of inequalities.} \quad (2.3)$$

The domain constraints are necessary in order to limit the search space into a reasonable domain. The linear equality constraints can be eliminated and used to decrease the number of design variables by representing some of the variables as a linear combination of the others. Another benefit of linearity of the constraints is that the search space is always convex. Convexity guarantees that the offsprings produced by some of the genetic operators fulfill the constraints automatically. These advantages give genetic techniques great flexibility in solving the system identification problem.

However, like any computation technique, a GA has limitations. Available computer resources affect the selection of the optimization parameters. In many cases when user defines very high mutation rate and small population size in optimization, problems arise causing non-convergence of the genetic algorithm. A small population and a limited number of genes are observed to result in similar offspring from different parents.

Another problem with GAs is computation time. GAs can require evaluation of thousands of candidate solutions before converging to the best solution. One way to improve this is to reduce time intervals without excessively compromising parameter accuracy.

Also, since GAs are stochastic search processes, where chance plays a dominant role, it does not always guarantee good results. This can be avoided again by increasing the population size or increasing the amount of mutation. Note that GAs are not generally used for problems easily optimized using conventional techniques. For difficult optimization problems, however, the power and flexibility of the genetic techniques overcome the limitations.

2.2. Genetic Operators and How They Work?

In this chapter genetic algorithm selected in accordance to work by Goldberg [2] is described. This is an abstract view of what Genetic Algorithm does and which functions it performs:

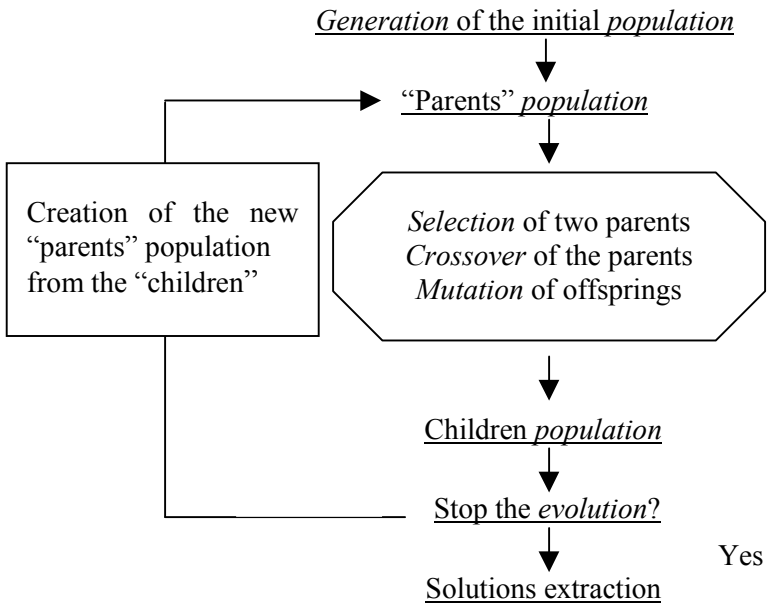


Figure 2.1. Genetic Algorithm chart.

This simple GA is composed of three genetic operators:

- Reproduction
- Crossover

- Mutation

The following discussion will describe each of these operators in details through one mathematical example. This is by no means a comprehensive review of literature [27]. Basically, the mechanics of this algorithm starts with a randomly generated initial population of strings to be able to generate successive populations of strings afterwards. A population of these strings is created randomly in the first step. It should be noted that population size is one of the most important parts of the GA. It has been found that too large population slows down the optimization, while a small population does not utilize the genetic operators effectively. The population size affects the convergence properties of a stochastic algorithm as shown in work by Dixon [13].

Mathematical function given below is a good example for explaining the roles of each of the genetic operators. Here, the goal is to find the maximum of the following:

$$f(x,y) = (x-7)^2 + (y-3)^2 \tag{2.4}$$

where $$x, y \in [0,7] \tag{2.5}$$

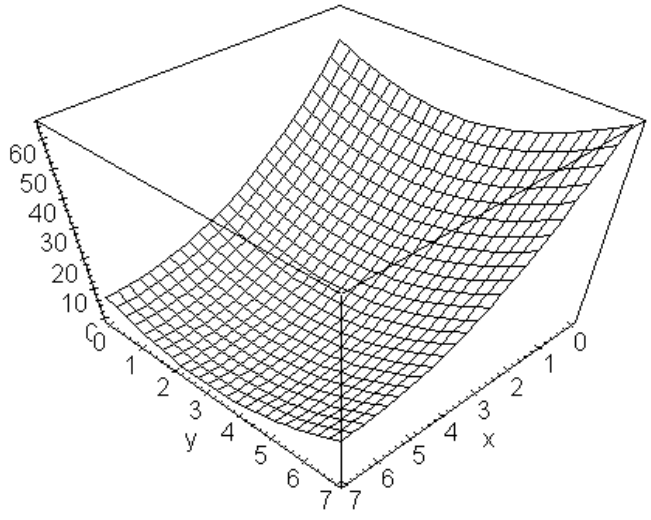


Figure 2.2. Domain of x and y in the mathematical example.

2.2.1. Initial Population

The first step in solving the problem is to generate initial population as binary presentation of six bits. Population size is defined as n = 4 (just to demonstrate effectiveness of the proposed algorithm).

The following table shows a possible randomly generated population of four six – bit numbers. The meaning of the string numbers is e.g. 100001 = > x = 100 bit equivalent to 4 and y = 001 bit, which is equivalent to 1.

Table 2.2. Example of an initial population.

No.	String	
1.	100001	(x = 4, y = 1)
2.	001100	(x = 1, y = 4)
3.	110010	(x = 6, y = 2)
4.	000100	(x = 0, y = 4)

2.2.2. Evaluation Function

In order to get solution from GA at each generation, the next step is to supply genetic algorithm with an evaluation function. According to this function, GA will be able to calculate fitness value for each solution. One-way to do this is to map the objective function to"fitness function": a non – negative merit (Goldberg [2]).

Two cases must be distinguished:

When the objective is maximization of a utility or profit function *f(x,y)*, the problem of negative *f(x,y)* values can be eliminated by transforming fitness according to the equation:

$$F(x,y) = f(x,y) + C_{min} \quad \text{when} \quad f(x,y) + C_{min} > 0, \tag{2.6}$$

$$\text{otherwise } F(x,y) = 0$$

When the objective is minimization of a cost function *g(x,y)* then the problem should be maximized to assure that the measure is non –negative by using the following cost – to – fitness transformation:

$$F(x,y) = C_{max} - g(x,y) \text{ when } C_{max} - g(x,y) > 0, \quad (2.7)$$

$$\text{otherwise } F(x,y) = 0$$

C_{min} or C_{max} may be chosen as an input coefficient, as the absolute value of the worst f – value, respectively the largest g – value in the current or last k generations, or as a function of the population variance.

In this example C_{min} = 0 because the objective function *f(x,y)* will never be negative. Thus $F(x,y) = f(x,y)$ (fitness is evaluation). The following table shows the evaluation of the four initial strings.

Table 2.3. Evaluation of the initial population.

No.	String	(*x*, *y*)	Fitness
1.	100001	(4, 1)	13
2.	001100	(1, 4)	37
3.	110010	(6, 2)	2
4.	000100	(0, 4)	50

2.2.3. Reproduction

The next step is to perform the reproduction, a process in which individual strings are copied according to their objective function values $f(x,y)$. The meaning of copying string according to their fitness values means that strings with a higher value have a higher probability of contributing one or more offspring in the next generation.

One of the easiest ways to implement reproduction into a genetic algorithm is to create a biased roulette wheel where each current string in the population has a roulette wheel slot sized in proportion to its fitness. To reproduce means to spin the weighted roulette wheel as many times as the population size (for example four times in this example). (Goldberg [2]). The next step is to divide individual's fitness value with the average of all fitness values and then calculate the expected count of this individual in the next generation. For this example the *average of all the fitness functions* is 25.5, so the expected count of individual one in the next generation is 13/25.5 = 0.51. Other expected counts are shown in Table 2.4 along with the normalized fitness values, which are equal to the fitness values divided by the total sum of all fitness values (e.g. 102 in this example), multiplied by 100%.

The normalized fitness gives of an individual to be chosen as a parent. A method to actually select an individual as a parent is to use a sum function $S_i = \sum_{j=1}^{i} f_j$, which gives the sum of all fitness values from individual *one* to individual *i*, and randomly and uniformly choose an integer between 0 and the sum of all fitness values. The first individual whose S_i is equal or greater than this integer will be chosen as a parent. The S_i values are shown in the Table 2.4.

Suppose the randomly chosen number is 53 then individual 4 will be chosen as a parent because S_4 is the first value that succeeds 53. This procedure will be repeated until all four parents are obtained. The parents are shown in the Table 2.4.

Table 2.4. Reproduction results.

No.	String	(*x,y*)	Fitness	Normalised	Si	Expected count	Actual
1.	100001	(4, 1)	13	12.7 %	13	0.51	1
2.	001100	(1, 4)	37	36.3 %	50	1.45	1
3.	110010	(6, 2)	2	2.0 %	52	0.08	0
4.	000100	(0, 4)	50	49.0 %	102	1.96	2

2.2.4. Crossover

Once a string has been selected for the reproduction, an exact replica of the string is made. This string is then entered into a mating pool, a new population, for further genetic operator action. When the two parents have been selected, the genetic algorithm combines them to create two new offspring. The crossover operator performs combination. One – point crossover is the oldest and most commonly used and will be explained in this work. There are some alternatives to one – points crossover among which there are two – point crossover and uniform crossover (Syswerda [13]), which can avoid some disadvantages that occur in one – point crossover.

In a single-point crossover position *k[1,2,...,l*-1], where k is the integer which lies in between 1 and string length less one. Two new strings are created by exchanging all characters between positions *k* and *l*. Figure 2.4. illustrates this process. For example, if one considers the following two individuals with 11 binary variables each:

individual 1 0 1 1 1 0 0 1 1 0 1 0

individual 2 1 0 1 0 1 1 0 0 1 0 1

The chosen crossover position is:

crossover position 5

After crossover the new individuals are created:

offspring 1 0 1 1 1 0| 1 0 0 1 0 1

offspring 2 1 0 1 0 1| 0 1 1 0 1 0

The role of the crossover operator is to allow the advantageous traits to be spread throughout the population in order that the population as a whole benefits from this chance discovery (Parker [13]). A proper selection of a crossover operator plays an important role in the success of an application using a genetic algorithm (Davis [13]).

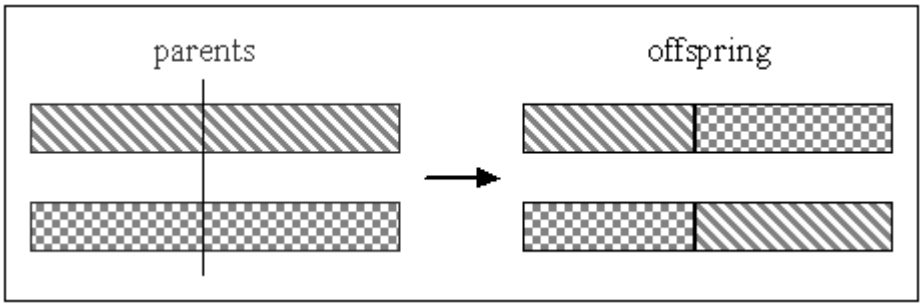


Figure 2.3. Single – point crossover.

2.2.5. Mutation

Another way to cause *chromosomes* created during a reproduction to differ from their parents is called *mutation.* Many researchers believe there is a need for this genetic operator, since even though reproduction and crossover effectively search and recombine the notions, occasionally they may become overzealous and lose some potentially useful genetic material (*one's* or *zero's* for example at some particular locations). In artificial genetic systems, the mutation operator protects against such an irrecoverable loss. In the GA used in this work, mutation is occasional (with small probability) random alteration of the value of a string position. It is defined by the user and usually the lower the rate is, the less chance the chromosomes of the children differ from those of their parents.

Table 2.5. Mutation operator.

Before mutation	After mutation
10**1**100	10**0**100

In the example being tested, only bit 4 (the third bit from the left) is being changed through mutation, as can be seen from the Table 2.5.

From the Table 2.6, it can be noted that a new string with high fitness has appeared. Thus, sum of the fitness values has increased from 102 to 163 and the average has increased from 25.5 to 40.8, and all this in one generation. Mutation rates as stated are

small in natural populations, leading to conclusion that mutation is appropriately considered as a secondary mechanism of genetic algorithm adaptation.

Table 2.6. New population and fitness after crossover and mutation.

No	Selected Parents	After Crossover	After	New Fitness
1.	10|0001	101100	10**0**100	10
2.	00|1100	000001	000001	53
3.	00010|0	000100	000100	50
4.	00010|0	000100	000100	50

Following the reproduction, crossover and mutation, in this simple demonstration example, it can be concluded that strings 1 and 2 were selected once (average fitness), string 3 was not selected (low fitness) and string 4 was selected twice (high fitness). Crossover provided the high fitness string 000001 (string 2) but also the low – fitness string 101100 (string 1) in which a mutation took place which, in this case, increased the fitness.

Other genetic operators and reproductive plans have been abstracted from the study of biological example. However, the three examined in this section, reproduction, simple crossover, and mutation, have proved to be both computationally simple and effective in attacking a number of important optimization problems. There are some advanced techniques to improve convergence and work of GAs and they will be just summarized in this work.

2.3. Advanced Techniques

Although the simple genetic algorithm is a very powerful tool on its own, there are ways to improve its techniques, resulting in a faster convergence. They are classified as follows:

- *Hybridization*, a way to let the genetic algorithm outperform the other algorithms by incorporating the positive features of those algorithms into the genetic algorithm (Davis [7]).

- *Fitness techniques*, (Goldberg [2], Davis [7]). A way to stop the extraordinary individuals in a population to create a premature convergence without guarantee of optimality. There are three ways to solve this problem by using *linear scaling*, *windowing* and *linear normalization*.

- *Elitism*, technique of copying the best member of each generation a number of times into the succeeding generation.

- *Advanced crossover methods*, (Goldberg [2], Davis [7]). The following techniques combine the most successfully certain combinations of features encoded on chromosomes (one of the things that one – point crossover can not perform well). These techniques are classified as *two – point crossover*, *uniform crossover*, *partially mixed crossover* and *uniform – based crossover*.

- *Advanced mutation methods,* (Davis [7]). Search for better solutions in the region of the best current solutions. They are defined as *uniform mutation*, *boundary mutation* and *non – uniform mutation.*

2.4. Genetic Algorithm in Optimization of Electric Machines

The problem of optimizing the parameters of electric machines from its startup transient is a specific subgroup of the very general class of system identification problems. GAs proved to be generally applicable to this type of problem even though there are many alternatives in the literature. The problem can be formulated as follows [14]:

Define a mathematical model for the observations, which is a function of unknown parameters θ such that the sum of differences between the observations and the model is as small as possible.

Consider the electric machine as a system described as a state space model of the form:

$$\dot{X} = f(X, U, \dot{U}, \theta, Z), \qquad Y = g(X, U, \dot{U}, \theta, Z) \tag{2.8}$$

where, X is the state vector of dimension n; U the input vector of dimension r; Y the output vector of dimension m; θ is the dependent parameter vector of dimension m; Z the independent parameter vector of dimension w. Under steady - state condition:

$$0 = f(X_o, U_o, 0, \theta, Z), \qquad Y = g(X_o, U_o, 0, \theta, Z) \tag{2.9}$$

where , (2.9) denotes a steady state value.

Presumably, U_o and Y_o are measurable and known, as it will be shown later in this work, X_o and θ are therefore dependent on Z according to equation (2.8). This means that the dynamic equation (2.9) is solvable after Z is estimated. More general, let Z be a parametric vector that includes also model structure parameters such as the order of dynamic model; let S be a set of admissible parameters, i.e. $Z \in S$, which guarantee that a chosen model equation has a solution; let the observation space be O. Then the parameter – to – output mapping is written as:

$$\Phi : S \rightarrow O \tag{2.10}$$

The major procedure of an optimization based parameter estimation method is to search the best parameter vector Z^* in the search space S, which minimizes an error function E, i.e.

$$E^* = minimize_{\ Z=Z^*,\ Z \in S}\ E(Z) \tag{2.11}$$

The error function E is usually taken as a nonnegative and monotonically increasing function of output error:

$$E = \int_{to}^{T} J\left(\left\| Y_m(t) - Y_c(t) \right\|\right) dt \qquad (continuous \quad time) \tag{2.12}$$

$$E = \sum_{k=0}^{N} J\left(\left\| Y_m(k) - Y_c(k) \right\|\right) \qquad (discrete \quad time) \tag{2.13}$$

where, $[t_0, T]$ is the observation interval; $\| \ \|$ denotes a norm; $J(e)$ is a monotonically increasing function; k denotes the k^{th} time sample; N is the number of all samples; $(\ )_m$ denotes the measured (or true) values and $(\ )_c$ computed values. The most widely used forms of $J(e)$ are the square function, or absolute function, the square root function or their combinations. If an average model is required to fit a series of tests, the error function may be taken as a sum of all the test errors.

Better parameters generally result in less error function. In GAs, larger fitness would reproduce more offspring. This will most likely lead to better parameter estimation. Noting that the error function is always positive, fitness f is usually chosen as an inverse of the error function, so searching the minimum error function is equivalent to searching maximum fitness function:

$$f(Z) = 1/E(Z), \quad maximize \ f(Z)_{Z=Z^*, Z\in S} \Leftrightarrow minimize \ E(Z)_{Z=Z^*, Z\in S} \tag{2.14}$$

It will be shown later that by using this approach to identify parameters, the error between the response of an electric model with the actual parameters and GAs identified parameters can be minimized very well. This can be proved by simulating the output responses with both measured and estimated parameters. One good point here is again insensitiveness of the GAs to measurement noise, which is the common case in practice.

Chapter 3

Induction Motor Model

This chapter investigates the startup behavior of induction machine and the dynamics associated with the transients of machine itself. A nonlinear induction motor model is described. Transformation to an arbitrary, rotating reference frame is introduced to interpret the induction motor model. A computer representation is developed according to [6], which can be conveniently changed to simulate the symmetrical induction machine in any reference frame. Induction motor simulation is performed for the following modes of operation:

- *Balanced conditions*
- *Unbalanced stator voltages*
- *Unbalanced rotor resistors*

The effectiveness of an induction motor model is demonstrated and tested on several induction machines during free acceleration.

Following this study, the traditional DC resistance, no load, and blocked rotor tests are discussed for determination of induction motor parameters. The data needed for computing the performance of a three-phase induction motor is acquired from these tests.

Standard test parameter identification of 7.5 hp induction motor is obtained based on this test and it can be found in the last part of this chapter.

3.1 Arbitrary reference frame transformations

3.1.1. Balanced Conditions

Three - phase induction motors are designed and manufactured such that all three phases of the winding are carefully balanced with respect to the number of turns, placement of the winding, and winding resistance. Due to the sinusoidal variation of mutual inductances with respect to the displacement angle θ [6], time – varying coefficients will appear in the voltage equations. This can be eliminated by transforming the voltages and currents of both, the stator and rotor to a common frame of reference. By doing this, analysis of machinery is greatly simplified and can take sets of variables from the fixed frame to any arbitrary rotating frame.

The two common reference frames used in the analysis of induction machine are the stationary and synchronously rotating reference frames. For power system studies, induction machine loads are often simulated in the synchronous rotating reference frame, as in the case of this work.

It is rather convenient to derive the equations of the induction machine first in the arbitrary reference frame, which is rotating at a speed ω, in the direction of the rotor rotation. The equations of transformation to this reference frame at angle $\beta\ (t)$ are given as:

$$K = \frac{2}{3}\begin{pmatrix} \cos\beta & \cos(\beta - \frac{2\pi}{3}) & \cos(\beta + \frac{2\pi}{3}) \\ \sin\beta & \sin(\beta - \frac{2\pi}{3}) & \sin(\beta + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{3.1}$$

The transformation angle $\beta(t)$, between q – axis of the reference frame rotating at speed ω and the a – axis of the stationary stator winding may be expressed as:

$$\beta(t) = \int_0^t \omega(t)dt \; + \; \beta(0) \tag{3.2}$$

where ω is the base electrical frequency.

The equations of the machine either in the stationary or synchronously rotating reference frames can simply be obtained by setting the speed of the arbitrary reference frame, ω, to zero and ω_e, respectively. Equation (3.1) and its inverse transformation along with (3.2) define a transformation to a frame that rotates synchronously with three phase sources in the assigned frame. For instance, three phase source given as,

$$V_{abc} = V_o \begin{pmatrix} cos(\omega \cdot t) \\ cos(\omega \cdot t - \frac{2\pi}{3}) \\ cos(\omega \cdot t + \frac{2\pi}{3}) \end{pmatrix} \cdot u(t) \tag{3.3}$$

is conveniently expressed in the synchronously rotating reference frame as a function of V_q and V_d. Thus,

$$V_{qdo} = V_o \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \cdot u(t) \tag{3.4}$$

The transformation equation from *abc* to *qdo* reference frame is given by,

$$V_{qdo} = K \cdot V_{abc} \tag{3.5}$$

This is an important simplification of the drive typically applied to an induction motor. The synchronously rotating reference frame is often referred to as the "*qdo*" frame. Note that under balanced conditions, where

$$i_a + i_b + i_c = 0 \tag{3.6}$$

$$V_a + V_b + V_c = 0 \tag{3.7}$$

only the currents i_q and i_d (V_q and V_d) need to be specified. Also, for balanced conditions V_{qs} and V_{ds} in the synchronously rotating reference frame are constant and could be easily implemented as input voltages in a computer simulation.

3.1.2. Unbalanced Stator Voltages

When the line voltages applied to a three - phase induction motor are not equal, unbalanced currents in the stator windings will result. This can occur in a power system due to a fault or a switching malfunction, which may cause unbalanced conditions to exist for a considerable period of time. Usually, a small percentage voltage unbalance will result in a much larger percentage current unbalance. Consequently, the temperature rise of the motor operating at a particular load and percentage voltage unbalance will be greater than for the motor operating under the same conditions with balanced voltages.

The effect of unbalanced voltages on a three - phase induction motors is equivalent to the introduction of a "negative sequence voltage" having a rotation opposite to that occurring with balanced voltages. This negative sequence voltage produces in the air gap a flux rotating against the rotation of the rotor, tending to produce high currents. A small negative sequence voltage may produce in the windings currents considerably in excess of those present under balanced voltage conditions.

In general the voltage unbalance (or negative sequence voltage) is defined as follows:

$$Percent\ Voltage\ \ Unbalance = 100 \cdot \frac{maximum\ \ voltage\ \ deviation\ \ from\ \ average\ \ value}{average\ \ voltage}$$

When the stator voltages become unbalanced, sinusoidal variations will appear in the d and q quantities regardless of the choice of reference frame. In this work, to simulate a system unbalance, the applied voltages are changed, at the instant of maximum *Vas*, from rated to:

$$\begin{aligned} V_{as} &= 0.77\cos(\omega_e \cdot t - 5.1^\circ) \\ V_{bs} &= 0.57\cos(\omega_e \cdot t - 9.3^\circ) \\ V_{cs} &= 0.98\cos(\omega_e \cdot t + 139.2^\circ) \end{aligned} \tag{3.8}$$

The dynamic performance of an induction machine during a temporary unbalance in the stator voltages is shown in Figure 3.1 on 10 hp p.u. induction machine. Following the change in applied voltages, the machine slows down since the torque load remains constant at the base torque values. Soon after steady state unbalanced conditions are reached, rated voltages are reapplied.

3.1.3. Unbalanced rotor resistors

In some applications where it is necessary to accelerate a large inertia mechanical load, a wound rotor induction machine equipped with a variable external rotor resistors is often used. As the speed of the machine increases, the value of the external rotor resistances decrease proportionally so as to maintain nearly maximum electromagnetic torque during most of the acceleration period. However, one should be careful in order not to unbalance the external rotor resistors during this process, otherwise a torque pulsation of twice the slip frequency occurs. For the purpose of analyzing unbalanced rotor resistors in induction machines, the following rotor resistances were considered for simulation,

$$R_{ar} = 0.2 \ \ pu, \ \ R_{br} = 0.1 \ \ pu, \ \ R_{cr} = 0.05 \ \ pu \tag{3.9}$$

The acceleration characteristics of the example 10 hp induction machine with unbalanced rotor resistors and a constant load torque are shown in Figure 3.2. Balanced, rated stator

voltages are applied and the torque load is maintained constant at 1.0 pu. The machine used in this case is the 6 – pole, 3 – phase, 220 – volt (line-to line), 10 hp, 60 Hz. The parameters expressed in per unit are:

Table 3.1. Parameters of 10 hp induction motor.

R_s	X_{ls}	X_m	$X_{lr'}$	$R_{r'}$
0.0453	0.0775	2.042	0.0322	0.0222

The dynamic performance of an induction machine during unbalanced conditions, unbalanced stator voltages and rotor resistors respectively is shown in the Figures 3.1 and 3.2.

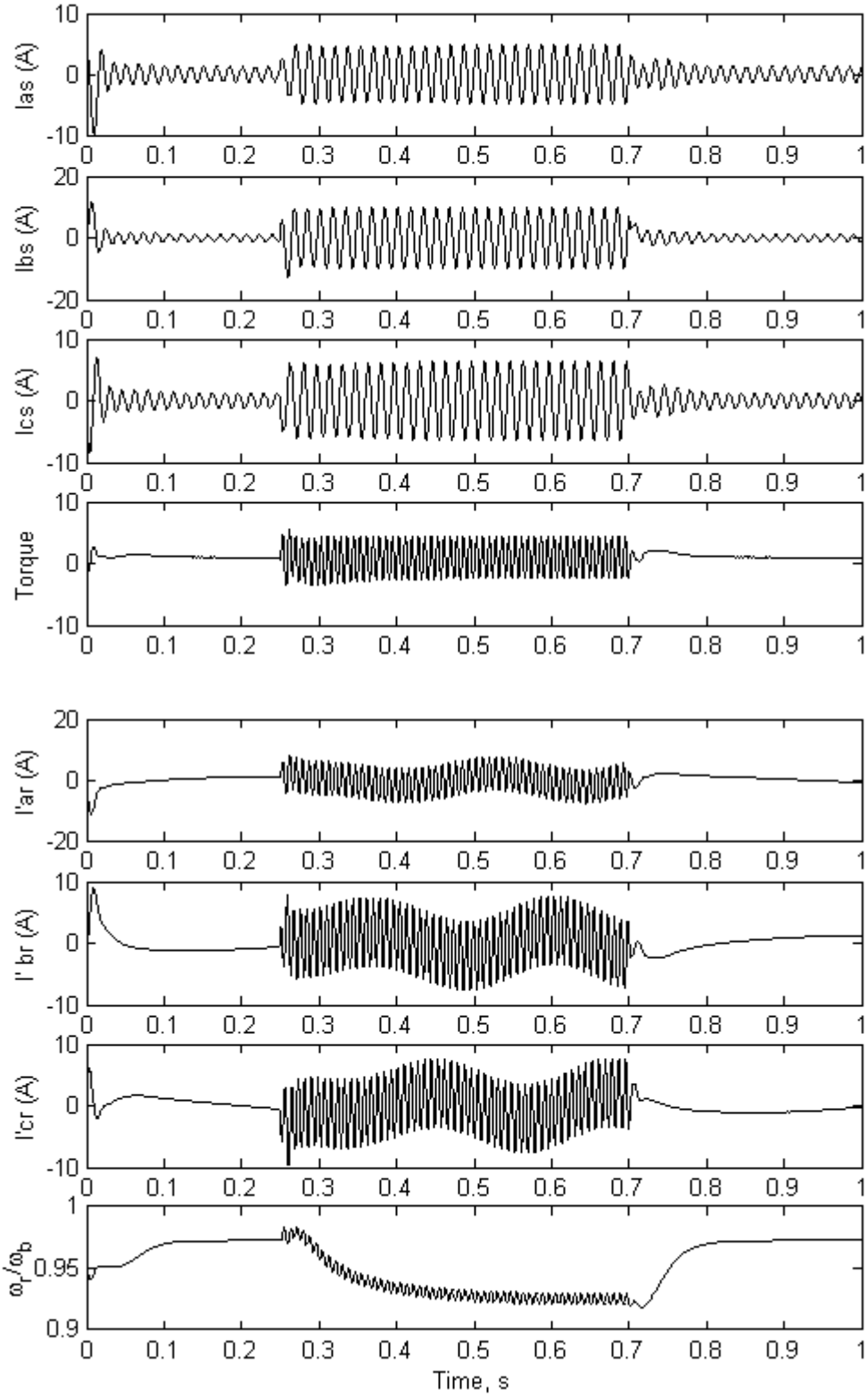


Figure 3.1. Performance of a 10 hp induction machine with unbalanced stator voltages.

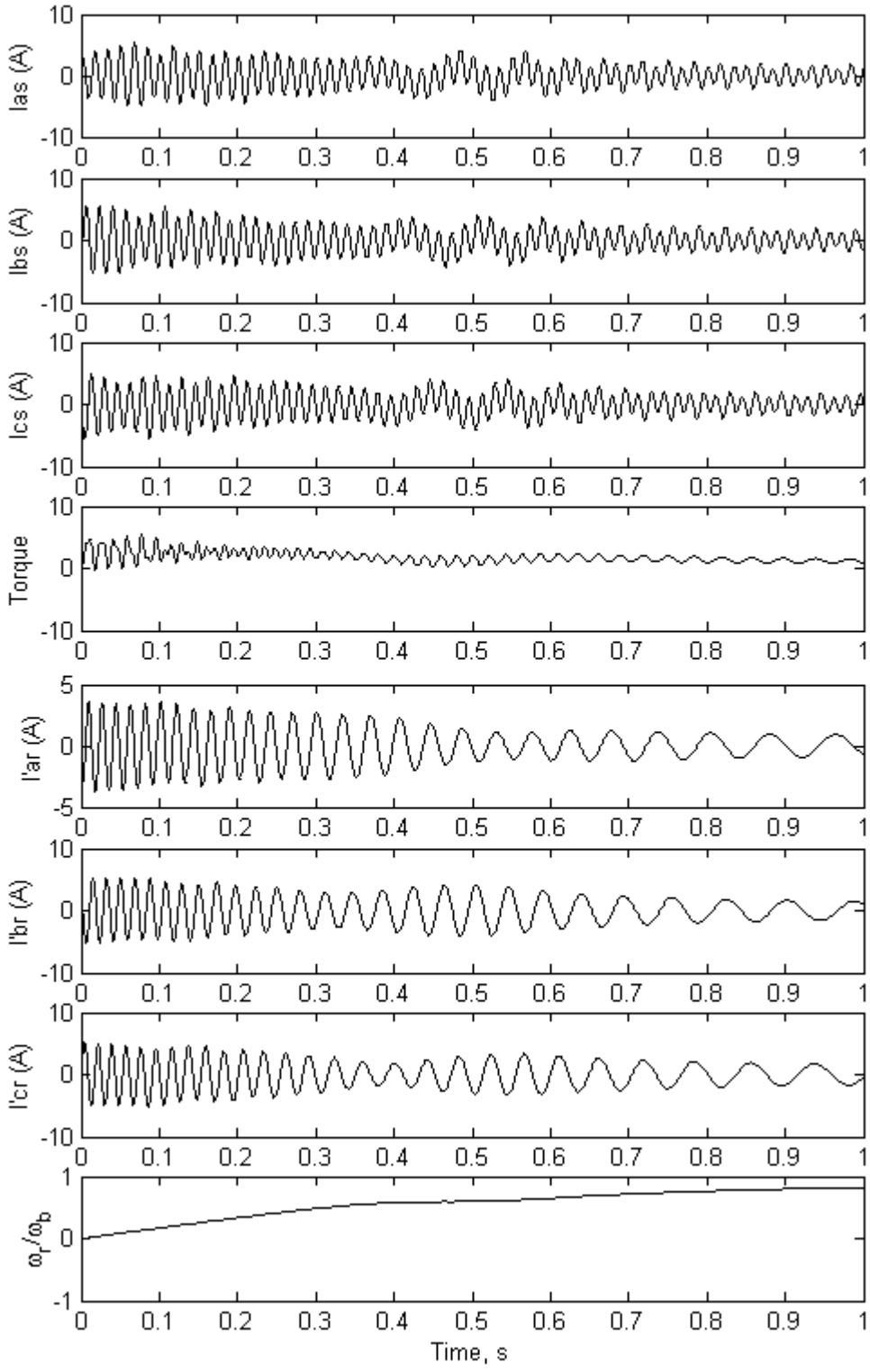


Figure 3.2. Performance of a 10 hp induction machine with unbalanced rotor resistors.

3.2. Induction Motor Model

In this thesis, nonlinear model same as in [6] is used. A three – phase, balanced machine is assumed. Figure 3.3 shows the equivalent circuit of the induction model.

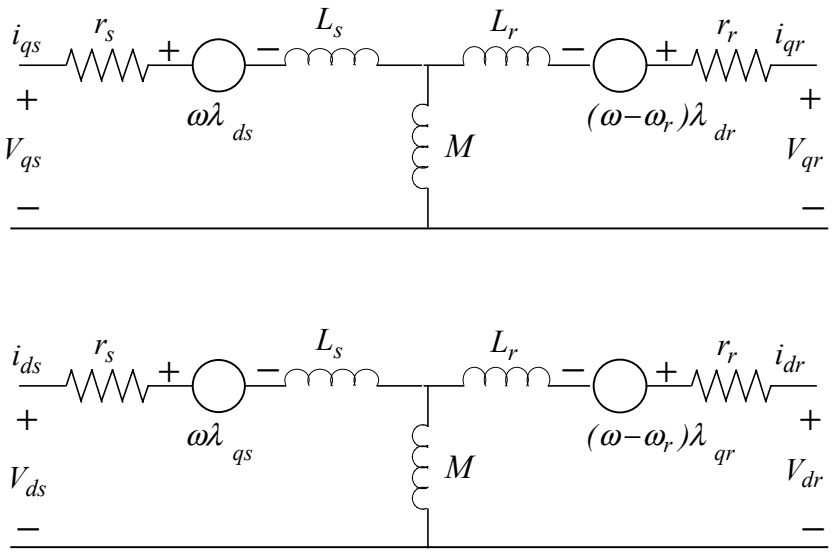


Figure 3.3. Induction motor equivalent circuit model.

The voltage Equations (3.10) are written in terms of flux linkages. By selecting flux linkages as independent variables, each *q*- and *d*- voltage equation contains only one derivative of flux linkage. This property makes it more suitable to implement a computer simulation of an induction machine with flux linkages as state variables rather than with currents.

Since the induction motor is an electromechanical device, it implies that the model also requires expressions for the electromagnetic torque and the speed of the machine. Equations (3.15 and 3.16) express the electromagnetic torque in terms of the currents and flux linkages respectively, and Equation (3.17) determines the rotational speed from the machine torque, load torque, and moment of inertia. It should be noted that the model neglects core loss as well as friction and windage loss.

$$\begin{bmatrix} V_{qs} \\ V_{ds} \\ V_{os} \\ V_{qr}' \\ V_{dr}' \\ V_{or}' \end{bmatrix} = \begin{bmatrix} \frac{r_s X_{rr}}{D} + \frac{p}{\omega_b} & \frac{\omega}{\omega_b} & 0 & -\frac{r_s X_M}{D} & 0 & 0 \\ -\frac{\omega}{\omega_b} & \frac{r_s X_{rr}'}{D} + \frac{p}{\omega_b} & 0 & 0 & -\frac{r_s X_M}{D} & 0 \\ 0 & 0 & \frac{r_s}{X_{ls}} + \frac{p}{\omega_b} & 0 & 0 & 0 \\ -\frac{r_r' X_M}{D} & 0 & 0 & \frac{r_r' X_{ss}}{D} + \frac{p}{\omega_b} & \frac{\omega - \omega_r}{\omega_b} & 0 \\ 0 & -\frac{r_r' X_M}{D} & 0 & -\frac{\omega - \omega_r}{\omega_b} & \frac{r_r' X_{ss}}{D} + \frac{p}{\omega_b} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{r_r'}{X_{lr}'} + \frac{p}{\omega_b} \end{bmatrix} \begin{bmatrix} \Psi_{qs} \\ \Psi_{ds} \\ \Psi_{os} \\ \Psi_{qr}' \\ \Psi_{dr}' \\ \Psi_{or}' \end{bmatrix} \tag{3.10}$$

where, $$X_{ss} = X_{ls} + X_M \tag{3.11}$$

$$X_{rr}' = X_{lr}' + X_M \tag{3.12}$$

$$D = X_{ss} \cdot X_{rr}' - X_M^2 \tag{3.13}$$

$$\begin{bmatrix} i_{qs} \\ i_{ds} \\ i_{os} \\ i_{qr}' \\ i_{dr}' \\ i_{or}' \end{bmatrix} = \frac{1}{D} \begin{bmatrix} X_{rr}' & 0 & 0 & -X_M & 0 & 0 \\ 0 & X_{rr}' & 0 & 0 & -X_M & 0 \\ 0 & 0 & \frac{D}{X_{ls}} & 0 & 0 & 0 \\ -X_M & 0 & 0 & X_{ss} & 0 & 0 \\ 0 & -X_M & 0 & 0 & X_{ss} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{D}{X_{lr}'} \end{bmatrix} \begin{bmatrix} \Psi_{qs} \\ \Psi_{ds} \\ \Psi_{os} \\ \Psi_{qr}' \\ \Psi_{dr}' \\ \Psi_{or}' \end{bmatrix} \tag{3.14}$$

The electromagnetic torque developed by the machine expressed in terms of currents is:

$$T_e = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right) M(i_{qs} \cdot i_{dr}' - i_{ds} \cdot i_{qr}') \tag{3.15}$$

where *Te* is positive for motor action.

Here, *P* is the number of poles, *M* is the magnetizing inductance (not X_M), and the rotor currents are as reflected to the stator. The above expression for torque can be easily written in terms of flux linkages per second and currents. Hence,

$$T_e = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)\frac{X_M}{D\omega_b}(\Psi_{qs} \cdot \Psi_{dr}' - \Psi_{ds} \cdot \Psi_{qr}') \tag{3.16}$$

where *D* is defined by Equation (3.13).

In this work, saturation is not considered. From basic mechanics, the action of a torque *T* is to produce an angular acceleration $d\omega/dt$. So, in order to obtain the dynamic characteristics of induction machine, it is necessary to solve the following equation, which simultaneously relates torque and speed. Thus,

$$T = (\frac{2}{P})J \cdot p\omega_r + T_L \tag{3.17}$$

where T_L is the load torque and *J* is the total inertia.

The above equation can be rewritten in a better way for computer simulation as:

$$\frac{\omega_r}{\omega_e} = (T - T_L)/\left(\frac{2}{P}\right)J\omega_e p \tag{3.18}$$

It is advisable when doing computer simulations of electric machines to eliminate sinusoidal quantities whenever possible. Therefore, unless it is necessary to retain either the stator or rotor variables, the simulation in the synchronously rotating reference frame is particularly desirable for studying dynamic behavior of machine.

3.2 Simulation of Induction Machine

A computer simulation of the induction motor is conveniently accomplished by solving for the flux linkages per second in terms of the voltages applied to the machine. The *qdo* voltages at stator terminals are considered as the input to the model equations of the induction machine to obtain the corresponding *qdo* currents in the given reference frame. Thus, the *qdo* currents are regarded as the output variables of the model.

In the model representation of induction machine, the following set of equations for currents is solved first [8]. Hence,

$$i_{qs} = \frac{1}{X_{ls}}(\Psi_{qs} - \Psi_{mq}) \tag{3.19}$$

$$i_{ds} = \frac{1}{X_{ls}}(\Psi_{ds} - \Psi_{md}) \tag{3.20}$$

$$i_{qr}' = \frac{1}{X'_{lr}}(\Psi'_{qr} - \Psi_{mq}) \tag{3.21}$$

$$i_{dr}' = \frac{1}{X'_{lr}}(\Psi'_{dr} - \Psi_{md}) \tag{3.22}$$

where,

$$\Psi_{mq} = X_m \cdot (i_{qs} + i_{qr}') \tag{3.23}$$

$$\Psi_{md} = X_m \cdot (i_{ds} + i_{dr}') \tag{3.24}$$

By eliminating currents in the equations above, resulting voltage equations can be solved for Ψ_{qs}, Ψ_{ds}, Ψ_{qr}', Ψ_{dr}' , which follows directly from the following equations:

$$\Psi_{qs} = \frac{\omega_e}{p} \cdot [V_{qs} - \frac{\omega}{\omega_e} \cdot \Psi_{ds} + \frac{r_s}{X_{ls}} \cdot (\Psi_{mq} - \Psi_{qs})] \tag{3.25}$$

$$\Psi_{ds} = \frac{\omega_e}{p} \cdot [V_{ds} + \frac{\omega}{\omega_e} \cdot \Psi_{qs} + \frac{r_s}{X_{ls}} \cdot (\Psi_{md} - \Psi_{ds})] \tag{3.26}$$

$$\Psi_{qr}' = \frac{\omega e}{p} \cdot [V_{qr}' - (\frac{\omega - \omega_r}{\omega_e}) \cdot \Psi_{dr}' + \frac{r_r'}{X_{lr}'} \cdot (\Psi_{mq} - \Psi_{qr}')] \tag{3.27}$$

$$\Psi_{dr}' = \frac{\omega_e}{p} \cdot [V_{dr}' + (\frac{\omega - \omega_r}{\omega_e}) \cdot \Psi_{qr}' + \frac{r_r'}{X_{lr}'} \cdot (\Psi_{md} - \Psi_{dr}')] \tag{3.28}$$

where,
$$\Psi_{mq} = X_{mq} \cdot (\frac{\Psi_{qs}}{X_{ls}} + \frac{\Psi_{dr}'}{X_{lr}'}) \tag{3.29}$$

$$\Psi_{md} = X_{md} \cdot (\frac{\Psi_{qs}}{X_{ls}} + \frac{\Psi_{dr}'}{X_{lr}'}) \tag{3.30}$$

X_{mq} and X_{md} are equal and defined as:

$$X_{mq} = X_{md} = \frac{1}{\frac{1}{X_m} + \frac{1}{X_{ls}} + \frac{1}{X_{lr}'}} \tag{3.31}$$

Since no external voltages are applied to the rotor windings, it follows that $V_{qr}' = V_{dr}' = 0$. The following figures 3.2, 3.3, 3.4 and 3.5 show the Matlab simulation of the different induction machines. Parameters of the induction machine are listed in Table 3.2. They are determined by using traditional standard tests and can be found in [6]. The simulation code along with Simulink blocks of induction machine are listed in Appendix B.

By starting from zero speed and applying the voltages as the input to the model, the acceleration of the machine can be obtained either with or without a load on the shaft. As can be seen from the figures, any type of the motor is subjected to pulsating torques during the startup. For example, a four-pole, 50 hp motor during starting condition will have oscillating torque with a peak of about 1600 Nm. One advantage of this model for

the motor is the ability to look at variables that would be difficult or impossible to measure. Figure 3.4 shows the stator phase currents as a function of time during the free acceleration of the different types of induction motors. Their frequency is essentially constant at 60 Hz, but the amplitude is much larger than rated current until the machine reaches breakdown torque. Once the machine reaches synchronous speed, the motor draws only a small current to provide the excitation and the losses of the stator and rotor windings.

Table 3.2. Induction machine parameters for different machines

Hp	Volts	Rpm	T_B (Nm)	I_B(abc)	r_s	X_{ls}	X_M	$X_{lr'}$	$r_{r'}$	J kgm2
3	220	1710	11.9	5.8	0.435	0.754	26.13	0.754	0.816	0.089
50	460	1705	198	46.8	0.087	0.302	13.08	0.302	0.228	1.662
500	2300	1773	1.98 x 10e3	93.6	0.262	1.206	54.02	1.206	0.187	11.06
2250	2300	1786	8.9 x 10e3	421.2	0.029	0.226	13.04	0.226	0.022	63.87

Another interesting point lies in difficulties in measuring the rotor currents, particularly in the case of squirrel – cage induction motors. At the start, the rotor currents have a 60 Hz frequency, but the frequency drops as the motor accelerates, reaching very low frequencies as the motor nears synchronous speed. Once the motor reaches the synchronous speed there is no relative motion between the rotor bars and the rotating magnetic field. This implies that the current in the rotor bars approaches zero.

Even though it is not practicable to account for all possible situations, the computer representation presented here is general and can be modified to accommodate many practical problems, including simulations involving several machines.

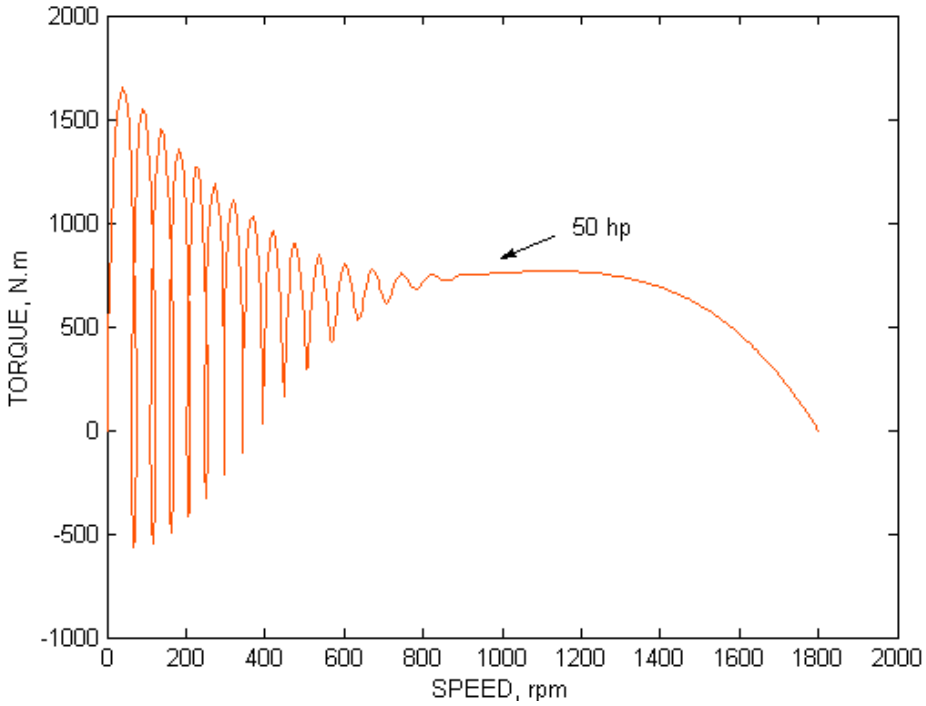


Figure 3.4. Torque during starting of 50 hp induction machine.

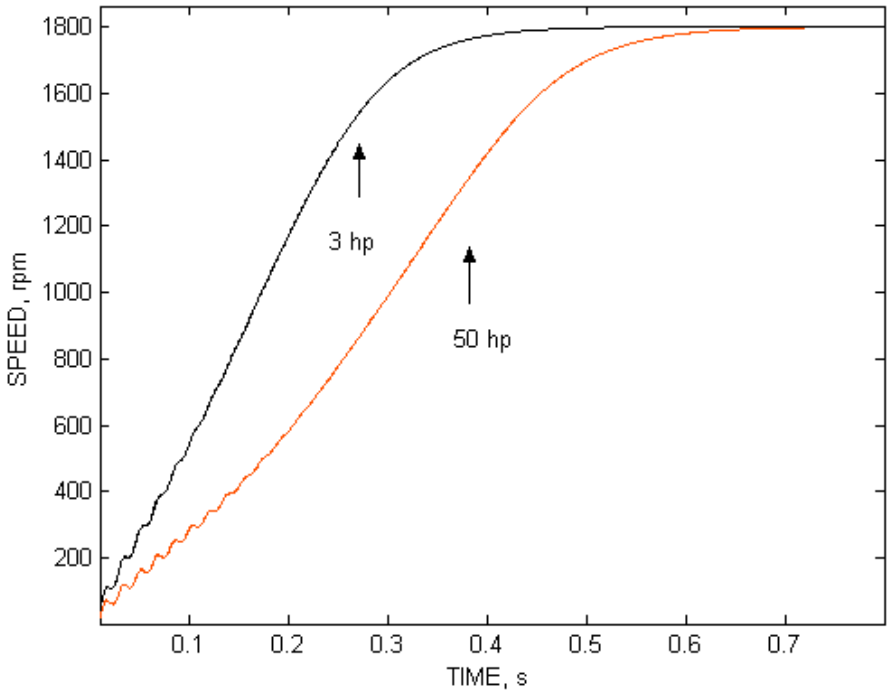


Figure 3.5. Rotor speed during starting of different induction motors.

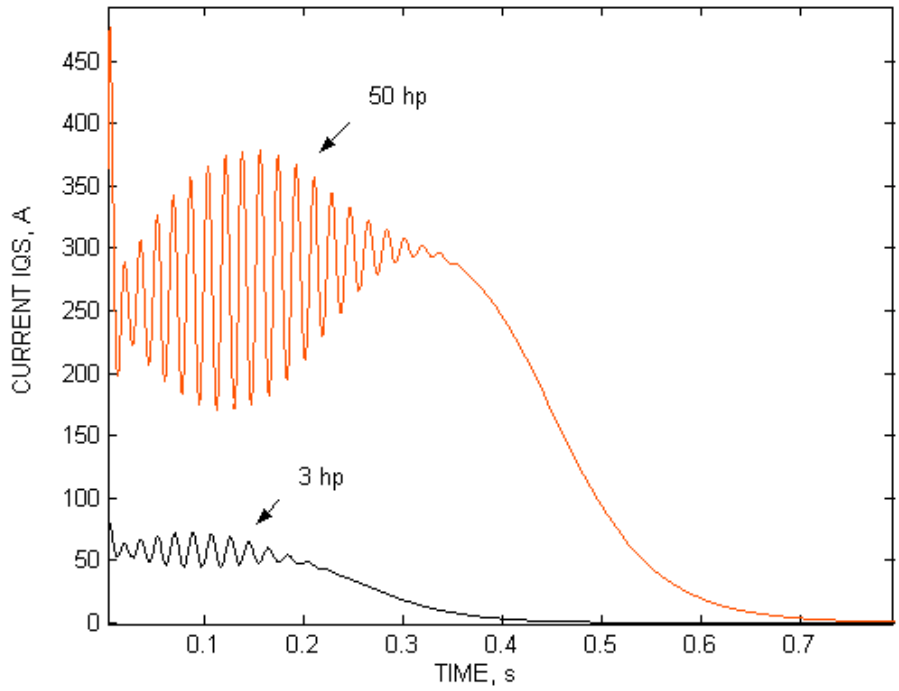


Figure 3.6. Three - phase stator currents during starting of different machines.

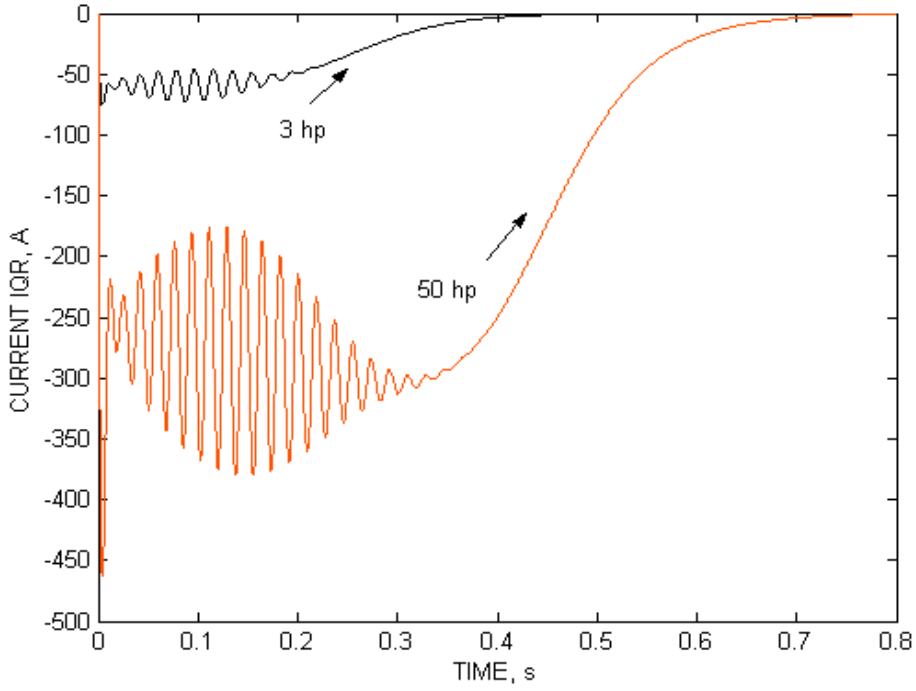


Figure 3.7. Three - phase rotor currents during starting of different machines.

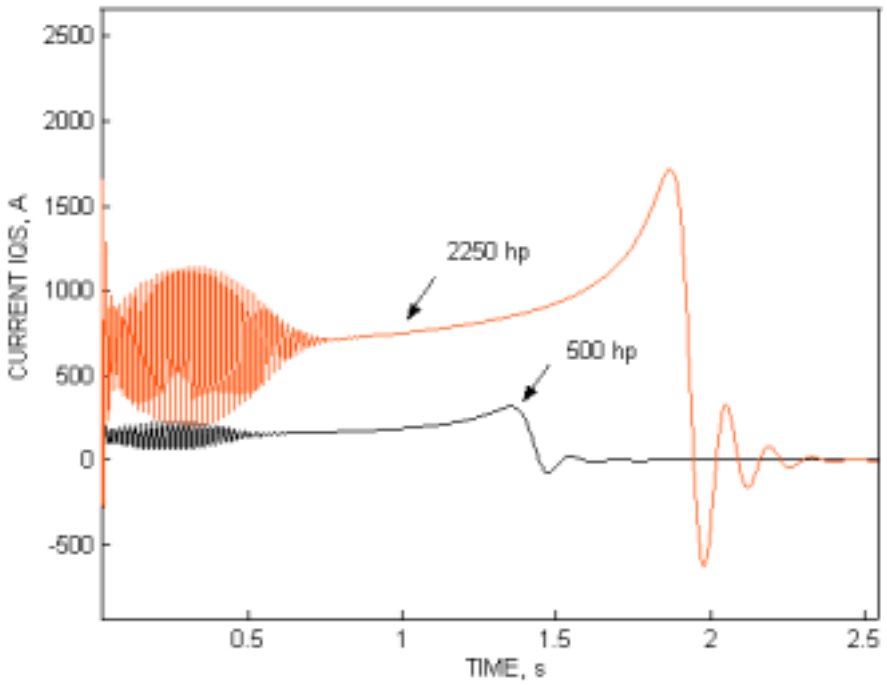


Figure 3.8. Three - phase stator currents during starting of different machines.

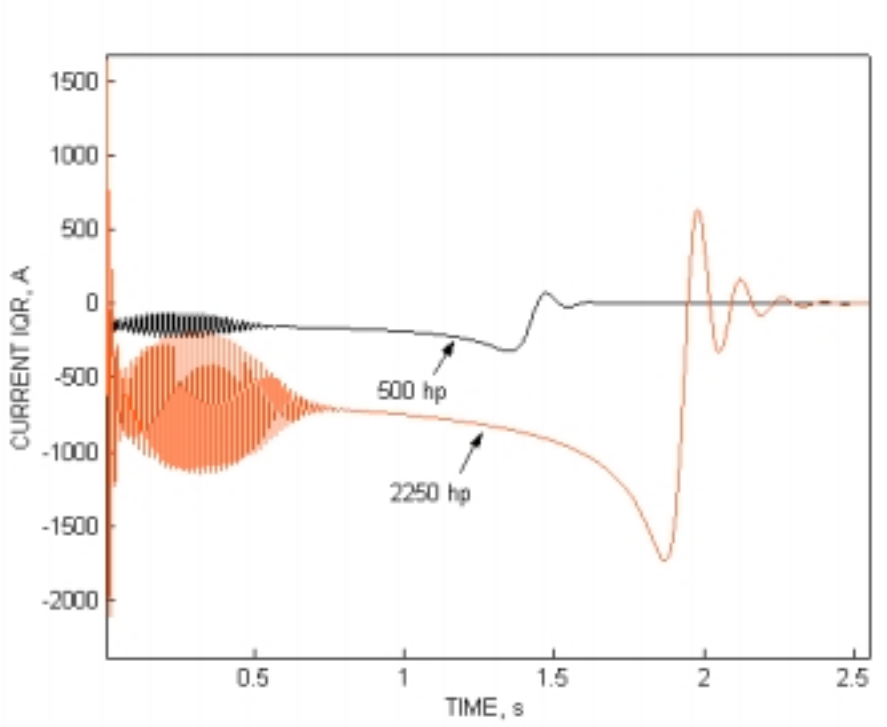


Figure 3.9. Three – phase rotor currents during starting of different machines.

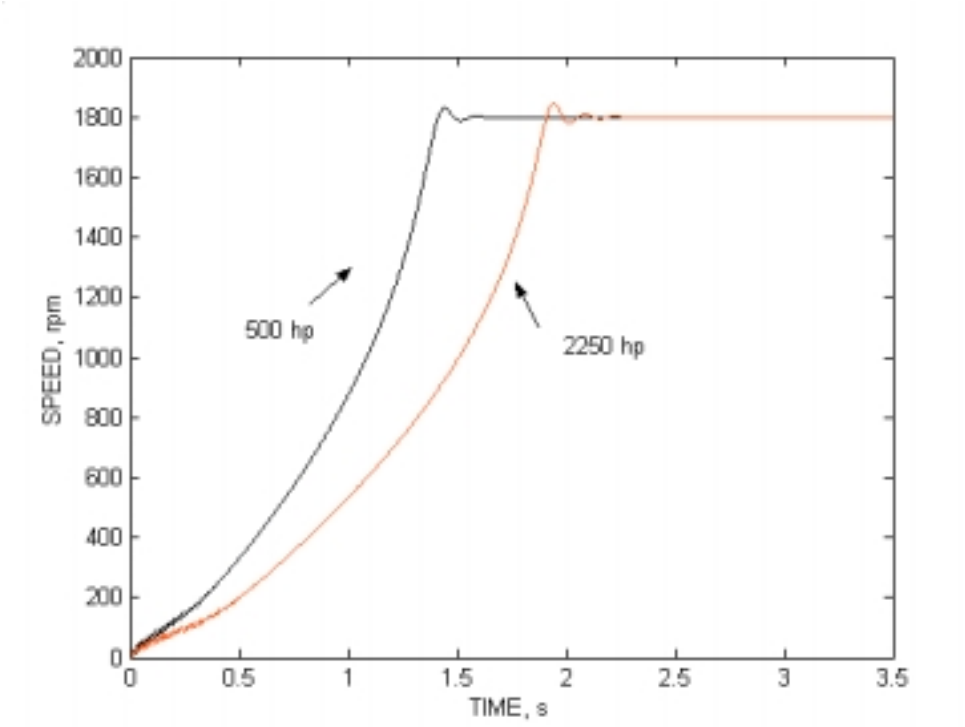


Figure 3.10. Rotor speed during starting of different induction motors.

3.3. DC, No Load and Blocked Rotor Tests

Reference [36] describes traditional methods of determining the parameters of an induction motor. These include a no – load test, a blocked rotor test and DC measurement. To calculate the parameters from the measurements, the steady state model in [5] is used.

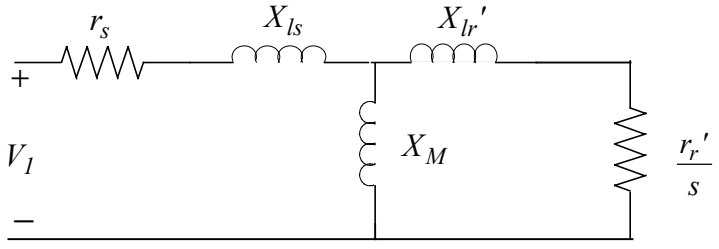


Figure 3.11. Equivalent circuit of an induction motor.

The no – load test on an induction motor gives information with respect to exciting current and no – load losses. The test is analogous to the transformer open circuit test and is performed on induction motor in which the shaft is unloaded and allowed to spin close to the synchronous speed. The test is taken at rated frequency (60 Hz) and with balanced three phase voltages applied to the stator terminals. The measurements in these tests are motor power P_{nl}, line-to-line voltage V_{nl}, and current I_{nl}. At no – load, the operating speed is very close to synchronous speed; the slip is $s \approx 0$, causing the current in the R_2 / s branch to be very small. The no – load rotor I^2R loss therefore is negligibly small. By neglecting rotor I^2R losses, the rotational loss P_R for normal running condition is

$$P_R = P_{nl} - q_1 I^2_{nl} R_1 \tag{3.32}$$

where

P_{nl} = Total three phase power input

I_{nl} = Current per phase

q_1 = Number of stator phases

R_1 = Stator resistance per phase.

The no load impedance is calculated using the measured voltage and current:

$$Z_{nl} = \frac{V_{nl}}{\sqrt{3} I_{nl}} \tag{3.33}$$

where V_{nl} is the line – to – line terminal voltage in the no – load test. The no load resistance can be computed from the real power:

$$R_{nl} = \frac{P_{nl}}{3 I_{nl}^2} \tag{3.34}$$

The no load reactance X_{nl} then is given as:

$$X_{nl} = \sqrt{Z^2{}_{nl} - R^2{}_{nl}} \tag{3.35}$$

The next step in standard identification of parameters of an induction motor is a blocked rotor test. Like the short circuit test on transformer this test gives information with respect to leakage impedances. The rotor is blocked so that it cannot rotate, and balanced three phase voltages are applied to the stator terminals. As in the no load case, power P_{bl}, voltage V_{bl} and current I_{bl} are measured. The IEEE test procedures recommend this test to be performed at 25 percent of rated frequency. The reason for that is to obtain a representative value of R_r' since, during normal operation, the frequency of the rotor currents is low and the rotor resistances of some induction machines vary considerably with frequency. In the test performed in this thesis, the blocked rotor test was carried out at 60 Hz (as in the case of high slip). Under the assumption that exciting current is neglected, the blocked – rotor reactance X_{bl} equals the sum of the normal frequency, stator and rotor leakage reactances X_1 and X_2 which is described:

$$X_\sigma = \frac{1}{2} X_{bl} \tag{3.36}$$

where X_σ represents X_1 and X_2 respectively. The magnetizing reactance is determined from the no – load test and the value of X_M. Thus,

$$X_M = X_{nl} - X_1 \tag{3.37}$$

The *DC* measurement is performed immediately after the blocked rotor test in order to reflect any heating of the stator that may have occurred during the test. This test directly provides an estimate of the stator resistance. During the *DC* test a *DC* voltage is applied across two terminals while the machine is at standstill. Stator resistance is calculated as:

$$R_{dc} = \frac{V_{dc}}{I_{dc}} \tag{3.38}$$

Since the stator of the motor used in this work is wye – connected, the per phase stator resistance is calculated as:

$$R_1 = \frac{R_{dc}}{2} \tag{3.39}$$

If the stator is delta – connected, the per phase stator resistance can be calculated as:

$$R_1 = \frac{3}{2} R_{dc} \tag{3.40}$$

Then the rotor resistance is computed from the given equations:

$$R = R_{bl} - R_s \tag{3.41}$$

$$R_r = R(\frac{X_{nl}}{X_M}) \tag{3.42}$$

DC – resistance, No – load and *blocked rotor tests* are performed on a 7.5 hp induction motor. Chapter 5 provides more details on this.

Chapter 4

Lab VIEW Software and Data Acquisition System

In this chapter an automatic procedure to acquire the input and output variables of an induction motor is described. The procedure starts by explaining the hardware instruments that are used for the data acquisition system (DAQ). The process, configured as a *virtual instrument* (VI), is implemented in the Lab VIEW environment to measure and save the signals of the three - phase voltages, three - phase currents and rotor speed of the induction motor. The features of the VI and Lab VIEW monitoring system provide enormous help in parameter identification of an induction machine since it gives information on acquired signals needed for the identification problem to be solved.

4.1. Data Acquisition System (DAQ)

The data acquisition (DAQ) VI library acquires waveforms and generates signals with all National Instruments (NI) plug-in and remote data acquisition products. The data acquisition system utilized here has a NI DAQ Card (6040E Families). (See reference for more details [35]). The card is installed on the PCI bus of the computer used for this research. The Input/Output Connector of the DAQ card has 68 pins for the analog and digital signals. Three-phase currents and voltages in order to be measured must give six analog inputs in the range of $\pm$10 volts (Note: if the input signal exceeds +10/-10 volts, the card will be damaged). Therefore, to measure high current or voltages or speed

special transducers must be employed to convert high amplitude signals within the required range. Two different types of transducers are used for currents, CTL – 51 and CTA201R, and for voltage VT7. To measure the speed, analog output is already provided by a digital tachometer (speed transducer), which converts speed in the range of 50 – 10000 rpm to 0 to 5 volts. The way in which analog inputs are connected to the DAQ card is given in Appendix A. A block diagram of the data acquisition blocks along with induction motor and transducers configured for this experiment is shown in the Figure 4.1.

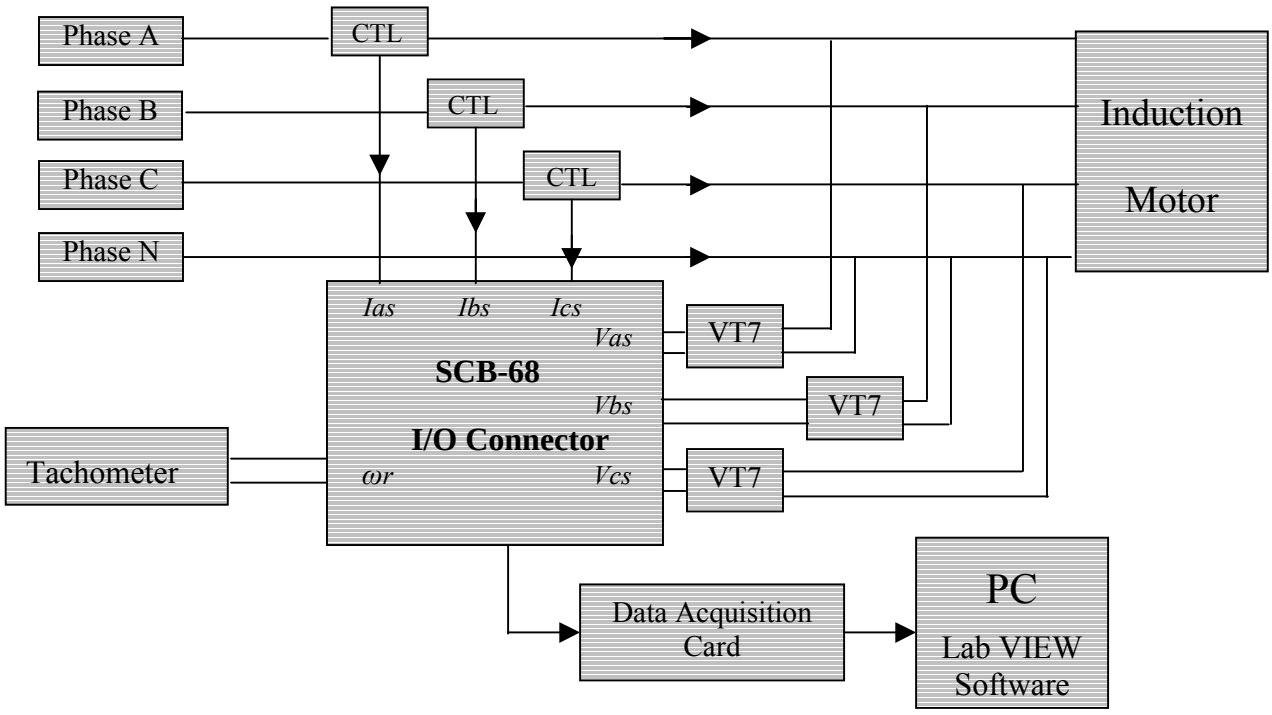


Figure 4.1. Data acquisition block diagram for a three-phase system.

DAQ systems are used in a wide range of applications in the laboratories, in the field, and in manufacturing plants. A general DAQ system includes:

- *Transducers* - attached to or submerged in the physical property (induction motor in this case) being measured. Sensors produce an output signal analog voltage proportional to a change in the signal (for example, for the case of three phase

voltages of induction motor – transducers convert input voltage in the range of –600 - +600 V to analog output voltage in the range of –10 to + 10 Volts. See Appendix A on more information about connections).

- *Signal Conditioning* – for conditioning signals in the range of milivolt (mV) to upgrade them to a level that the plug – in hardware can read.

- *Plug – in Hardware Devices* – plug in boards, interfaces, and external digital signals which are stored by the computer for users to analyze with software.

- *Driver Software* – takes information from the hardware device and transfers it into the computer's memory.

- *Computing Platform* – desktop or laptop PC that receives and uses the data.

4.1. Lab VIEW Software

Lab VIEW is a visual programming language for collection, analysis and display of measured data. The software was originally produced by NI (38). It incorporates *a graphical user interface* (GUI) programming environment to produce programs that mimic laboratory instruments. These programs are called *Virtual Instruments* (VI) since they actually imitate real work surface instruments.

Lab VIEW utilizes G programming language, a graphical language used for description of connections and relations between objects in a block diagram. G language has many differences compared to textual language. This program is executed linearly, meaning one by one command. The next command is executed after the previous has been finished. Before starting the process of the program, all the inputs represented by graphical blocks must be well defined and determined. It is also good to note that within this language it is possible to create a diagram with a few parallel branches, thus realizing simultaneous execution of numerous operations at the same time. The main idea of programming in G language is to create VI and include them in other ones, thus making a

sub – virtual instrument. For each instrument it is necessary to define input and output variables and to draw the icon. VIs resemble the functions in textual programming language, where the functions are used instead of icons. Each VI in Lab VIEW consists of:

- *Front Panel User Interface*
- *Block Diagram*

These user interfaces provide interactive control of the software system.

4.1.1. Front Panel User Interface

Front panel is the graphical user interface for creating commands for displaying instruments and measurements results. This interface collects user input and displays program output. The *front panel* usually consists of different *control* and *indicator* objects. *Control objects* are used for data input, while *indicator objects* are used for data output.

Front panel is very convenient to create and to operate with. It provides a control of the system at runtime by simply operating the various objects on the window, whether it be by changing a position, size and name of the object or defining different data types. One example of *Front Panel* and VI designed in the power laboratory for the purpose of acquiring stator currents, stator voltages and speed of the rotor shaft of induction motor is shown in the Figure 4.2. Figure 4.3 shows front panel configurations for reading and displaying saved data from the file. After building the *front panel* window it is necessary to explain the behavior of VIs. The employed code for this operation is defined as a block diagram.

4.1.2. Block Diagram

The *block diagram* contains the graphical source code of VIs and is constructed in Lab VIEW's graphical programming language G. All the functions displayed in the *front panel* must be controlled and performed in the *block diagram*, which is the actual executable program of VI application. This program, called graphical visual language, specifies data types and presumes values and connections between objects. The components of a block diagram, icons, represent lower – level VIs, named as *built – in functions* and *program control structures*. The *block diagrams* designed for obtaining signals from the induction motor are shown in the Figures (4.4, 4.5, 4.6 and 4.7). In order to append and read that data during the free acceleration of the motor, two different block diagrams are created, each performing its own operations. The small rectangular objects or nodes in the figures represent symbols from the front panel. These nodes are connected by *wires*, which define the flow of data through the program. The execution of a node occurs when all its inputs are available. Inputs are pre-specified by the user by entering the number of devices used, channels for measuring signals, time scan rate, etc. in the front panel window. When a node finishes executing, it releases all its outputs to the next node (object) in the dataflow path.

4.2.3. Lab VIEW Features

Lab VIEW uses a patented dataflow-programming model, which is different from the linear architecture of text-based languages. Because it is the flow of data between objects on a block diagram and not sequential lines of text which determines execution order in Lab VIEW, it allows easy build – up of diagrams that execute multiple operations simultaneously. Consequently, Lab VIEW is a multitasking system capable of running multiple execution threads and multiple VIs concurrently [38].

Lab VIEW VIs are modular in design, so any VI can run on its own or be used as part of another VI (subVI). With this modularity, it gives the user flexibility to design a

whole hierarchy of VIs and subVIs that serve as building blocks in any number of applications. Also, it provides easy modification, interchange, and combination of VIs in order to accommodate other applications.

In many applications, execution speed is a critical consideration. Lab VIEW is the only graphical programming system with a compiler that generates optimized code with execution speeds comparable to compiled C programs. To further improve performance, the user can analyze and optimize time-critical sections of code with the built-in Profiler. In this way, productivity with graphical programming will be increased without sacrificing execution speed.

4.2.4. Measure and Save Data to a File

Measure and Save Data to a File VI demonstrates a simple way to write the acquired signals of either voltages, currents or speed (i.e. scaled) to the text file readable by the spreadsheet programs. During the *acquiring and writing to a file* process, each row of data is scanned and each column is represented as a different channel. After the program creates the file, it initializes and starts the acquisition. It converts the scan rate to a string ending with an *end of line* character and writes it as the *first line* in the file.

4.2.4.1. Front Panel Design

The front panel includes a *control of digital data* and *waveform chart*. Figure 4.2 shows a *front panel* example created for the purpose of acquiring different signals. On the left hand side of the graph there are hardware setup requirements, which are initialized before starting any application. This example uses the circular buffer technique of data acquisition whereby data are acquired into a circular acquisition buffer. The scan backlog indicates how much data remains in the buffer after each retrieval, and is an indication of how well the application is keeping up with the acquisition rate. If the backlog increases with the time, this indicates that the scanning is too fast and will eventually overwrite the circular buffer. In each iteration VI reads scanned data from the

acquisition buffer, converts the data to a spreadsheet string format, and writes it to the file. During *each scan interval* acquired data can be viewed on a *waveform chart*, Figure 4.2. After activating the stop button, the program automatically stops the acquisition and closes the file. It is relatively easy to eliminate plotting to decrease overhead, but there should be concern about this since writing to a spreadsheet file is inherently slower and consumes more local and file memory than writing to a binary file. (Note: See the description field on the controls and indicators for operating instructions). One good feature of Lab VIEW is the opportunity to visually control and observe the flow of data by employing different indicator functions. One example in this case is the tachometer shown below the waveform chart in the Figure 4.2, which gives rotational speed values of the induction motor. Also, the acquired signal values of three - phase currents, three - phase voltages and speed can be viewed simultaneously on the digital indicators located at the right hand side of the front panel.

4.2.4.2. Wiring Diagram Design

In the wiring diagram designed each of the previously shown control blocks and indicators must have functions well defined to perform the required operations. Part I of the diagram configures channels and registers devices. The following icons are connected to the AI Config before start of the application:

- *File dialog box* – for opening an existing file, creating new file, or replacing file.
- *Device icon* – for assigning device numbers to the DAQ device before configuring files (before calling AI Config intermediate function).
- *Input limits* – for defining the expected signal limits for the channels. These limitations are defined by input limits array of clusters.

- *Channels* – for indicating channels for each of the analog signals used. (e.g. order of the channels for three phase voltages, three phase currents and rotor speed can be found in Appendix A).

- *Buffer size* – for specifying the buffer size, which represents the number of scans, each buffer holds. By default it is automatically assigned to 1000 scans.

- *Interchannel delay* – for specifying the waiting time between sampling channels within a scan. Lab VIEW selects a default interchannel delay automatically of value -1, giving the hardware time to settle between channels.

After specifying the inputs, VI can be configured by activating the AI Config intermediate VI function. This VI configures the hardware and allocates a buffer for an analog input operation. Then, VI Al Start – icon starts a buffered analog input operation. This VI sets the scan rate and the number of scans to acquire. After activating this function VI starts an acquisition of data.

In Part II of the wiring diagram a *For Loop* is defined for continuously reading and saving data to the file. Inside of the *For Loop* the acquired data are sent to the waveform chart and saved. The tachometer indicator reads the status of scanned signal of speed. The following icons are connected for these operations:

- *For Loop* – for executing its sub diagram until a Boolean value wired to the conditional terminal is False.

- *AI Read* – for reading data from a buffered data acquisition.

- *Array to Spreadsheet String* - for converting an array of any dimension to spreadsheet string.

- *Close File* – for writing contains of the buffer to disc, updating the directory entry of the file and closing the file.

- *AI Clear* – for stopping an acquisition associated with analog input task and for releasing associated internal resources, including buffers.

- *General Error Handler* – if an error occurs, the VI creates a description of the error and optionally displays a dialog box, which can be viewed on the front panel window. Inside the For Loop there is a NOR gate, which has ability to stop the scanning procedure. Input command for the NOR gate is from the STOP button. The file is closed when the loop iterations are stopped.

In Part III of the wiring diagram the *Sequence Structure* is formed, which has the ability to control the order of execution of nodes, which are not data dependent. In this case offset time for *X(0)* and *ΔX(0)* time are sent to the waveform chart in order to adjust the time axis scale using the buffers values in Part II and I.

4.2.5. Read Data from a File

The objective of this VI is to read the data already written to the TXT. files and to display the data or preferred parts on the waveform graph. The acquired data are read in the same format and with the same scan rate in which they are saved. Hence, since originally data are saved in ASCII format using string data, they must be read as a string data with one of the I/O file VIs.

4.2.5.1. Front Panel

Figure 4.2 shows the *front panel* VI created for the displaying already saved data. Before starting this application *control indicators,* which are located below the waveform chart must be initialized. Scan rate is the fixed one, which is read from the file. The number of channels should also be indicated. Start time and time interval can be defined with respect to the interval in which user wants to view the waveforms. This implies that in this VI there is an option to zoom in and out different parts of the graphs. After specifying start time and time interval, the START button is activated and waveforms

defined in certain time interval are displayed. Each time the START button is activated again the waveform of the next time interval can be viewed on the graph.

4.2.5.2. Wiring Diagram

This executable code starts with reading the first row of the file, which is scan rate (shown on digital indicator – front panel). By specifying the start time and scan rate, the *For Loop* can begin the process of reading the data. The following executable function is employed to perform this operation:

- *Read from Spreadsheet File* - reads specified number lines or rows from a numeric text file beginning at the specified character offset and converts the data to 2D, single – precision array of numbers. The VI opens the file before reading from it and closes it afterwards.

Inside the First Loop the buffer data are sent to the transposed waveform chart in the *front panel* for displaying the waveforms. The number of channels in each iteration is calculated from the file and sent to its digital indicator. There are two options here:

- *To end the file*

- *To stop the reading.*

Another *For Loop* is defined inside of the first *For loop*, mainly employed to assist in the finalization of the process. In this *For Loop*, two NOR gates are utilized to finish the reading procedure. They get input commands from the *end of file* output signal, which comes from *read file* icon and *stop button* icon controlled by the user. By default, the NOR output signals are defined as *true,* which implies that *for loops* should continue reading data; otherwise one of the stop conditions happens.

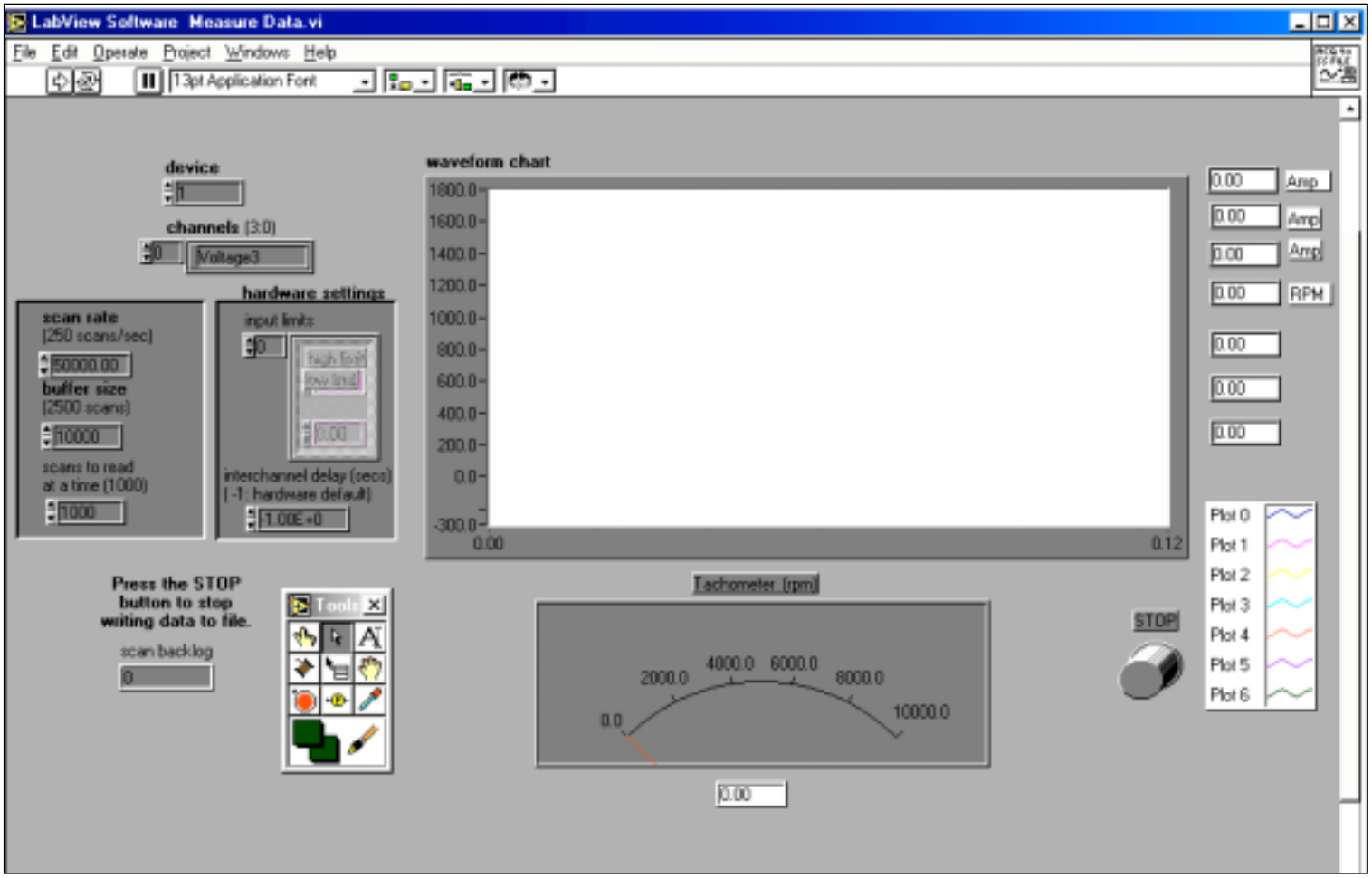


Figure 4.2. Front Panel for measuring and saving input signals of induction motor.

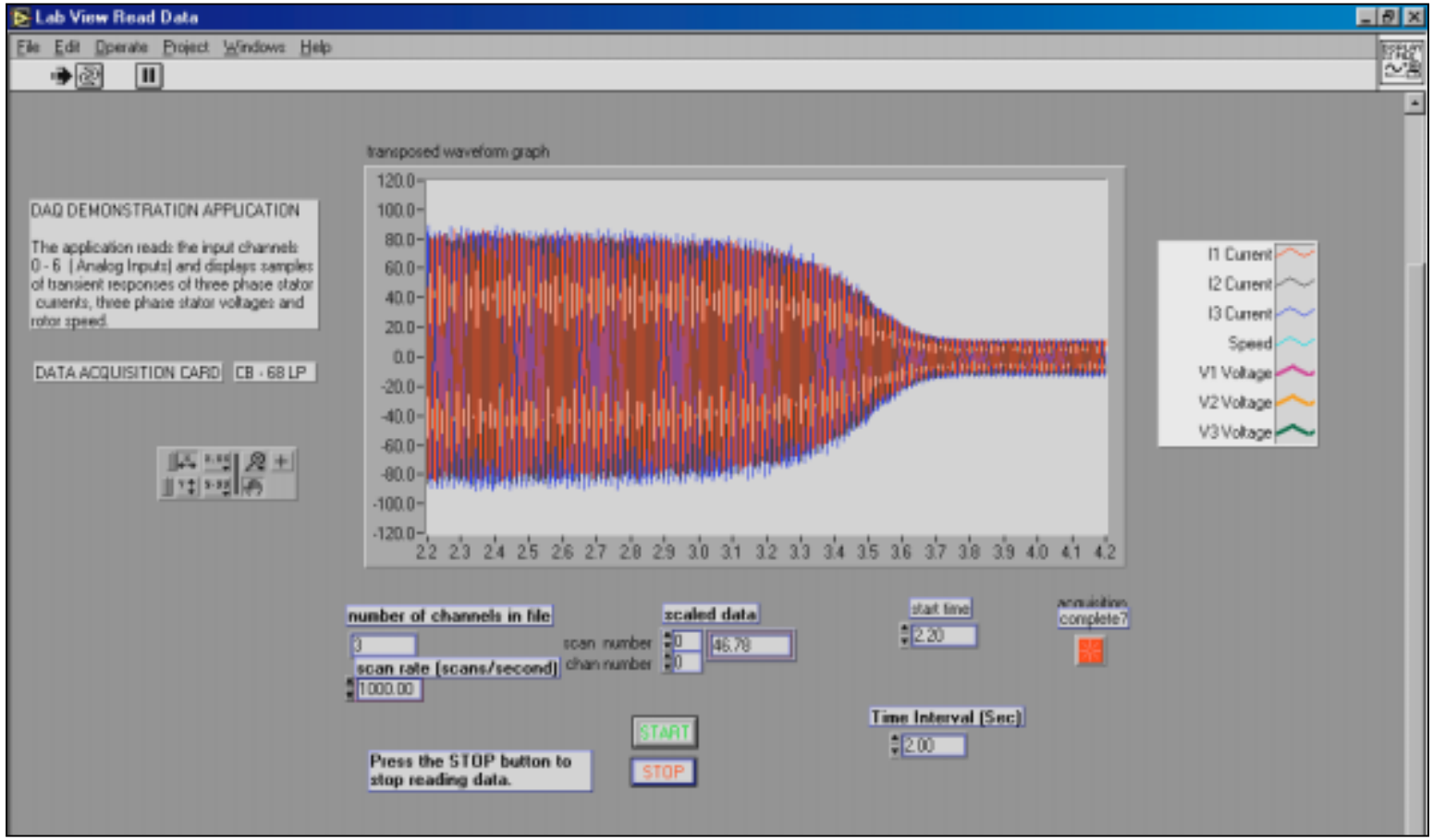


Figure 4.3. Front Panel for reading and displaying already saved signals of induction motor. Transient stator currents.

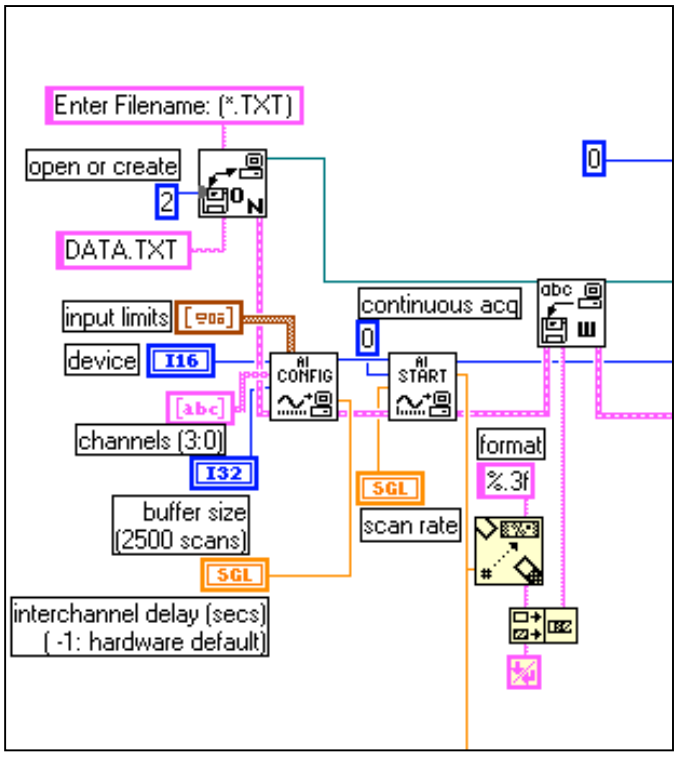


Figure 4.4. Block diagram for measuring and saving data to a file. (I part)

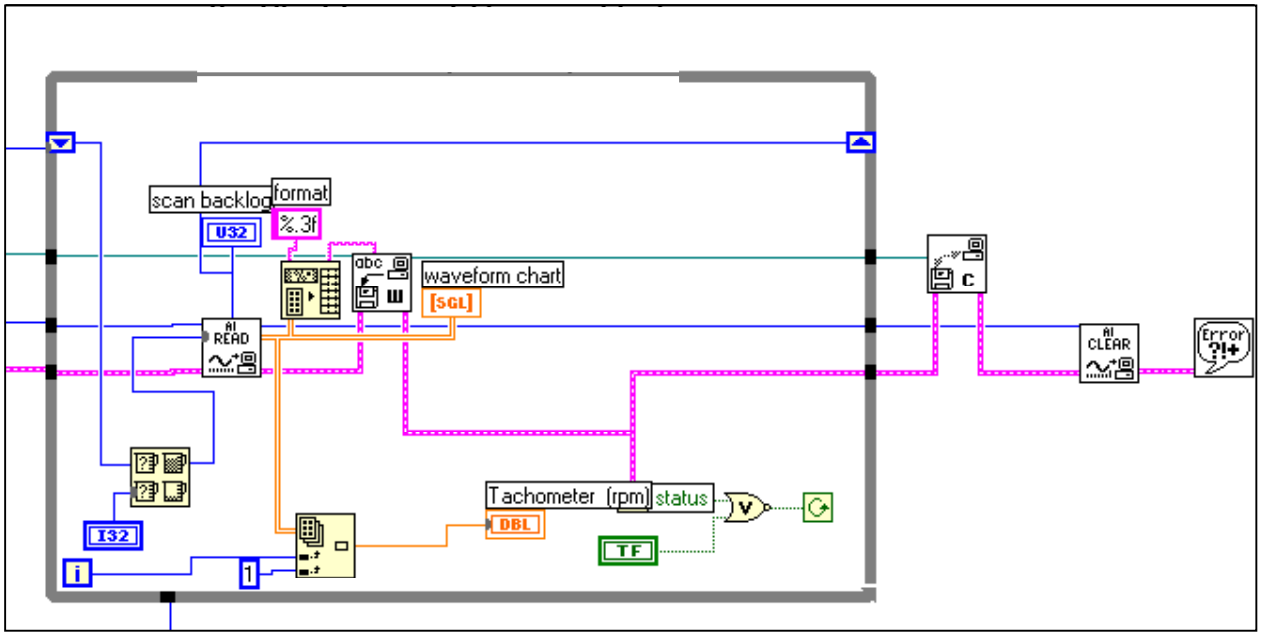


Figure 4.5. Block diagram for measuring and saving data to a file. (II part).

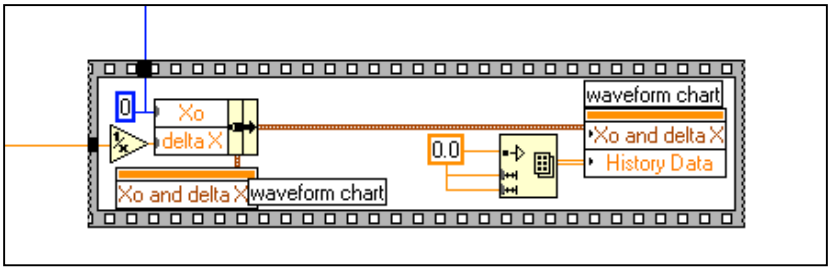


Figure 4.6. Block diagram for measuring and saving data to a file. (III part).

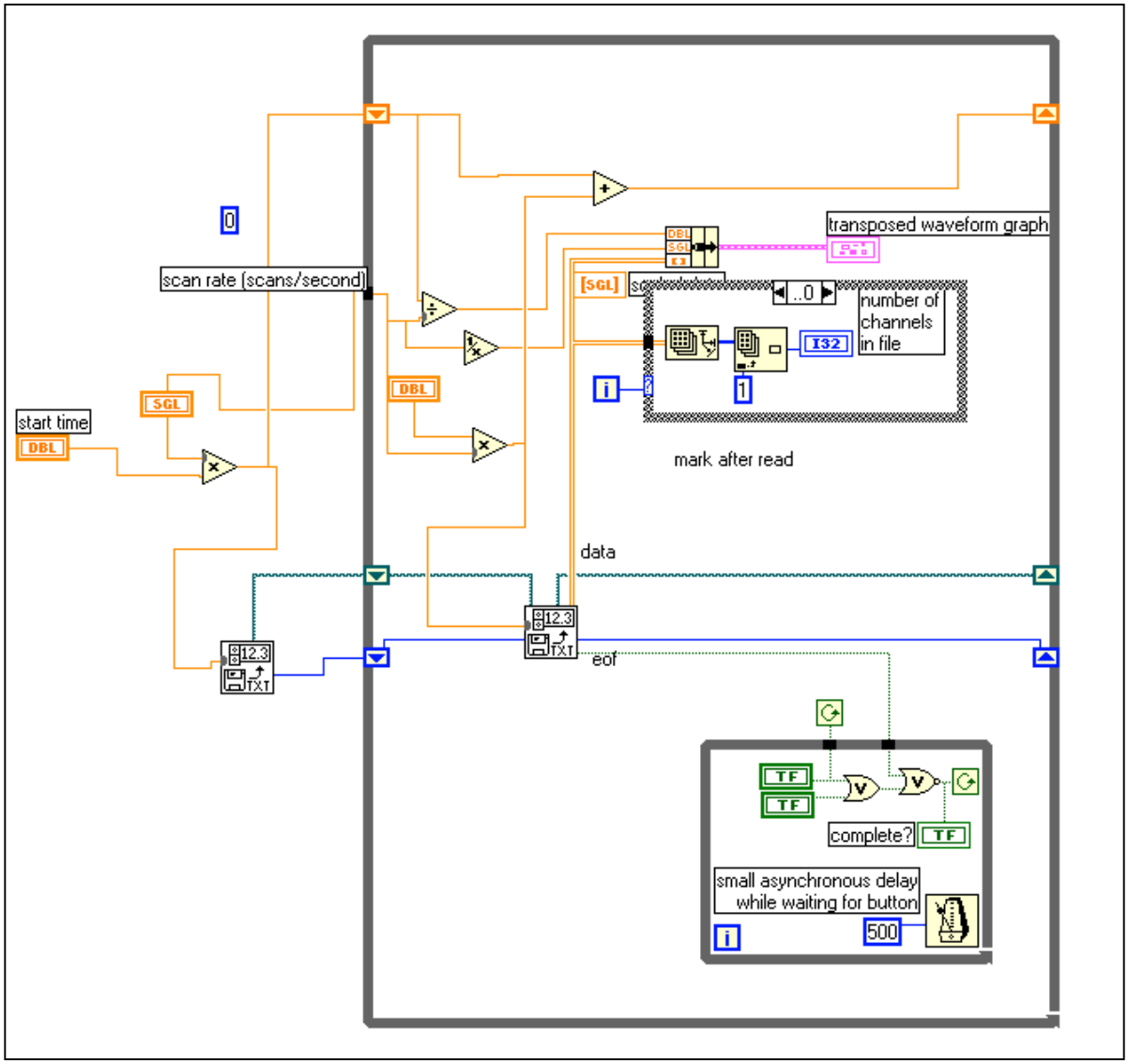


Figure 4.7. Block diagram for reading and displaying signals from the files.

Chapter 5

Case Studies and Results

This chapter discusses the experimental testing and implementation of GAs to the problem identification of electric parameters of induction motor. The standard test procedure of *DC resistance*, *no load* and *blocked rotor* is applied to the 7.5 hp induction motor to obtain parameters. Then, a free acceleration test is performed, and with the help of DAQ and Lab VIEW Software three - phase voltages, three - phase currents and speed are recorded. Next, the GA method is used to estimate parameters of induction motors. Thus, the method is based on the following:

- Experimental Test Procedure.
- Selection of the Mathematical Model.
- Implementation of GA to the problem identification.

5.1. Experimental Test Procedure

The first test applied for determination of parameters is based on IEEE standard procedure. The following table gives information on parameters obtained after performing *DC resistance*, *no – load* and *blocked rotor tests* of a three – phase, 7.5 hp, 60 Hz, 220 V, induction motor.

Table 5.1. No load, blocked rotor and DC resistance test results.

7.5 hp Induction Motor	No Load Test	Blocked Rotor Test	DC Resistance Test
P_{input} = 6582.35 W	V_{nl} = 223.3 V	V_{bl} = 33.8 V	V_{dc} = 11.74 V
S = 7743.94 VA	I_{nl} = 9.095 A	I_{bl} = 19.429 A	I_{dc} = 20 A
$I_{full\ load}$ = 20.34 A	P_{nl} = 0.763 KW	P_{bl} = 0.601 KW	

Both *no load* and *blocked rotor* test are performed at 60 Hz frequency. According to the IEEE Standard Test Procedure given in [36], this case applies for the situation when the slip is near unity thus, right after the machine start – up, when the starting currents have the largest values. If, however, one is interested in the normal running characteristics, the blocked rotor test should be taken at about rated current and the frequency must be reduced, since the value of the rotor effective resistance and leakage inductances at the low rotor frequencies corresponding to small slips may differ appreciably from their values at normal frequency. No attempt was made in this work to compose the test at reduced frequency since there was no adjustable frequency drive in equipment setup.

Table 5.2. Symbols and units of basic variables and parameters.

Symbol	Variable or Parameter	Units
R_s	Stator Resistance	Ω
X_{ls}	Leakage Stator Impedance	Ω
X_m	Magnetizing Impedance	Ω
X_{lr}'	Leakage Rotor Impedance	Ω
R_r'	Rotor Resistance	Ω
ω	Base Electrical Frequency	*rad/s*
ω_e	Synchronous Speed	*rad/s*
ω_r	Rotor Speed	*rad/s*
p	Time Derivative, *d/dt*	*1/s*

From the given equations in Chapter 3 it follows that:

$$R_s = \frac{R_{dc}}{2} = 0.2935 \ (\Omega) \tag{5.1}$$

from the *no – load* test:

$$R_{nl} = 3.075 \ (\Omega) \qquad \text{and} \qquad X_{nl} = 13.837 \ (\Omega) \tag{5.2}$$

and *blocked rotor* test:

$$R_r' = 0.5307 \ (\Omega) \qquad \text{and} \qquad X_{bl} = 0.8523 \ (\Omega) \tag{5.3}$$

According to IEEE empirical distribution for different motor classes and based on name – plate data of the motor it follows that:

$$X_{ls} = 0.5 \cdot (X_{bl}) = 0.426 \ (\Omega) \tag{5.4}$$

$$X_{lr}' = 0.5 \cdot (X_{bl}) = 0.426 \ (\Omega) \tag{5.5}$$

$$X_m = X_{nl} - X_{ls} = 13.411 \ (\Omega) \tag{5.6}$$

The parameters are summarized in the following table:

Table 5.3. Standard test parameters for 7.5 hp induction motor.

R_s	X_{ls}	X_m	$X_{lr'}$	$R_{r'}$
0.2935	0.426	13.411	0.426	0.2372

The next step is then applied to measure three - phase voltages, three - phase currents and rotor speed. Machine variables are recorded during the start- up conditions. Three phase voltages are generated from the Motor - Generator Group (Note: See Appendix A for more information). A three-phase wound rotor induction motor rated at 7.5 hp is tested. The configuration test for the experiment can be found in Appendix A,

where Data Acquisition System and Lab VIEW Software are employed for recording the waveforms. Voltage and current signals in the range of [-10 V, +10 V] are acquired by the means of voltage and current transducers. A digital transducer – tachometer is used to generate analog voltage input at the range of 0 to 5 volts. The number of scans reading per second is selected as 1000 and free acceleration period is recorded for the duration of approximately 3.5 (s). No attempts were made to reduce measurement noise. This is due to the fact that transducers are calibrated to give very small error in measurements. Another reason is insensitivity of the GAs to the noise signal. After employing Lab VIEW Software, acquired signals are saved in the form of spreadsheet files for different operating conditions. Data collected are transformed to the files suitable for the identification method by the GAs. Source code of the GAs is listed in Appendix C.

The boiler – plate data from the test of induction motor is given in Table A.1. in Appendix A. No mechanical load, except for the rotor inertia and windage, was attached to the motor. Figure 5.1 shows the stator voltages transients during the starting of the induction motor. It is interesting to note that during this period since both stator and rotor are deeply saturated by large inrush currents, there was a 50 % voltage drop at the terminal of the machine. This voltage change can affect the stability of the system while starting current can cause large fluctuation of the rotor torque which will be shown in results for different machines.

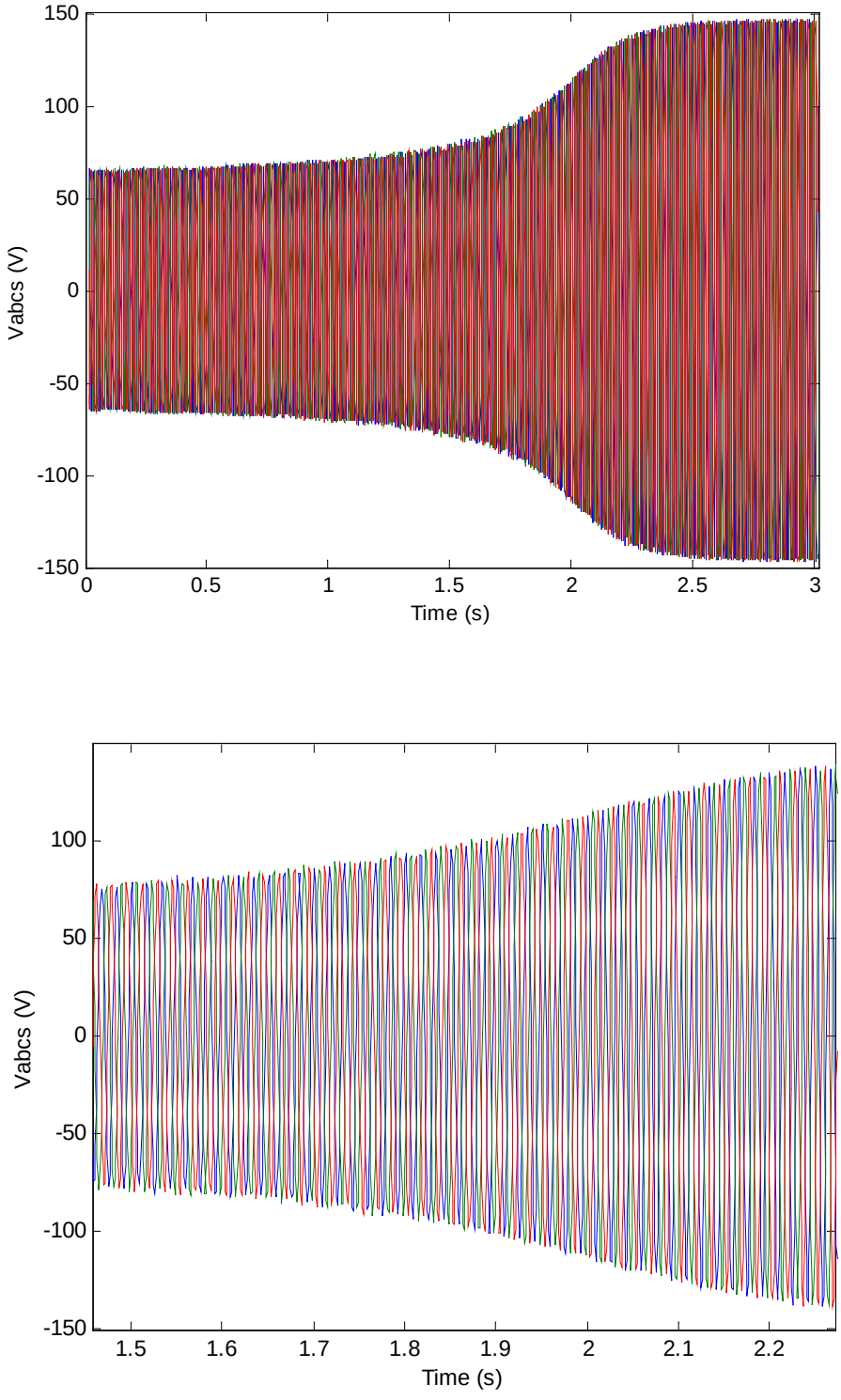


Figure 5.1. Three – phase voltages during starting of induction motor.

5.2. Selection of the Mathematical Model

The induction motor model described in Chapter 3 is selected for the study. This model represents the nonlinear dynamic behavior of the induction machine, where all the electric parameters are assumed to be varying depending on the operating conditions. The objective of the identification is to determine a vector of parameters, which represent the resistances and inductances of the motor,

$$\theta = [R_s \;\; X_{ls} \;\; X_m \;\; X_{lr}' \;\; R_r']$$ (5.7)

Problem definition is formulated as to find the parameters with a given set of measurements:

- Three phase stator voltages and currents.
- Rotor Speed.

Induction machine voltages are transformed to the synchronously rotating reference frame:

$$V_{qdo} = K \cdot V_{abc}$$ (5.8)

$$\text{where} \quad V_{qdos} = V_o \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \cdot u(t) \quad \text{and} \quad V_{qdor} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$ (5.9)

For the identification algorithm employed, stator and rotor fluxes are defined as state variables:

$$X = [\Psi_{qs} \;\; \Psi_{ds} \;\; \Psi_{os} \;\; \Psi_{qr} \;\; \Psi_{dr} \;\; \Psi_{or}]$$ (5.10)

Rotor speed is included as seventh state variable in the state space equation form. By employing one of the ode solvers in Matlab (ode45, ode15s) and solving the equation (3.10) in Chapter 3, solution for the fluxes and speed can easily be obtained. State space equation (3.10) is a function of:

$$pX = f(\theta,\ X,\ V) \tag{5.11}$$

where X is considered as a state variable vector, V as the input vector from the supply and θ the parameter vector to be identified. Note that, state equation solved for fluxes (3.10) is a nonlinear differential equation since rotor speed as a state variable appears inside of matrices.

5.3. Implementation of the GA to the problem identification

In this study, the GAs are utilized to identify the electric parameters of the induction machine. The application of the GAs to parameter identification [20] is defined as:

- Initial population of parameters is generated as $\theta^{(o)} = \left\{ \theta_i^{(o)} \middle|\ i = 1,2,\cdots,l \right\}$ with randomly selected individuals. Each individual is constrained by the following condition:

$$\theta_{\min j} \le \theta_{ij}^{(g)} \le \theta_{\max j} \qquad \text{where} \qquad j = (1,2,\cdots,m) \tag{5.12}$$

  where $\theta_{\min j}$ and $\theta_{\max j}$ are the limits of the *j*th element of the parameter vector, $\theta_i^{(g)}$ given by a priori knowledge. $\theta_{ij}^{(g)}$ denotes the estimation of the *jth* element of the *i*th individual in the *g*th generation of the parameter. The process starts with *g* = 0.

- Each individual $\theta_i^{(g)}$ is used to calculate the state variable $X(t)$ and the general performance $Y(t)$ corresponding to the measured data defined by equation (5.11):

$$pX(t) = f(\theta, X(t), V(t)) \tag{5.13}$$

$$Y(t) = C \cdot X(t) \tag{5.14}$$

and the performance index or cost function

$$E(\theta_i^{(g)}, t) = Y(t) - \overline{Y}(t) \tag{5.15}$$

where $Y(t)$ corresponds to the measured or actual performance vector, and $\overline{Y}(t)$ is the simulated one. Then the fitness is computed by the means of minimization error between measured and simulated data. The quality of fitness is defined as:

$$Fitness\ (\theta_i^{(g)}, t) = \frac{K}{E(\theta_i^{(g)}, t)} \tag{5.16}$$

where the K is defined as the gain value.

- A new generation of $\theta^{(g+1)}$, with the same individual number of $\theta^{(g)}$, is formed by means of reproduction, crossover, and mutation based on the previous $\theta^{(g)}$.

- The criteria for the stop of the process is defined as:

If generation > 5:

$$\frac{\theta\ (t) - \theta\ (t-3)}{\theta\ (t)} < \varepsilon \quad \& \quad \theta(t) > \theta(t-3)\,.$$

If generation >1:

$$\frac{\theta\ (t) - \theta\ (t-1)}{\theta\ (t)} < \varepsilon \quad \& \quad \theta(t) > \theta(t-1), \quad \text{or}$$

$$g = MaxGen, \tag{5.17}$$

where ε is defined as a very small number (10e-3 in this optimization process) and *MaxGen* represents the maximum generation number. In order to compare different

solutions of identification problem, the following performance indices are considered for this study:

$$E(\theta_i^{(g)},t)=\sum_j\left|Y_j(t)-\overline{Y_j}(t)\right| \tag{5.18}$$

$$E(\theta_i^{(g)},t)=\sum_j\left(Y_j(t)-\overline{Y_j}(t)\right)^2 \tag{5.19}$$

$$E(\theta_i^{(g)},t)=\sqrt{\sum_j\left(Y_j(t)-\overline{Y_j}(t)\right)^2} \tag{5.20}$$

In this work there are no explicit constraints, but an allowable range of values is assumed for each parameter. For the purpose of optimization, programs developed in [30] were used. Genetic operators for parameter optimization are given in Table 5.4. The description of these operators and their properties can be found in [2]. The maximum number of generation was set to 300 for the first data analysis and then to 400 in second case. The best individuals in terms of maximum fitness function are considered to be the solution.

Table 5.4. GA operator values.

Number of Bits	Number of Population	Crossover Probability	Mutation Probability	Max No Generations
10	40	0.7	0.033	300
10	30	0.75	0.02	400

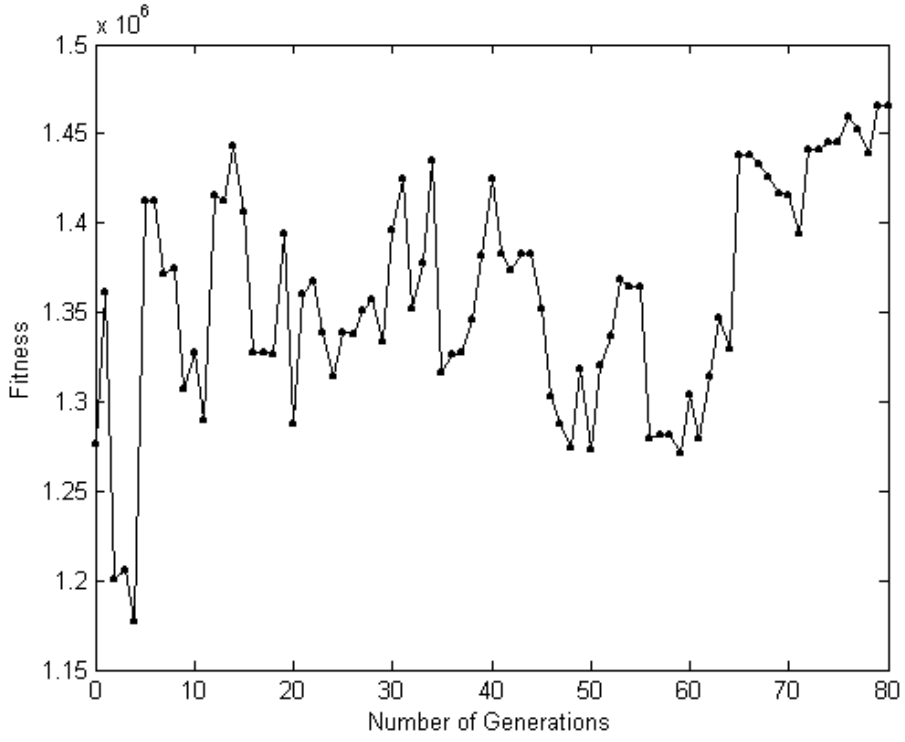


Figure 5.2. Fitness function growth with respect to number of generations.

6.4. Results

The identification method is tested on the following induction machines:

- Three theoretical examples of induction motors (3 hp, 50 hp and 2250 hp)
- Physical 7.5 hp induction motor.

6.4.1 Simulation Results

Before applying the genetic algorithm to the real experimental test data, a simulation study was conducted to prove the validity of the approach. The theoretical examples are chosen with widely different horsepower ratings in order to confirm applicability of the method to motors with various sizes. The following tables

summarize the results obtained from running the GA and selecting the best two data sets for each machine. The estimated parameters agree very closely with the actual value.

Table 5.5. Estimated parameter values of the 3 hp induction motor.

Data Set	R_s	X_{ls}	X_m	$X_{lr'}$	$R_{r'}$
1	0.3971	0.6839	26.4829	0.7985	0.8141
2	0.395	0.750	25.82	0.723	0.803
Actual	0.435	0.754	26.13	0.754	0.816

Table 5.6. Estimated parameter values of the 50 hp induction motor.

Data Set	R_s	X_{ls}	X_m	$X_{lr'}$	$R_{r'}$
1	0.0897	0.3031	13.0229	0.3295	0.2300
2	0.0839	0.2503	13.0347	0.2527	0.2395
Actual	0.087	0.302	13.08	0.302	0.228

Table 5.7.Estimated parameter values of the 2250 hp induction motor.

Data Set	R_s	X_{ls}	X_m	$X_{lr'}$	$R_{r'}$
1	0.0359	0.2003	13.0694	0.2020	0.0215
2	0.0274	0.2443	12.7507	0.1375	0.0172
Actual	0.029	0.226	13.04	0.226	0.022

Figures (5.3 – 5.11) compare the actual parameter response with that predicted by using the GAs identified parameters. Rotor speed, three-phase stator current and torque during transients are shown. For these results the two different cases were analyzed:

1. Stator voltages as the only input to the induction motor.

2. Stator voltages and rotor speed as the input to the induction motor.

From the tables given above it is obvious that the GA was able to identify parameters with a good accuracy. The errors between the responses of the induction motor with the actual parameters and the genetic algorithm identified parameters are very small.

6.4.1 Experimental Results

In this section the GA method was evaluated with experimental data collected in the laboratory for a wound rotor 7.5 hp induction motor. Several different operating conditions were recorded during a free acceleration test. With the assumption that parameters are changing during different time intervals, GA was run several times for each of these short periods. The best results obtained from each set are selected in the terms of maximum fitness value. Simulation results are summarized in Table 5.8. Figures (5.12 – 5.17) compare the actual and predicted parameter responses.

Table 5.8. Estimated nonlinear parameter values of the 7.5 hp induction motor.

No.	Time Interval	R_s	X_{ls}	X_m	$X_{lr'}$	$R_{r'}$
1.	0.3139	0.2762	0.2759	16.4809	0.250	0.2865
2.	0.6140	0.4	0.2043	19.1105	0.2008	0.2996
3.	0.9141	0.3918	0.2078	15.2590	0.2043	0.2980
4.	1.2142	0.2762	0.2759	16.4809	0.2500	0.2867
5.	1.5143	0.3815	0.2219	12.6197	0.2239	0.2582
6.	1.8144	0.3774	0.2043	16.0117	0.2149	0.2167
7.	2.1145	0.3830	0.2051	18.8954	0.2121	0.1829
8.	2.4146	0.4769	0.7161	19.1105	0.7601	0.3804
9.	2.7147	0.4488	0.6827	17.9179	0.6698	0.4488
10.	3.0	0.3818	0.5695	18.7195	0.4721	0.2758

In order to prove the robustness of the method, induction machine is simulated with two different operating conditions:

- Balanced three-phase input stator voltages 200 V line – to line.

- Balanced three – phase input stator voltages 220V line – to line.

It can be noted that even difference between the input voltages is not that big, resulting output responses of rotor speed, stator and rotor currents would be affected by this difference. It can be seen from the responses that GAs seem to fit data well for both of these conditions. Not surprisingly, it performs better using the data representing high transients of the machine since there is more information available on the model. After a period of 1.7 seconds (200V operating condition) or 1.2 seconds (220V operating condition) when the machine is approaching steady state and when inrush stator currents are reaching steady state values, estimation of the parameters becomes a difficult problem. One possible reason for this is that during the steady state rotor currents would become infinitesimally small.

After making comparison between measured stator currents and those simulated with the parameters obtained in power laboratory following IEEE standard test, it can be noted that at the very first instance of time, when slip is almost unity, results are matching quite well. Not surprisingly since IEEE standard test is performed at 60 Hz and this result was actually expected.

In conclusion, resulting responses with both GAs identified and IEEE standard determined test parameters seem to fit measured ones during steady state well.

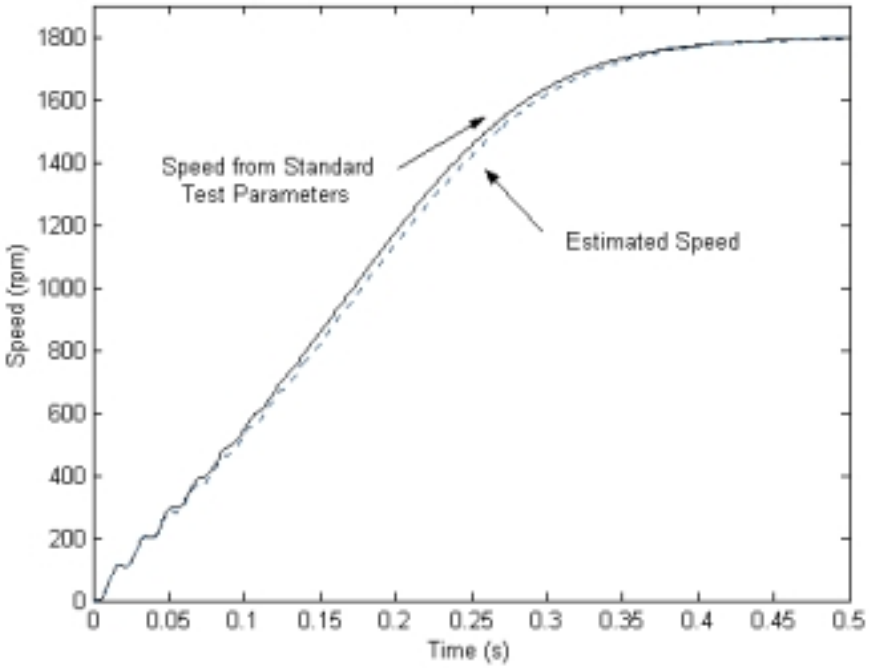


Figure 5.3. GAs estimated and actual rotor speeds of 3 hp induction motor. (Note: actual parameter values are taken from [6] and they are obtained by IEEE standard test).

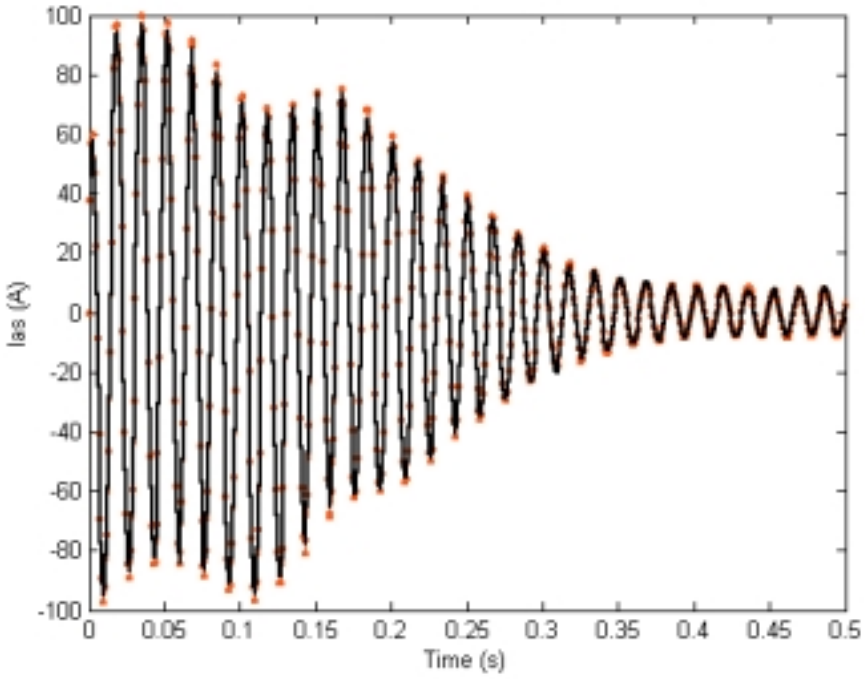


Figure 5.4. GAs estimated and actual stator currents of 3 hp induction motor. (black solid– actual, black dotted – estimated).

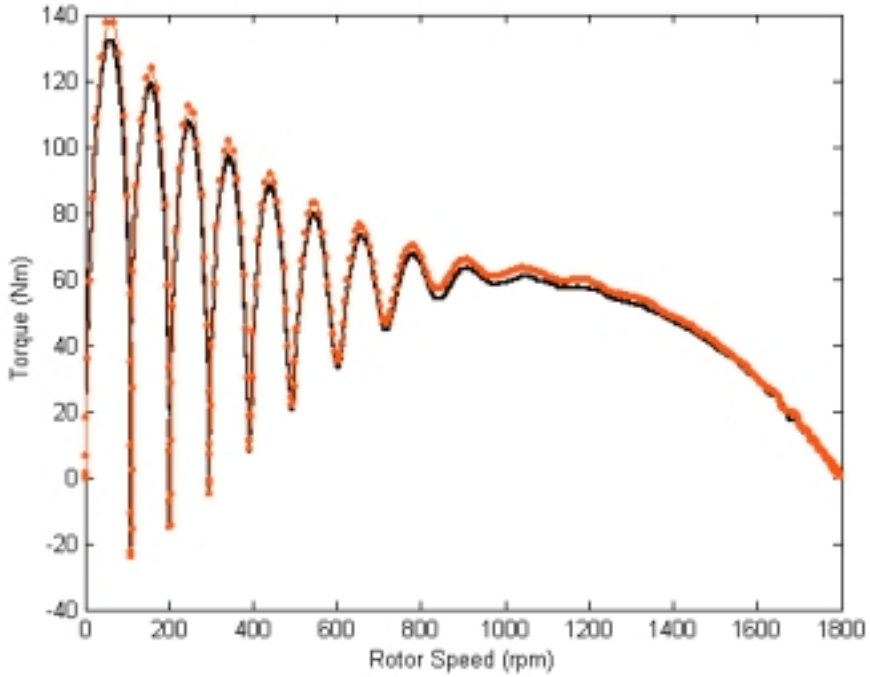


Figure 5.5. GAs estimated and actual torques of 3 hp induction motor (black solid– actual, black dotted – estimated).

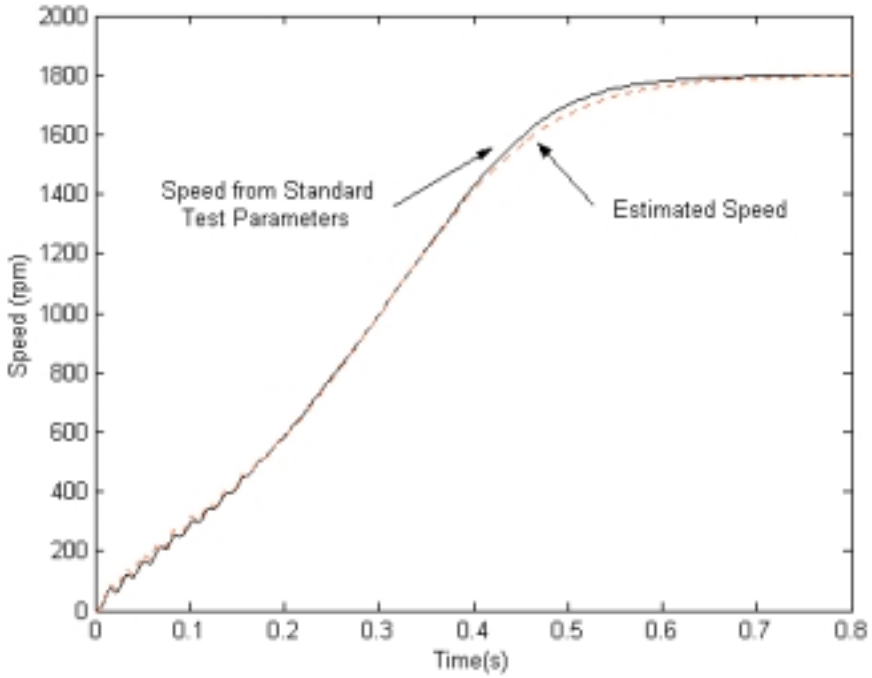


Figure 5.6. GAs estimated and actual rotor speeds of 50 hp induction motor. (Note: actual parameter values are taken from [6] and they are obtained by IEEE standard test).

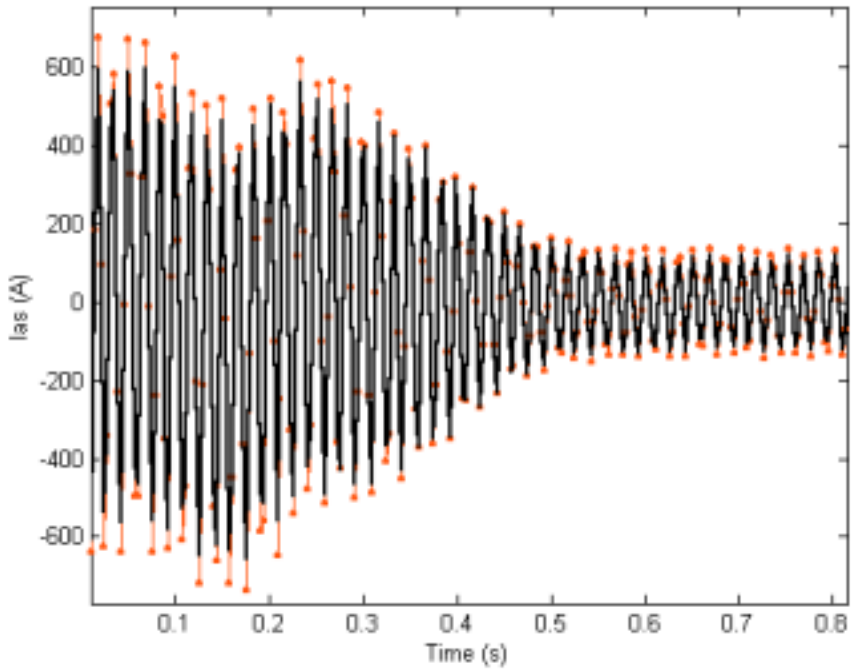


Figure 5.7. GAs estimated and actual stator currents of 50 hp induction motor. (black solid – actual, black dotted – estimated).

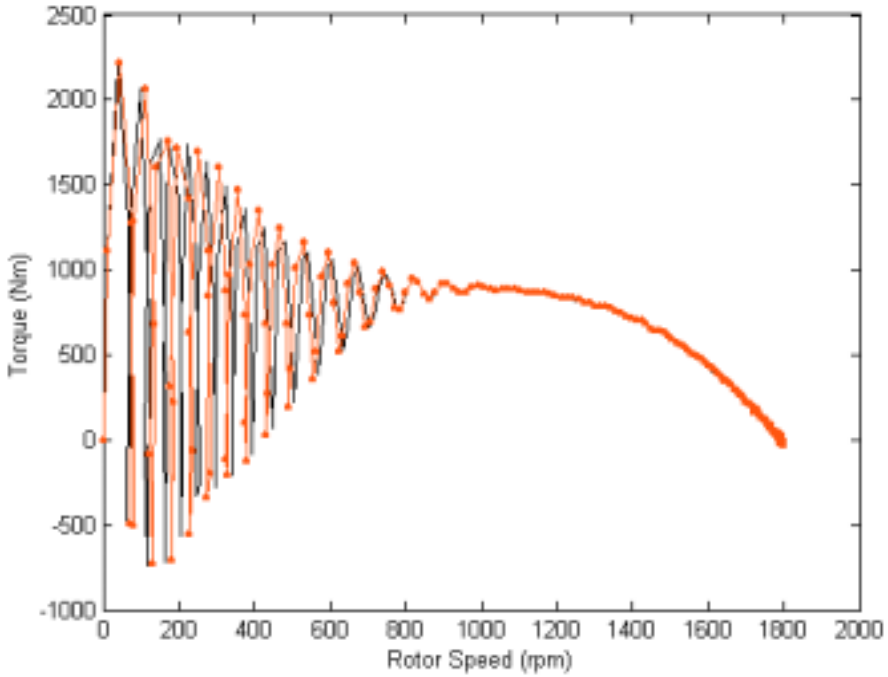


Figure 5.8. GAs estimated and actual torques of 50hp induction motor.(black solid– actual, black dotted – estimated).

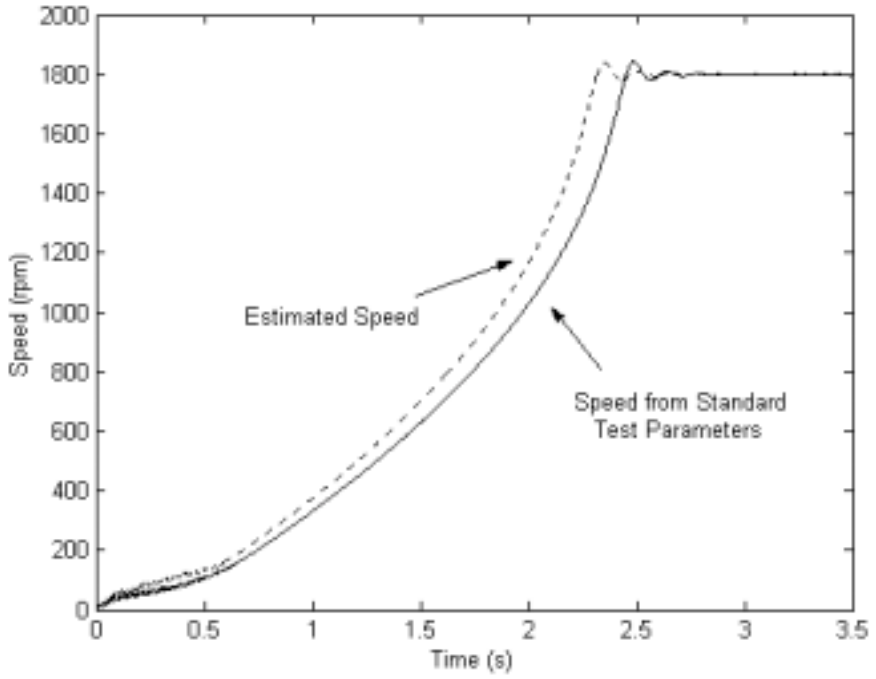


Figure 5.9. GAs estimated and actual rotor speeds of 2250 hp induction motor. (Note: actual parameter values are taken from [6] and they are obtained by IEEE standard test).

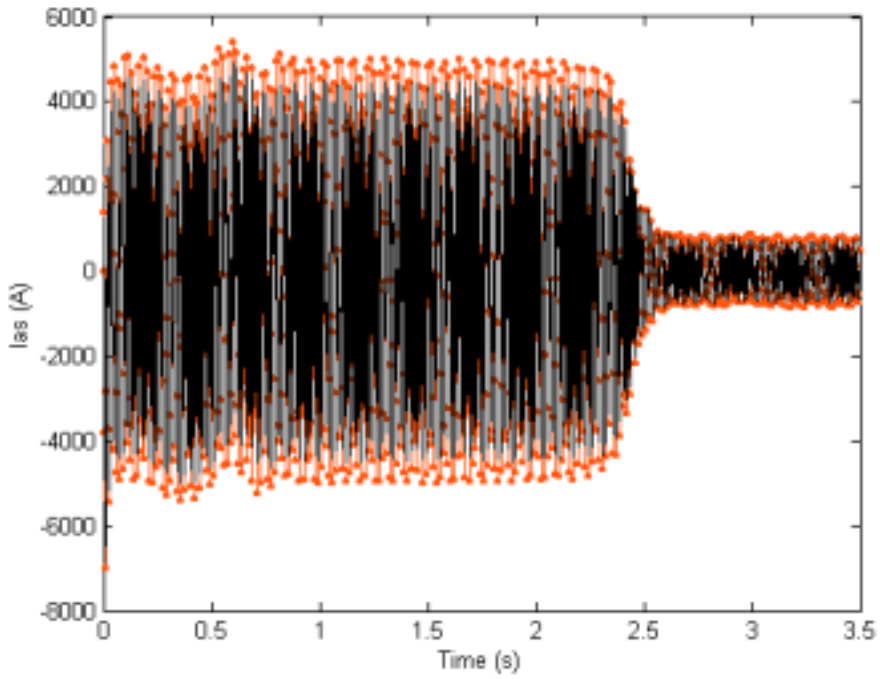


Figure 5.10. GAs estimated and actual stator currents of 2250 hp induction motor. (black solid– actual, black dotted – estimated).

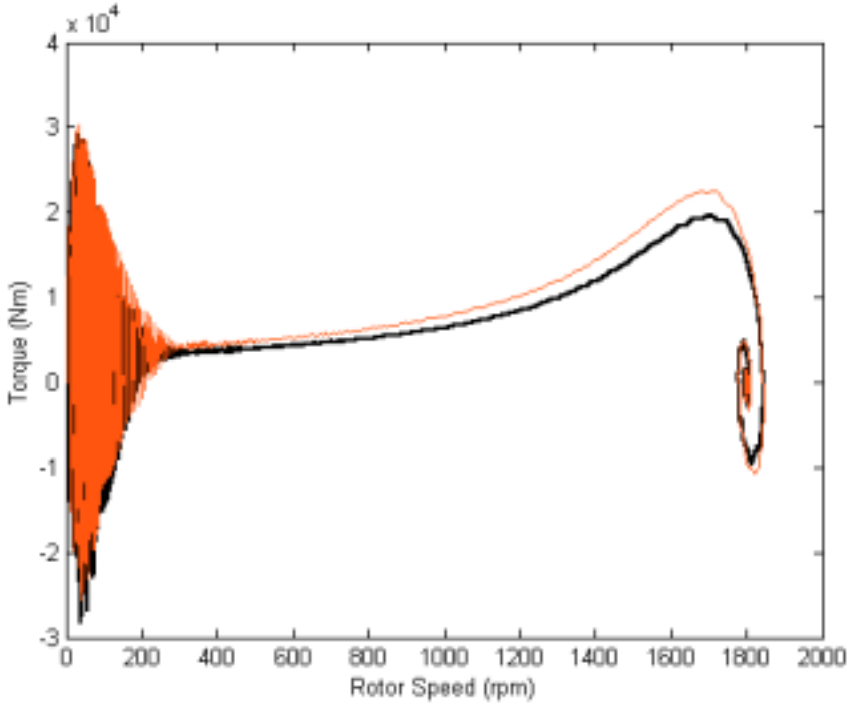


Figure 5.11. GAs estimated and measured torques of 2250 hp induction motor.(black solid– actual, black dotted – estimated).

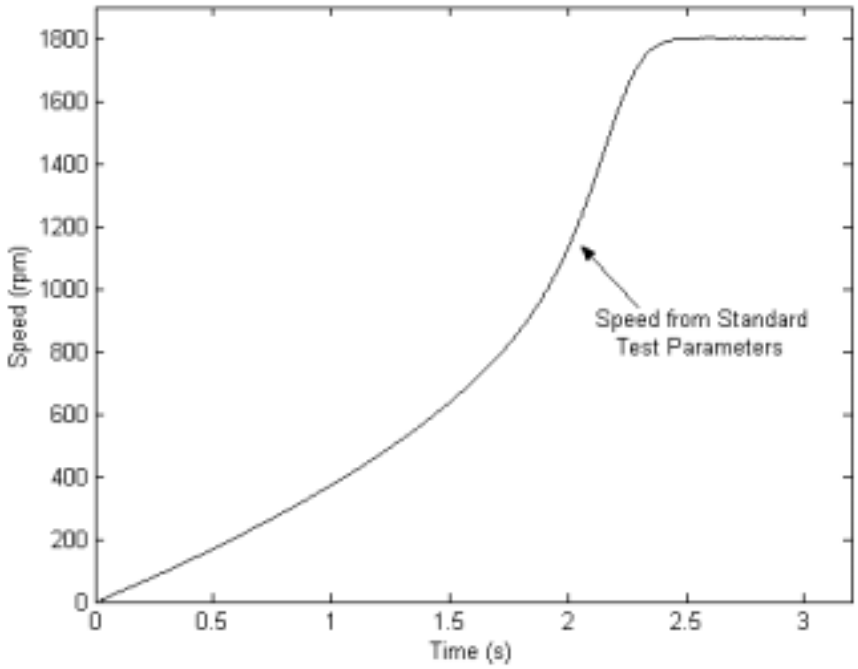


Figure 5.12. Rotor speed of 7.5 hp induction motor obtained with IEEE standard test parameters. (200V balanced three - phase stator voltage generated).

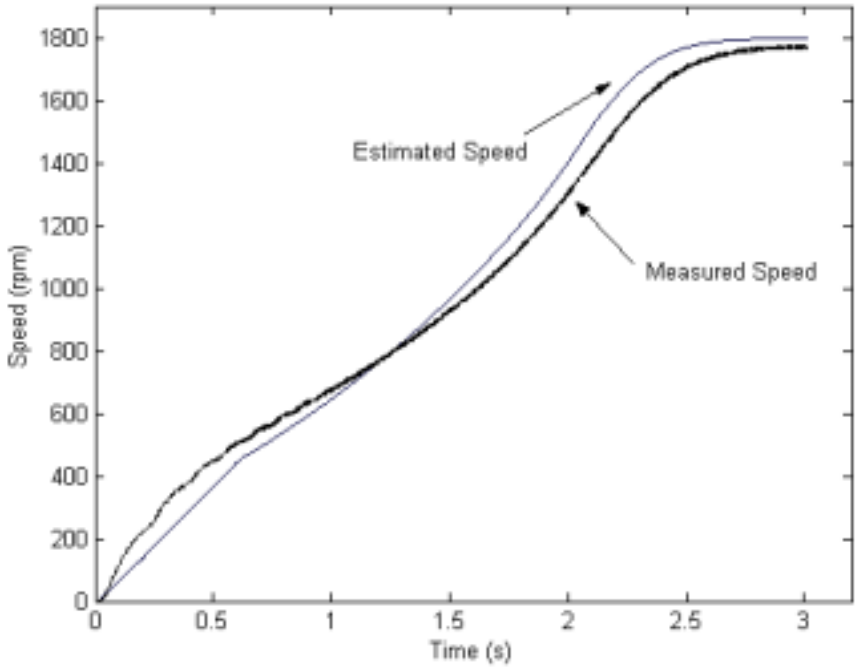


Figure 5.13. Estimated and measured rotor speeds of 7.5 hp induction motor. (200V balanced three – phase stator voltage generated).

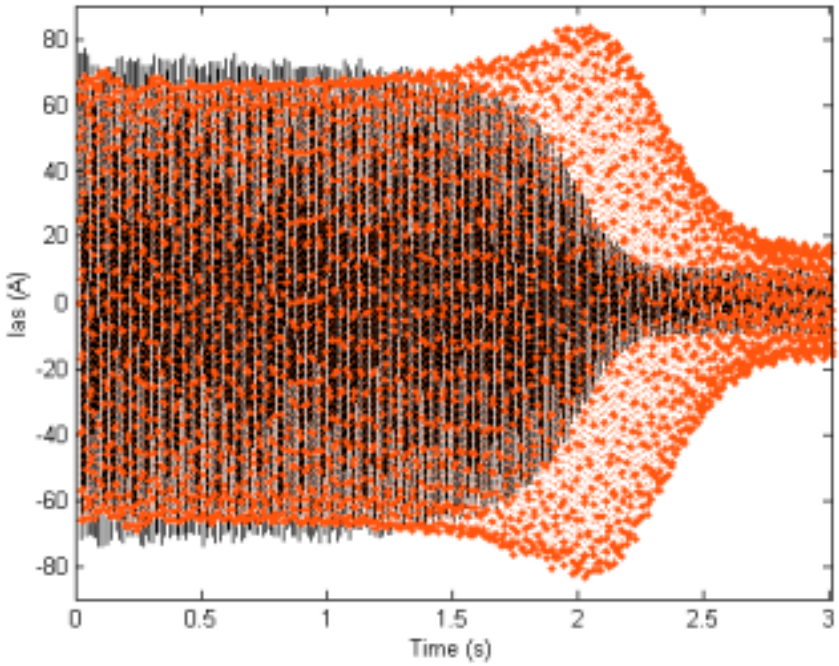


Figure 5.14. Simulated and measured stator currents of 7.5 hp induction motor. (200V). (black solid– measured, black dotted – simulated).

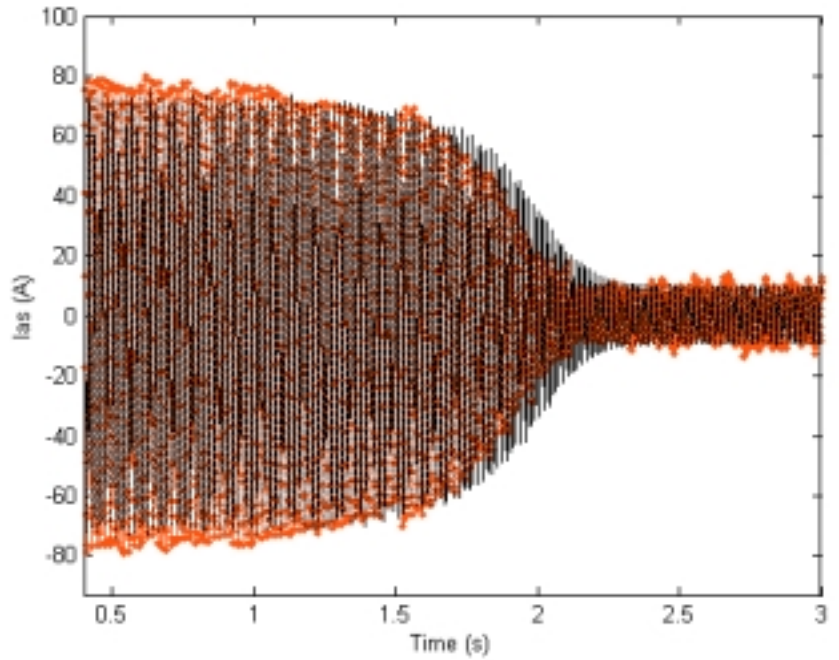


Figure 5.15. GAs estimated and measured stator currents of 7.5 hp induction motor. (200 V). (black solid– measured, black dotted – estimated).

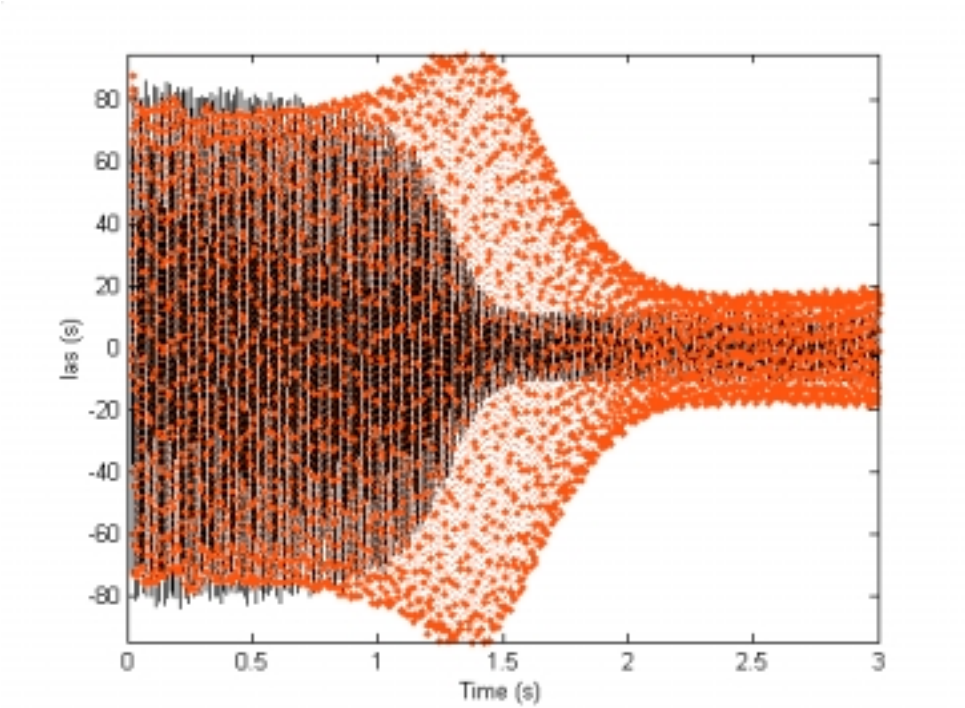


Figure 5.16. Simulated and measured stator currents of 7.5 hp induction motor. (220V). (black solid– measured, black dotted – simulated).

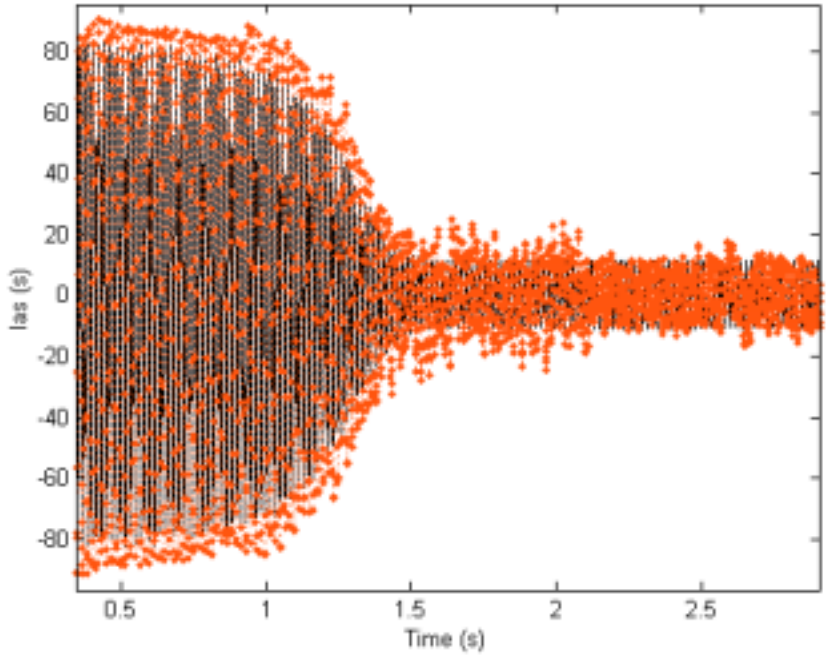


Figure 5.17. GAs estimated and measured stator currents of 7.5 hp induction motor. (220V). (black solid – measured, black dotted – estimated).

Chapter 6

Conclusion

Lab VIEW Sofware demonstrated its ability to easily acquire signals of the physical system, which is the induction motor in this work. The process, configured as a *virtual instrument* (VI), is developed in the Lab VIEW environment, which provides enormous help in parameter identification of induction machines. It is proved that Lab VIEW Sofware, with the flow of data between objects on a block diagram, allows easy buildup of diagrams that execute multiple operations simultaneously. Consequently, Lab VIEW VIs are also modular in design, so any VI can run on its own or be used as part of another VI (subVI). This feature is demonstrated by designing digital tachometer as subVI inside of complete virtual instrument used in the acquiring process. This characteristic provides the user with the flexibility to design a whole hierarchy to serve as building blocks in any number of applications.

In the second part of the thesis a GA developed in Matlab [30] was used for identification of electric parameters of induction motor. A fifth order nonlinear model of an induction machine is selected for the study. A vector of five electric parameters is sought, which represents the dynamics of the induction machine. The nonlinear equations of the machine are expressed in the synchronously rotating reference frame. Free acceleration transient data, which contains more information on machine

nonlinearity is selected for study. Computer simulation of the machine is developed in Matlab environment, which can be conveniently changed in order to accommodate different reference frames. Two cases were studied:

- Theoretical validation of the method applied to several induction machines (3 hp, 50 hp and 2250 hp machine).

- Physical validation of the method applied to 7.5 hp induction motor in power laboratory.

A three – phase wound rotor 7.5 hp induction machine is tested at two different operating conditions (200 V and 220 V line – to – line balanced three phase voltages) and the free acceleration test data is collected with the help of Lab VIEW Software and DAQ.

The results obtained after the testing show that GA proved to be a good method for reproduction of the input – output behavior of induction motor. Furthermore, GA seems to be robust since results proved to be valid for two different operating conditions of the 7.5 hp induction motor. Since it was not necessary to filter the data or even to measure the rotor currents, the method has demonstrated its robustness to deficiencies in the data. Unlike the traditional *DC resistance*, *no load*, and *blocked rotor tests*, which assume that the parameters are constant, regardless of operating conditions, GA can find a model that is valid for a specific operating point. Also, experimental tests with Lab VIEW Software and DAQ confirm the ability of the tools to easily measure and create different interfaces by using this graphical development.

GA proved to be an accurate off line algorithm for this type of the problem. It can be further improved by:

- Incorporating advanced GA techniques to increase the convergence speed and accuracy of the proposed method.

- Studying different performance indices in order to select the best pay off function to this type of problem.
- Utilizing data sets at different loading conditions in order to obtain more accurate control of the machine.

Appendix A

Hardware Setup and Lab View Software

The following document gives information on hardware instruments and equipments, which are used for acquiring the signals in the experiment. The second part describes a general purpose of Lab VIEW Software designed to easily acquire external signals. It is included here because of the applicability to further work on this subject.

Experiment Setup

The identification method described in the preceding section is applied to a 7.5 hp Induction Motor. The following figure shows the picture of the motor along with the boilerplate information from the motor. When the machine is at standstill without mechanical load, the three – phase AC power is applied to the stator terminals. The induction motor is supplied with the power from motor – generator group which data can be found in Table A1. Three - phase voltages applied to induction motor are changed in order to record several different operating conditions of the machine during free acceleration. The experimental setup along with voltage and current transducers is shown in the following Figure A2. The three - phase stator voltages, three – phase stator currents and the rotor speed are recorded by the means of voltage and current transducers respectively and tachometer, which is used to measure the speed of the rotor shaft.

General Electric Induction Motor	
Type: M	Model: 5M2 84E955
HP: 7.5	Volts: 110/220
Rating: Cont	Cycles: 60
Frame 284	RPM: 1715
Phase 3	Service Factor: 40
Fl Amp: 41.4/20.7	Nema Design: C
Sec Amp: 23.1	Sec Volts: 151

Figure A1. 7.5 hp induction motor and boiler plate information used in final testing.

Table A1. Boiler plate information for motor generator group.

Westinghouse Synchronous Motor	
HP: 30	Exc Amp: 5.95
Volts: 220	Exc Volts: 115
Amperes: 66	%Load 100
PF % 100	Hours 24
Phase 3	Temp Rise: 40
Cycles: 60	RPM: 1200

Westinghouse AC Generator	
15 KVA	Cycles: 60
Volts: 120/240	RPM: 1200
Amps: 72/36	Exc Amps: 13
PF % 80	Exc Volts: 115
Phase 3	S.O. 18W637

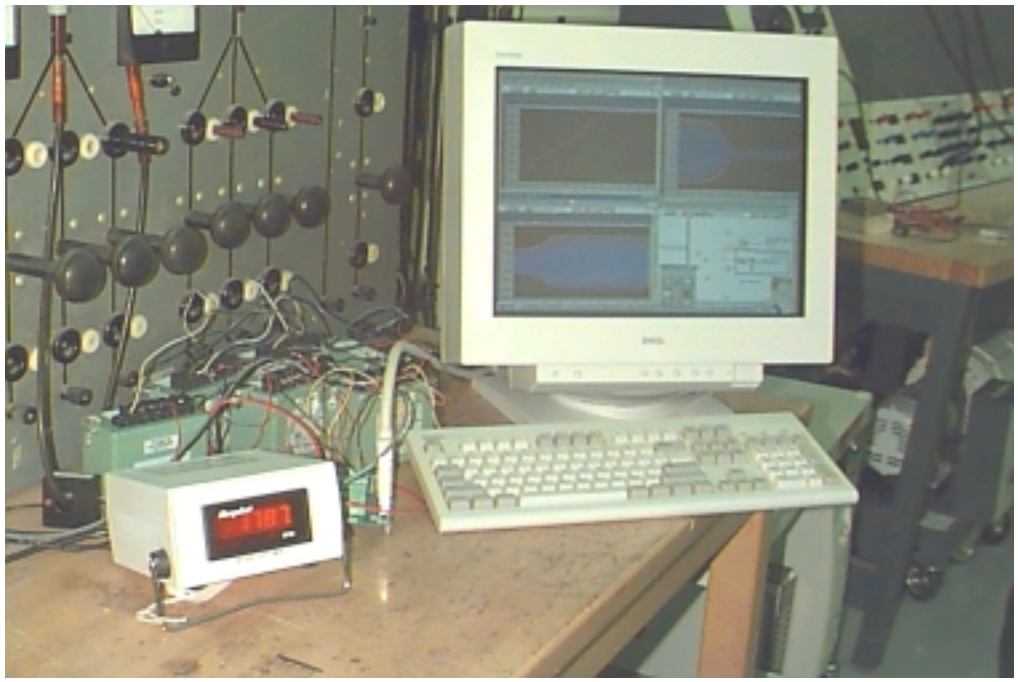


Figure A2. Experimental setup in power laboratory.

Data Acquisition System is used to acquire currents, voltages and speed waveforms. The signal data are saved on the spreadsheet files in Lab VIEW Environment. (See reference [35] for more details). This data acquisition system uses a National Instrument DAQ Card (6040E Families). The card has been installed on the PCI bus of the computer in the power lab (shown in the picture). The I/O Connector of the DAQ card has 68 pins and is connected to a protected I/O connector (SCB-68: which has 68 screw terminals) via a SH6868 shielded cable. (Gray cable).

Both, the analog and digital signals can be connected to the SCB-68. To measure currents and voltages of a three-phase system, there are six $\pm$10 Volts analog inputs. For measurement of speed, there is a 0 to 5 Volt analog input. The analog input channels are:

- Three inputs for measuring Current. (Channels 0, 1, 2).
- One input for measuring Speed (Channel 3).
- Three inputs for measuring Voltage. (Channel 4, 5, 6).

These Signals are connected to SCB-68 pins in the following order:
==========
Channel Number: 0
Channel Name: Current1
Pin Numbers: 68, 34 (68 positive, 34 Negative)
==========
==========
Channel Number: 1
Channel Name: Current2
Pin Numbers: 33, 66 (33 positive, 66 Negative)
==========
==========
Channel Number: 2
Channel Name: Current3
Pin Numbers: 65, 31 (65 positive, 31 Negative)
==========
==========
Channel Number: 3

Channel Name: Tachometer
Pin Numbers: 30, 63 (30 positive, 63 Negative)
==========
==========
Channel Number: 4
Channel Name: Voltage1
Pin Numbers: 28, 61 (28 positive, 61 Negative)
==========
==========
Channel Number: 5
Channel Name: Voltage2
Pin Numbers: 60, 26 (60positive, 26 Negative)
==========
==========
Channel Number: 6
Channel Name: Voltage3
Pin Numbers: 25, 58 (68 positive, 34 Negative)

It should be noted that all DAQ card analog inputs are voltages in the range of -10 to +10 V. (Note: if the input signal exceeds +10/-10 V, the card will be damaged). Therefore, in order to measure high current or voltages or speeds, special transducers are needed which convert high amplitude signals within the range of -10/+10. The following figure shows the transducers connected to DAQ used in final testing.

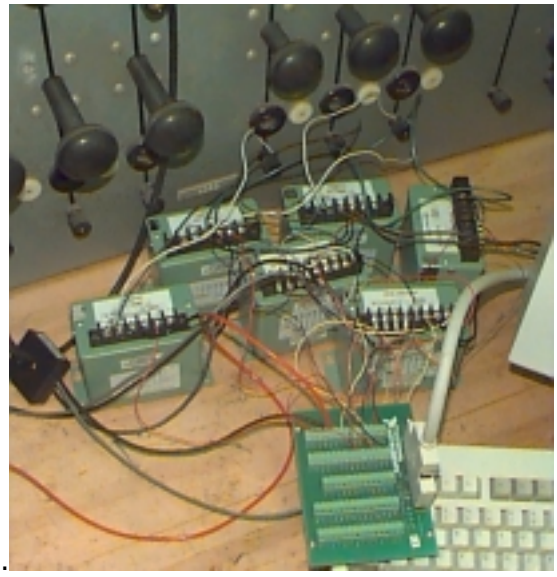


Figure A3. Voltage and current transducers with DAQ connector.

They are mainly two different transducers used in this experiment: Hall effect current transducers and AC voltage transducers. Speed is measured using a digital tachometer. The following paragraphs give more information on this.

Current Transducers (CTL-51, CTA201R):

CTL-51 is a Hall effect sensor and CTA201R is its transducer. These transducers measure high currents in the range of -35/+35 A and produce an output signal proportional to the input. The output range is a -10/+10 Volts.

Fig. A4 depicts the connection between CTA and CTL. The wire, the current through which is to be measured, goes through the CTL window, e.g. refer to line L1 going through the CTL window. So, for the three-phase system, there are three transducers (CTL/CTA), which measure the currents of phases A, B and C. Each line goes through the window of its corresponding sensor (CTL). The current instrument requires 115V power supply (refer to INST PWR connections to pins 7 and 8 in Figure A4).

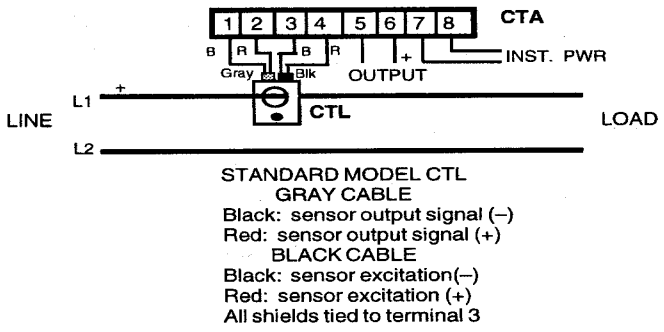


Figure A4. CTA/CTL connections.

The output signal (pins 5,6) is connected to SCB-68 current channel pins for each phase. For example, for channel 0 which measures the current of phase A, one should connect pin 6 of CTA to pin 68 of SCB-68 and similarly pin 5 to 34.

Voltage Transducers (VT7):

These transducers measure high voltages in the range of -500/+500 V and produce an output signal proportional to the input. The output range is -10/+10 V. These transducers have 1500 V isolation and are able to measure input high voltage signals with frequencies up to 10 kHz. For each phase, a VT7 transducer is required. The VT7 power supply is 115 VAC, which should be connected to pins 3,4. The following paragraphs give information about connections:

Pin Connections:

Pins 1,2: Input AC High Voltage Signal (-500/+500v).

Pins 3,4: Power Supply (115 Volt)

Pins 5,6: Output AC Low Voltage Signal (-10/+10v) which are connected to SCB-68 Voltage terminals.

For Voltage1 Channel of SCB-68, Pin numbers 28, 61 are connected to the output pins of the first VT7 (pins 5,6).

For Voltage2 Channel of SCB-68, Pin numbers 60, 26 are connected to the output pins of the second VT7 (pins 5,6).

For Voltage3 Channel of SCB-68, Pin numbers 25, 58 are connected to the output pins of the third VT7 (pins 5,6).

The three VT7 transducers measure voltages of Phases A, B and C respectively.

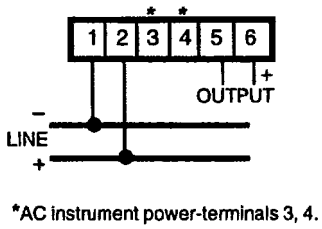


Figure A5. VT7 connection diagram

Speed Transducer (Digital Tachometer)

This transducer measures the speed of the rotating shaft by a reflective photo sensor. It has a digital indicator, which shows the measuring speed. It also has a DC Analog output, which is proportional to the measured speed and should be connected to SCB-68 terminals. Pin 30 (SCB-68) should be connected to positive tachometer output (RED) and pin 63 to negative tachometer output (BLACK). This Tachometer has a transducer, which measures speed in the range of 50-10000 rpm and provides a proportional output of 0- 5 volts, which is connected to DAQ system.

Table A2. Connections for a three-phase system.

Channel	**Type of Transducer**	**Phase**	**Transducer Pins**	**SCB-68 Pins**
Current 1	CTL51	**A**	5,6	68,34
Current 2	CTL51	**B**	5,6	33,66
Current 3	CTL51	**C**	5,6	65,31
Tachometer	HPT601	**-**	RED, BLACK	30,63
Voltage 1	VT7	**A**	5,6	28,61
Voltage 2	VT7	**B**	5,6	60,26
Voltage 3	VT7	**C**	5,6	25,58

After making the connections given in Table A2 currents, voltages and speed are ready to be measured. Lab VIEW Software helps one to design programs for measurement purposes. For this DAQ system, two files have been designed: Measure and Save Data to a File.vi , and Read Data from a File.vi .

Measure and Save Data to a File.vi: Acquire voltages, currents and speed of the three-phase system and save the measured data on a spreadsheet files. *(Front Panel).*

Read Data from a File.vi: Read the data, which is already saved on a file and has the ability to zoom different parts of the display signals. *(Front Panel).*

Measure and Save Data to a File.vi :

Upon measurements of any of the input signals the following settings should be available:

- Device (1) equal to 1.
- Channels equal to Current1, Current2 and Current3.
- Scan rate should be equal to 1000 scans/rate. (1000 scans per second for each channel).
- Buffer size should be equal to 4000.
- Scans to be read at a time equal to 100. (Each screen refreshes after 100 samples).
- Scan back log indicates 0.

Before running, one should to be careful that the Y-axis scales are in the same range as the measured values. For example, currents with 20A peak values, the Y-axis range should be between -25 to +25A. In order to change the Y-axis scale user must be in STOP mode. After all the steps above are accomplished, RUN button can be activated. In the case RUN button is indicate as broken, there is a problem with the program. After activating RUN, the next step is to enter the file name in the desired path and save the information. The program starts running and any acquired signals will be displayed on the screen at that moment and at the same time saved on the spreadsheet files as .TXT files. In these files, there is one column for each channel.

Read Data from a File.vi :

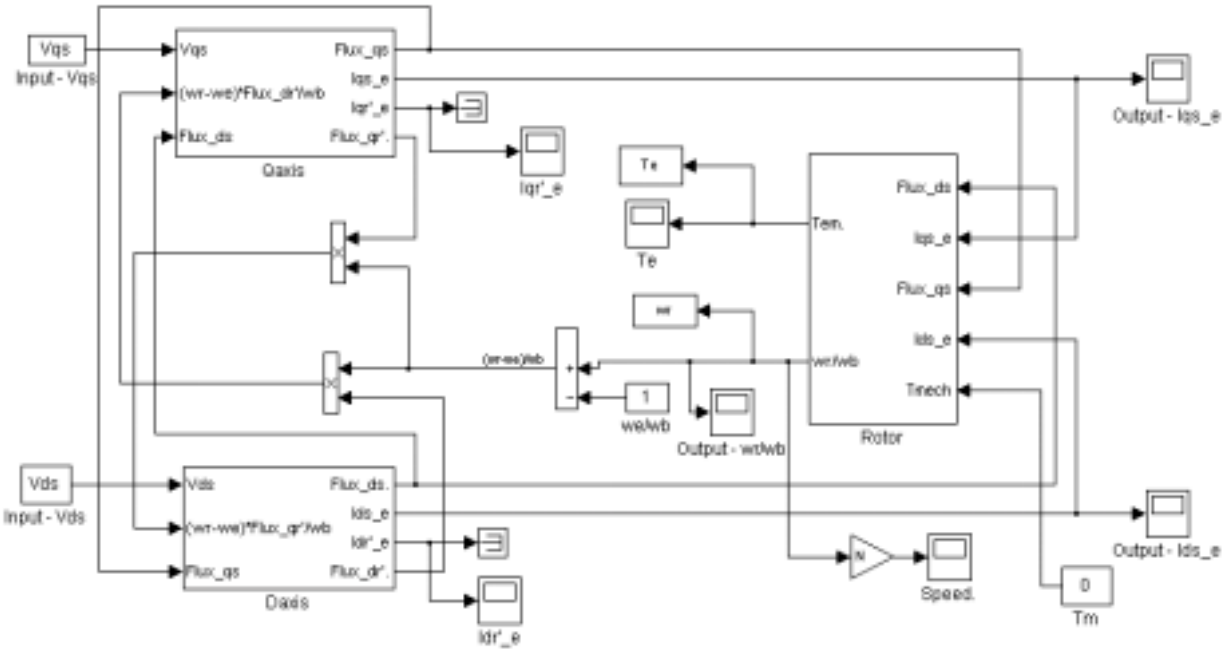
Read Data from a File.vi is a program that enables one to open the spreadsheet file created by means of the Measure and Save Data to a File.vi Program. Upon reading the acquired data the following settings should be available:

- Scan rate must be equal to the one used in Measure and Save Data to a File.vi
- Set the start time of reading signals. (It is usually set up to zero).
- Set the time interval in which the signal is acquired.
- Make sure that Y – axis scale is large enough in comparison to the maximum value of the acquired signals.
- Run the Lab View program and click on the open VI and specify your desired filename.
- The complete saved data will be displayed as the waveform from starting time to the final time specified by the user.

Another note is that this program allows the user to zoom and display different part of data. In order to perform this operation, user should specify starting and final time of the interest and to make sure that the scan rate is the same as it was sampled and saved by Measure and Save Data to a File.vi file; otherwise, the data will not be displayed or the user might see the wrong waveforms.

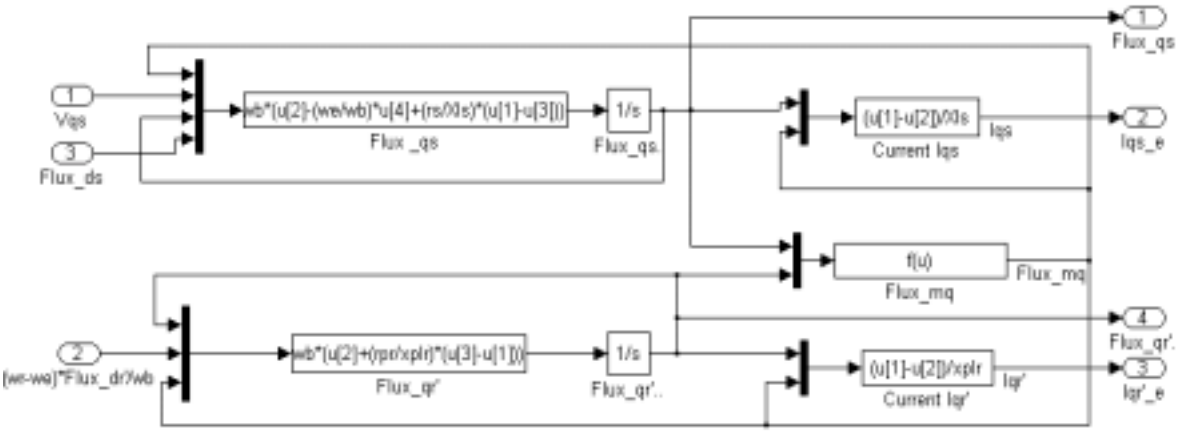
Appendix B

Simulink Model



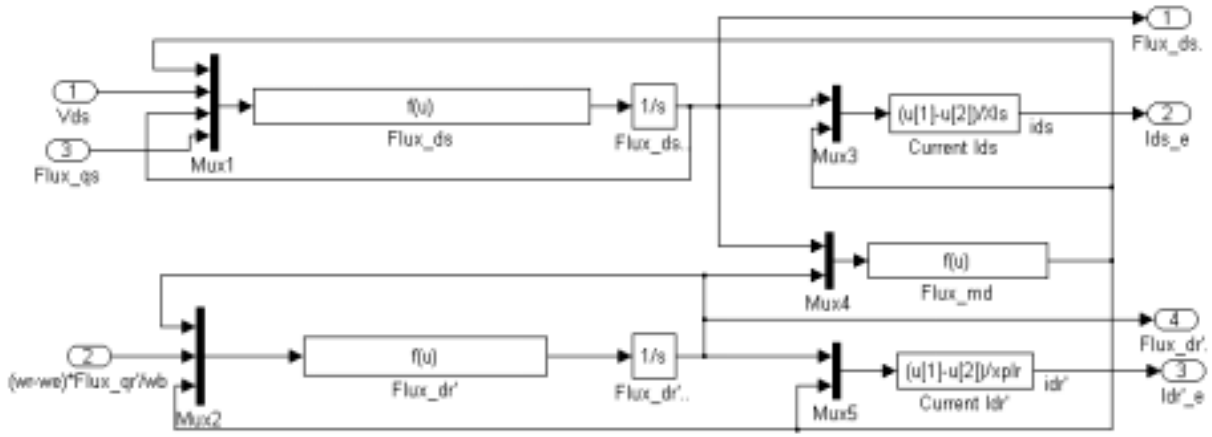
a.) Overall diagram of induction machine.

Figure B1. Simulation of a three - phase induction machine in the synchronously rotating reference frame.



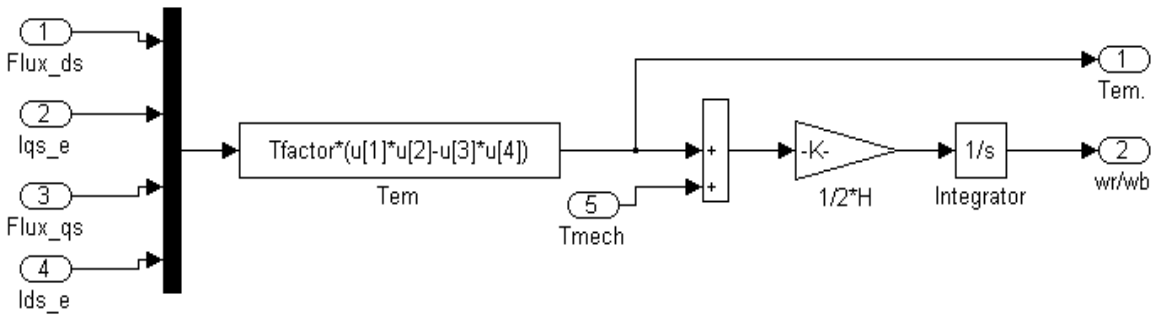
b.) Inside Q axis block.

Figure A2. (*cont*.): Simulation of a three – phase induction machine in the synchronously rotating reference frame.



b.) Inside D axis block.

Figure A3. (*cont*.): Simulation of a three – phase induction machine in the synchronously rotating reference frame.



b.) Inside rotor block.

Figure A4. (*cont*.): Simulation of a three – phase induction machine in the synchronously rotating reference frame.

Appendix C

Source Code

Simulation of the Induction Machine

```
%Simulation of the induction Machine following the reference [7].
%Objective is to solve nonlinear differential equation (3.10) and
%(3.14). The file "Flux" calculated the derivative of the state
%variables, which are defined as fluxes.

global fo Vs w wb WrCoeff TeCoeff
global rs Xrrp D Xm Xss Xlrp Xls rs rrp

choice = menu('Select Induction Machine','3 hp Induction Motor','2250
hp Induction Motor',
'50 hp Induction Motor','500 hp Induction Motor');
if choice == 1
   % Parameters for 3-hp machine
   P = 4;         fo = 60;
   Vs = 220/sqrt(3);
   rs = 0.435;    Xls = 0.754;   Xm = 26.13;    Xlrp = 0.754;
   rrp = 0.816;   J = 0.089; Pn=3*746.7;
   Tf = 0.5;
   Mach = '3-hp machine';
elseif choice == 2
   % Parameters for 2250-hp machine
   P = 4;         fo = 60;
   Vs = 2300/sqrt(3);
   %rs = 0.029;    Xls = 0.226;   Xm = 13.04;    Xlrp = 0.226;   rrp =
0.022;
   rs=0.0359; Xls= 0.2003; Xm=13.0694; Xlrp=0.2020; rrp=0.0215;
   J = 63.87;
   Tf = 3.5;
   Mach = '2250-hp machine';
elseif choice ==3
    % Parameters for 50-hp machine
   P = 4;         fo = 60;
   Vs = 460/sqrt(3);
```

```
   rs = 0.087;    Xls = 0.302;   Xm = 13.08;    Xlrp = 0.302;
   rrp = 0.228;   J = 1.662;
   Tf = 1.8;
   Mach = '50-hp machine';
elseif choice ==4
    % Parameters for 500-hp machine
   P = 4;         fo = 60;
   Vs = 2300/sqrt(3);
   rs = 0.262;    Xls = 1.206;   Xm = 54.02;    Xlrp = 1.206;
   rrp = 0.187;   J = 11.06;
   Tf = 1.8;
   Mach = '500-hp machine';
end

we=2*pi*fo;
w = we;   % Work in the synchronous /stationary reference frame

Tstep=Tf/2000;

tv=(0:Tstep:Tf);     %Time vector for Vqdos
we = 2*pi*fo;

%Define voltages of three phases a, b, c as input to the model

Vas = sqrt(2)*Vs*cos(we*tv);
Vbs = sqrt(2)*Vs*cos(we*tv - (2*pi/3));
Vcs = sqrt(2)*Vs*cos(we*tv + (2*pi/3));

Vabcs = [Vas; Vbs; Vcs];

% Convert Vabcs to the QDO reference frame.

Vqdos = zeros(3, length(tv));
for pt = 1:length(tv);
   th = w*tv(pt);

   Ks = 2/3*[cos(th) cos(th-(2*pi/3)) cos(th+(2*pi/3)); ...
             sin(th) sin(th-(2*pi/3)) sin(th+(2*pi/3));  .5 .5 .5];
   Vqdos(:,pt) = Ks*Vabcs(:,pt);
end


%Solution for the Equations (3.10) and (3.14) will follow directly
%after calling ode solver. Results of the state space equation are
%flux variables.

Xss  = Xls  + Xm;
Xrrp = Xlrp + Xm;
D = Xss*Xrrp - Xm^2;
wb = we;

WrCoeff = P/(2*J);
TeCoeff = .75*P*Xm/(D*wb);
```

```
C = 1/D*[ Xrrp        0         0       -Xm         0          0;
           0        Xrrp        0        0         -Xm         0;
           0          0       D/Xls      0          0          0;
          -Xm         0         0       Xss         0          0;
           0        -Xm         0        0         Xss         0;
           0          0         0        0          0       D/Xlrp];

FL0 = zeros(8,1);
[t, FL] = ode15s('Flux',tv,FL0);
Iqdos = C(1:3,:)*FL(:,1:6).';   % Equation (3.14)
Iqdor = C(4:6,:)*FL(:,1:6).';   % Equation (3.14)
wr = FL(:,7);
theta = FL(:,8); % Theta-r

rpm = wr*(2/P)*60/(2*pi);  % Mechanical rpm

Te  =  TeCoeff*(FL(:,1).*FL(:,5)  -  FL(:,4).*FL(:,2));     %  equation
(3.16)

% Convert Iqdo back to Iabc

Iabcs = zeros(3, length(t));
Iabcr = zeros(3, length(t));


th1=w*t;
for pt = 1:length(t);
   th = th1(pt);

   Ks_inv = [cos(th) sin(th) 1; cos(th-(2*pi/3)) sin(th-(2*pi/3)) 1;
...
             cos(th+(2*pi/3)) sin(th+(2*pi/3)) 1];
   Iabcs(:,pt) = Ks_inv * Iqdos(:,pt);

   thr = theta(pt);
   b = th-thr;
   Kr_inv = [cos(b) sin(b) 1; cos(b-(2*pi/3)) sin(b-(2*pi/3)) 1; ...
             cos(b+(2*pi/3)) sin(b+(2*pi/3)) 1];
   Iabcr(:,pt) = Kr_inv * Iqdor(:,pt);
end
```

```
function Flux = Flux(t,FL);
% FL = a 8-vector = [The 6 flux linkages in (3.10); Omega-r; Theta-r];
% This function calls ode15s (one of the differential solvers) to
%obtain solution for the fluxes Equation (3.10).

we = 2*pi*fo;   %Omega-e, the angular frequency of the synchronous
frame

w=we; % Work in the synchronous reference frame

% Conversion of three phase voltages to synchronous QDO frame
```

```
Vas = sqrt(2)*Vs*cos(we*t);
Vbs = sqrt(2)*Vs*cos(we*t - (2*pi/3));
Vcs = sqrt(2)*Vs*cos(we*t + (2*pi/3));
Vabcs = [Vas; Vbs; Vcs];

th = w*t;

Ks = 2/3*[cos(th) cos(th-(2*pi/3)) cos(th+(2*pi/3)); ...
          sin(th) sin(th-(2*pi/3)) sin(th+(2*pi/3));  .5 .5 .5];

Vqdos = Ks*Vabcs;

Vqdor = [0;0;0];

% Equation (3.10) in the state space form:

wr=FL(7);

A = -wb*[rs*Xrrp/D    w/wb      0    -rs*Xm/D       0          0;
         -w/wb     rs*Xrrp/D    0        0      -rs*Xm/D       0;
           0           0      rs/Xls     0          0          0;
         -rrp*Xm/D     0        0    rrp*Xss/D  (w-wr)/wb      0;
           0       -rrp*Xm/D    0   -(w-wr)/wb  rrp*Xss/D      0;
           0           0        0        0          0      rrp/Xlrp];

Flux = zeros(8,1);
Flux(1:6) = A*FL(1:6) + wb*[Vqdos; Vqdor];
Te = TeCoeff*(FL(1)*FL(5) - FL(4)*FL(2));     % (equation 3.16)
Flux(7) = WrCoeff*Te;     % The equation for d(wr)/dt is (3.17)
Flux(8) = FL(7);          % Theta-r = integral(wr)
```

Genetic Algorithm Implementation

```
% This source code runs GA and as a result gives the optimized
% parameters with the maximum fitness value.
% Genetic Algorithm M-files used here are from MathWorks, Inc. [27]
% They are adjusted to this problem optimization.
% The original algorithm implemented here is taken
% from Genetic Algorithms in Search, Optimization, and Machine
% Learning,David E. Goldberg.

% The following Matlab Functions are used in the implementation:
```

```
function [xopt,stats,options,bestf,fgen,lgen] = genetic(fun, ...
          x0,options,vlb,vub,bits,P1,P2,P3,P4,P5,P6,P7P,P8,P9,P10)
%GENETIC tries to maximize a function using a genetic algorithm.
%       X=GENETIC('FUN',X0,OPTIONS,VLB,VUB) uses a genetic algorithm
%       to find a maximum of the fitness function
%       FUN (usually an M-file: FUN.M).  The user may define all or
%       part of an initial population X0 (or supply an empty argument
%       in which case an initial population will be chosen randomly
```

```
%       between the lower and upper bounds VLB and VUB.  Use OPTIONS
%       to specify optional parameters such as population size and
%       maximum number of generations produced.
%
%            The  default  algorithm  uses  a  fixed  population  size,
OPTIONS(11)
%       and no generational overlap.  Three genetic operations:
%       reproduction, crossover, and mutation are performed.

%       Crossover in mating occurs with probability Pc=OPTIONS(12)
%       and the crossover index is randomly selected.  Each feature
%       of the offspring can mutate independently with probability
%       Pm=OPTIONS(13).  Default options are OPTIONS(11:13)=[30 1 0].
%       The default maximum generations OPTIONS(14) is 100.
%
%      X=GENETIC('FUN',X0,OPTIONS,VLB,VUB,BITS) allows the user to
%      define the number of BITS used to code non-binary parameters
%      as  binary  strings.   Note:  length  (BITS)  must  equal  length
(VLB).
%
%      X=GENETIC('FUN',X0,OPTIONS,VLB,VUB,BITS,P1,P2,...) allows up
%      to ten arguments, P1, P2, ... to be passed directly to FUN.
%      F=FUN(X,P1,P2,...).
%
%      [X,STATS,OPTIONS,BESTF,FGEN,LGEN]=GENETIC(<ARGS>)
%         STATS   - [max min mean std] for each generation
%         OPTIONS - options used
%         BESTF   - Fitness of individual X (i.e.: best fitness)
%         FGEN    - first generation population
%         LGEN    - last generation population
```

```
function [gen,lchrom,coarse,nround] = encode(x,vlb,vub,bits)
%ENCODE Converts from variable to binary representation.
%   [GEN,LCHROM,COARSE,nround] = ENCODE(X,VLB,VUB,BITS)
%       encodes non-binary variables of X to binary.  The variables
%       in the i'th column of X will be encoded by BITS(i) bits.  VLB
%         and  VUB  are  the  lower  and  upper  bounds  on  X.   GEN  is  the
binary
%       representation of these X.  LCHROM=SUM(BITS) is the length of
%       the binary chromosome.  COARSE(i) is the coarseness of the
%       i'th variable as determined by the variable ranges and
%       BITS(i).  ROUND contains the absolute indices of the
%       X which where rounded due to finite BIT length.
```

```
function [x,coarse] = decode(gen,vlb,vub,bits)
%DECODE Converts from binary to variable representation.
%   [X,COARSE] = DECODE(GEN,VLB,VUB,BITS) converts the binary
%       population GEN to variable representation.  Each individual
%       of GEN should have SUM(BITS).  Each individual binary string
%       encodes LENGTH(VLB)=LENGTH(VUB)=LENGTH(BITS) variables.
%       COARSE is the coarseness of the binary mapping and is also
```

```
%       of length LENGTH(VUB).
```

```
function [new_gen,selected] = reproduc(old_gen,fitness)
%REPRODUC selects individuals proportional to their fitness.
%   [NEW_GEN,SELECTED] = MATE(OLD_GEN,FITNESS) selects
%       individuals from OLD_GEN proportional to their FITNESS
%       NEW_GEN will have the same number of individuals as OLD_GEN.
%       SELECTED contains the indices (rows) of the selected
%       individuals (ie: NEW_GEN=OLD_GEN(SELECTED,:)).
```

```
function [new_gen,mating] = mate(old_gen)
%MATE   Randomly reorders (mates) OLD_GEN.
%   [NEW_GEN,MATING] = MATE(OLD_GEN) performs random reordering
%       on OLD_GEN.  NEW_GEN is the new reordering.  Individual in
%       row 1 is to be mated with individual in row 2, etc.  MATING
%       is the reordering vector (ie: new_gen=old_gen(mating,:)).
```

```
function [new_gen,sites] = xover(old_gen,Pc)
%XOVER  Creates a NEW_GEN from OLD_GEN using crossover.
%   [NEW_GEN,SITES] = XOVER(OLD_GEN,Pc) performs crossover
%       procreation on pairs of OLD_GEN with probability Pc.
%       Crossover SITES are chosen at random (re: there will be
%       half as many SITES as there are individuals.
```

```
function [new_gen,mutated] = mutate(old_gen,Pm)
%MUTATE Changes a gene of the OLD_GEN with probability Pm.
%   [NEW_GEN,MUTATED] = MUTATE(OLD_GEN,Pm) performs random
%       mutation on the population OLD_POP.  Each gene of each
%       individual of the population can mutate independently
%       with probability Pm.  Genes are assumed possess Boolean
%       Alleles.  MUTATED contains the indices of the mutated genes.
```

```
function fitness = IMfitness(x);
%IM fitness simulates the induction machine and as an output gives
%stator currents and rotor speed. It calls the main genetic function
%which further calls additional sub functions (reproduc, mate, xover
%etc.) It calculates the fitness at every step after calling genetic
%main function.

global fo Vs w wb WrCoeff TeCoeff we
global rs Xrrp D Xm Xss Xlrp Xls rs rrp indexx Vabcs tv wr_actual J P

indexx=0;

% Declare global variables
global fo Vs w wb WrCoeff TeCoeff Iabcs_actual
```

```
global rs Xrrp D Xm Xss Xlrp Xls rs rrp indexx Vabcs  Vqdos tv wr

% Declare parameters as variables to be optimized:

rs = x(1);
Xls = x(2);
Xm = x(3);
Xlrp = x(4);
rrp = x(5);
%J=x(6);


————————————————— Parameters for 3hp machine  —————————————————

P = 4; fo = 60;
Parms=[rs Xls Xm rrp Xlrp];
J=0.089;

% Extract actual (measured) data - stator voltages, stator currents
and %rotor speed

load numeric3hp.mat  % 3hp machine data

%load numeric2250hp.mat  %2250hp machine data
%load numeric50hp.mat    %50hp machine data
%load numeric7.5hp.mat   %7.5hp machine data


————————————————— Actual parameters for 3 hp machine  —————————————————
   %P = 4;  fo = 60;
   %Vs = 220/sr3;
   %rs = 0.435; Xls = 0.754; Xm = 26.13;  Xlrp = 0.754; rrp = 0.816;
   %J = 0.089; Pn=3*746.7;
   % Tf = 0.5;

————————————————— Actual Parameters for 2250 hp machine  —————————————————

   %P = 4; fo = 60;
   %Vs = 2300/sr3;
   %rs = 0.029; Xls = 0.226; Xm = 13.04; Xlrp = 0.226;  rrp = 0.022;

% Define flux linkages as the first six variables
% Define wr = rotor speed as the seventh variable
% Stator currents are output variables in the form (3.14)
% Stator voltages are the input variables

Xss  = Xls  + Xm;
Xrrp = Xlrp + Xm;
D = Xss*Xrrp - Xm^2;
wb = we;    % Choose synchronous speed (in rad/s) as the p.u. base.

WrCoeff = P/(2*J);
TeCoeff = .75*P*Xm/(D*wb);
```

```
C = 1/D*[ Xrrp        0         0       -Xm         0          0;
           0        Xrrp        0        0         -Xm         0;
           0          0       D/Xls      0          0          0;
          -Xm         0         0       Xss         0          0;
           0        -Xm         0        0         Xss         0;
           0          0         0        0          0       D/Xlrp];

FL0 = zeros(6,1);      % initialization for the first interval

[t, FL] = ode15s('flux',tv,FL0);
AAA2=FL;
Iqdos = C(1:3,:)*FL(:,1:6).';
Iqdor = C(4:6,:)*FL(:,1:6).';

Te = TeCoeff*(FL(:,1).*FL(:,5) - FL(:,4).*FL(:,2));     % (4.6-6)

% Now convert Iqdo back to Iabc

Iabcs = zeros(3, length(t));

th=w*t;

for pt = 1:length(t);
   th = th(pt);

   Ks_inv = [cos(th) sin(th) 1; cos(th-(2*pi/3)) sin(th-(2*pi/3)) 1;
...
             cos(th+(2*pi/3)) sin(th+(2*pi/3)) 1];
   Iabcs(:,pt) = Ks_inv * Iqdos(:,pt);

end

Iabcs_estimate=Iabcs;   % estimate currents which are outputs from the
%model obtained after simulating machine

%Error is defined as the difference between stator measured and
%estimated currents
%Speed is also included in some cases in performance index beside
%stator currents

%Error=sqrt(sum((Iabcs_actual(1,:)-
Iabcs_estimate(1,:)).^2+(Iabcs_actual(2,:)-
Iabcs_estimate(2,:)).^2+(Iabcs_actual(3,:)-Iabcs_estimate(3,:)).^2));

%Error=(sum((Iabcs_actual(1,:)-
Iabcs_estimate(1,:)).^2+(Iabcs_actual(2,:)-
Iabcs_estimate(2,:)).^2+(Iabcs_actual(3,:)-Iabcs_estimate(3,:)).^2));

Error=sum(abs(Iabcs_actual(1,:)-
Iabcs_estimate(1,:))+abs(Iabcs_actual(2,:)-
Iabcs_estimate(2,:))+abs(Iabcs_actual(3,:)-Iabcs_estimate(3,:)));

fitness=10e09/(Error);  % Gain is defined as 10e09 - However, this is
%subject to changes for different type of motors
```

```
fitness;
```

```
function Flux = flux(t,FL);

%This function calls ode15s (one of the differential solvers) to
obtain %solution for the fluxes Equation (3.10).

%Input data are in discrete form.

global indexx Vabcs tv

global fo Vs w wb WrCoeff TeCoeff Iabcs_actual
global rs Xrrp D Xm Xss Xlrp Xls rs rrp wr J P


we = 2*pi*fo; % Omega-e, the angular frequency of the synchronous
frame

w=we; % Work in the synchronous reference frame

th = w*t;

Ks = 2/3*[cos(th) cos(th-(2*pi/3)) cos(th+(2*pi/3)); ...
          sin(th) sin(th-(2*pi/3)) sin(th+(2*pi/3));  .5 .5 .5];

%Extract actual (measured) data - stator voltages, stator currents and
%rotor speed

load numeric3hp.mat  % 3hp machine data

%load numeric2250hp.mat  %2250hp machine data
%load numeric50hp.mat    %50hp machine data
%load numeric7.5hp.mat   %7.5hp machine data
t;
tv;
indexx=indexx+1;

index=find(tv>=t);
indic=min(index);

A=wr_actual';   %wr_actual = measured rotor speed

B=A./((2/P)*60/(2*pi));

if indic==1
    Vqdos_index=Vabcs(:,1);
    wr_index=B(1,:);
else
   t;
   indic;
```

```
%Stator voltages
m=(Vabcs(:,indic)-Vabcs(:,indic-1))./(tv(indic)-tv(indic-1));
d=Vabcs(:,indic)-m.*tv(indic);
Vqdos_index=Ks*(m.*t+d);


%Rotor Speed
n=(B(indic,:)-B(indic-1,:))./(tv(indic)-tv(indic-1));
c=B(indic,:)-n.*tv(indic);
wr_index=(n.*t+c);

end

Vqdor = [0;0;0];

A = -wb*[rs*Xrrp/D    w/wb      0    -rs*Xm/D       0          0;
         -w/wb     rs*Xrrp/D    0        0      -rs*Xm/D       0;
           0           0      rs/Xls     0          0          0;
         -rrp*Xm/D       0           0      rrp*Xss/D   (w-wr_index)/wb
0;
           0          -rrp*Xm/D      0     -(w-wr_index)/wb   rrp*Xss/D
0;
           0           0        0        0          0      rrp/Xlrp];

Flux = zeros(6,1);
Flux(1:6) = A*FL(1:6) + wb*[Vqdos_index; Vqdor];
Te = TeCoeff*(FL(1)*FL(5) - FL(4)*FL(2));
```

%Run GA

```
clear all;

global x Parms wr_actual
global fo Vs w wb  Ks we Iabcs_actual
global tv   Xrrp  D  Xm  Xss  Xlrp  Xls  rs  rrp  indexx  Vabcs   Vqdos  tv
wr_actual


bits = [10 10 10 10 10];  %Define the number of bits for GA

% Actual Parameters for 7.5 Hp machine from no load test, blocked
rotor % test in the Power Laboratory
% rs=0.2935;   Xls=0.2557;   Xm=13.5813;   Xlrp=0.5966;    rrp=0.2372;

%vlb = [0.1 0.1 10 0.2 0.1];
%vub = [0.8 1 20 1 0.8];


% Actual Parameters for 2250 hp machine - obtained from IEEE standard
%tests (reference)
%rs = 0.029; Xls = 0.226;  Xm = 13.04;   Xlrp = 0.226;    rrp = 0.022;

%vlb=[0.001 0.1 10 0.1 0.005];
```

```
%vub=[0.1 1 20 0.5 0.05];

% Actual Parameters for 3 hp machine - obtained from IEEE standard
%tests (reference)
%rs = 0.435;  Xls = 0.754;  Xm = 26.13;  Xlrp = 0.754; rrp = 0.816;
%J=0.089

vlb=[0.1 0.2 20 0.5 0.5];
vub=[1 1 30 1 1];

% Actual Parameters for 50 hp machine - obtained from IEEE standard
tests (reference)
% rs = 0.087;  Xls = 0.302;  Xm = 13.08;   Xlrp = 0.302;   rrp =
0.228;

%vlb=[0.05 0.1 10 0.1 0.1];
%vub=[0.1 1 20 1 0.5];


options = foptions([1 1e-4]);  % First:printing  second:Terminate
options(11)=30;  % Number of population - before it was 30
options(13) = 0.033;  %Mutation probability

options(12)=.70; %Crossover probability

options(14)=300; % Max number of gen

load numeric3hp.mat  % 3hp machine data

%load numeric2250hp.mat  %2250hp machine data
%load numeric50hp.mat    %50hp machine data
%load numeric7.5hp.mat   %7.5hp machine data

we = 2*pi*60; % Omega-e, the angular frequency of the synchronous
frame
w=we; % w
Vqdos = zeros(3, length(tv));     % Initialize to 0.
for pt = 1:length(tv);
   th = w*tv(pt);
      % If omega is not constant, theta = integral of omega
  Ks = 2/3*[cos(th) cos(th-(2*pi/3)) cos(th+(2*pi/3)); ...
            sin(th) sin(th-(2*pi/3)) sin(th+(2*pi/3));  .5 .5 .5];
  Vqdos(:,pt) = Ks*Vabcs(:,pt);
end


[x,stats,options,bf,fgen,lgen]=
genetic('IMfitness',[],options,vlb,vub,bits);

x % best fitness
```

References

[1] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer – Verglar 3rd edition, 1996.

[2] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison – Wesley Publishing Company, Inc. 1989.

[3] Steven R. Shaw. Numerical Methods for Identification of Induction Motor Parameters. Master's thesis, MIT, January 1997.

[4] Dennis J. Evangelista. Parameter Estimation of Induction Motors using PWM Inverters.Master's thesis, MIT, May 1999.

[5] A. E. Fitzgerald, C. Kingsley, S. D. Umans. *Electric Machinery*. McGraw-Hill, fourth edition, 1983.

[6] Paul C. Krause, Oleg Wasynczuk, Scott D. Sudhoff. *Analysis of Electric Machinery*. IEEE Press, 1995.

[7] Lawrence Davis. *Handbook of Genetic Algorithms*. International Thomson Computer Press, 1996.

[8] P. C. Krause, C. H. Thomas. Simulation of Symmetrical Induction Machinery. *IEEE Transactions on Power Apparatus and Systems*, November 1965.

[9] Richard R. Bishop, Gill G. Richards. Identifying Induction Machine Parameters Using a Genetic Optimization Algorithm. *IEEE Proceedings* – 1990 Southeastcon.

[10] P. Ju, E. Handschin, Z. N. Wei. Application of genetic algorithm to nonlinear dynamic modeling. 1995 Elsevier Science B. V.

[11] Christiaan Moons, Bart De Moor. Parameter Identification of Induction Motor Drives. *Automatica*, August 1995.

[12] Pragasen Pillay, Ray Nolan, Towhidul Haque. Application of Genetic Algorithms to M otor Parameter Determination for Transient Torque Calculations. *IEEE Transactions on Industry Applications*, October 1997.

[13] Sakari Palko. Structural Optimisation of an Induction Motor using a Genetic Algo rithm and a Finite Element Method. Doctoral's Dissertation, August 1996.

[14] Jussi Lahteenmaki. *Genetic algorithm in optimization of electric machines*. Helsinki University of Technology.

[15] Michael T. Wishart, Ronald G. Harley. Identification and Control of Induction Machines Using Arificial Neural Networks. *IEEE Transactions on Industry Applications*, June 1995.

[16] H. Razik, C. Defranoux, A. Rezzoug. Identification of Induction Motor using a Genetic Algorithm and a Quasi – Newton Algorithm. *CIEP Mexico*, October 2000.

[17] F. Alonge, F. D'Ippolito, G. Ferrante, F.M. Raimondi. Parameter Identification of induction motor model using genetic algorithms. *IEEE Proceedings*, 1999.

[18] F. Alonge, F. D'Ippolito, F.M. Raimondi. Parameter Identification of Induction Motors Least Squares vs. Genetic Algorithms. *Istituto di Automatica e Sistemistica, University of Palermo*, Italy, 1999.

[19] Pui Yan Chung, Melik Dolen, Robert D. Lorenz. Parameter Identification for Induction Machines by Continuous Genetic Algorithms. *ANNIE 2000 Conference*, November 2000.

[20] K.S. Huang, W. Kent, Q.H. Wu, D. R. Turner, Parameter Identification of an Induction Machine Using a Genetic Algorithms. Proceedings of the 1999 IEEE, August 1999.

[21] Phaderm Nangsue, Pragasen Pillay, Susan E. Conry. Evolutionary Algorithms for Induction Motor Parameter Determination.

[22] Reid Maust. Induction Motor Modeling Using a Genetic Algorithm. EE 330 Project, May 1999.

[23] Steven R. Shaw, Steven B. Leeb. Identification of Induction Motor Parameters from Transient Stator Current Measurements. *IEEE Transactions on Industrial Electronics,* February 1999.

[24] Seung-Ill Moon, Ali Keyhani, Srinivas Pillutla. Nonlinear Neural – Network Modeling of an Induction Machine. *IEEE Transactions on Control Systems Technology*, March 1999.

[25] H.B. Karayaka, A. Keyhani. Induction Machine Parameter Tracking from Test Data via PWM Inverters. *IEEE Industry Applications Society*, October 1997.

[26] JA de Kock, FS van der Merwe, HJ Vermeulen. Induction Motor Parameter EstimationThrough an Output Error Technique. *IEEE Transactions on Energy Conversion*, March 1994.

[27] Simon Hart, Abolfazl Ranjbar, B.E. Mulhall. Practical Combined Parameter Identification and State Estimation of Machines. *Proceedings volume from the 8th IFAC Symposium*, UK, September 2000.

[28] Robin Biesbroek. GA Tutorial. http://www.estec.esa.nl/outreach/gatutor/Default.htm.

[29] MATLAB, High Performance Computation and Visualization Software, *The Matworks Inc*., 2001.

[30] Genetic Algorithms. Implementation based on GAs from David E. Goldberg. Andrew F Potvin. *The MathWorks, Inc*. January 1994.

[31] Basics on GAs. http://www.d.umn.edu/ece/lis/GA/ga_1_basics.html.

[32] LabVIEW Data Acquisition Basics Manual. *National Instruments*. January 1998.

[33] LabVIEW User Manual. *National Instruments*. January 1998.

[34] DAQ PCI E Series User Manual. *National Instruments*. July 1997.

[35] Kourosh Seghisigarchi. DAQ User Manual. WVU 1999.

[36] IEEE Standard Test Procedure for Polyphase Induction Motors and Generators. *Electric Machines Committee of the IEEE Power Engineering Society, USA*. IEEE Std 112-1996. INSPEC Accession Number: 5621352.

[37] Yu-hua-Wang, Birdwell-JD. Dynamic identification of the model parameters for an induction motor. *Conference Proceedings of IEEE SOUTHEASTCON*, 1982.

[38] National Instrument and LAB View Software. http://www.ni.com/.