Application of Genetic Algorithms to Motor Parameter Determination for Transient Torque Calculations

Pragasen Pillay, *Senior Member, IEEE,* Ray Nolan, and Towhidul Haque

***Abstract*—This paper applies genetic algorithms to the problem of induction motor parameter determination. Generally available manufacturers' published data like starting torque, breakdown torque, full-load torque, full-load power factor, etc., are used to determine the motor parameters for subsequent use in studying machine transients. Results from several versions of the genetic algorithm are presented, as well as a comparison with the Newton–Raphson method.**

***Index Terms*— Genetic algorithms, motor parameter determination, Newton–Raphson.**

I. Introduction

THE problem of induction motor parameter estimation and its use in the prediction of motor performance has been addressed by several researchers [1]–[8]. Deep-bar machines were considered in [1], while the estimation of motor parameters from standstill tests were considered in [2] and [3]. Particular attention was paid to leakage reactances in [4] and [5], while in [6], the extended Kalman filter was used to address the problem of the rotor time constant for vector-controlled drives. Other techniques were used for motor parameter estimation in [7] and [8]. The one common denominator in these papers is the high accuracy demanded in the parameter determination, especially in vector-controlled drives.

In the world of relaying and power system protection, however, extreme accuracy of the order attempted by the researchers above is not needed [9], [10]. The selection of breakers, current ratings of current transformers, etc., are not done to 1% accuracy; typically, 10% or 15% is sufficient. At the same time, the input data available for the determination of the motor parameters is generic at best. That is, the individual designer's data is typically not available (especially for an old motor in a plant). Tests to determine the parameters of such motors are out of the question for machines running continuously, which happens quite often in the petrochemical industry, for example.

This paper addresses the determination of suitable motor parameters for system-type studies such as these, where the input data available may be little more than what is available on the nameplate, like starting torque, breakdown torque, full-load torque, full-load power factor, full-load efficiency, etc. It is desirable to be able to extract the motor parameters from such data, so that torque transients can be calculated, for example, during autoreclose operation of the distribution breaker by the power company. The Newton–Raphson method has been previously used, but with convergence problems relating to the initial starting points and the requirement for iteration [11]. In this paper, two different techniques, the Newton–Raphson and genetic algorithms, are used to extract the motor parameters from the readily available and, hence, generic available data. Several different induction machines are tested, and the results are compared.

Paper IPCSD 97–48, presented at the 1994 Industry Applications Society Annual Meeting, Denver, CO, October 2–7, and approved for publication in the IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS by the Electric Machines Committee of the IEEE Industry Applications Society. This work was supported by the Electric Power Research Institute, Entergy Services, Louisiana State University, and Mobil Oil. Manuscript released for publication May 27, 1997.

P. Pillay was with the Electrical Engineering Department, University of New Orleans, New Orleans, LA 70148 USA. He is now with the Electrical and Computer Engineering Department, Clarkson University, Potsdam, NY 13699-5720 USA.

R. Nolan was with the Electrical Engineering Department, University of New Orleans, New Orleans, LA 70148 USA. He is now with Marrero, Couvillon & Associates, Metairie, LA 70002 USA.

T. Haque was with the Electrical Engineering Department, University of New Orleans, New Orleans, LA USA. He is now with Denro, Inc., Gaithersburg, MD 20877 USA.

Publisher Item Identifier S 0093-9994(97)07043-6.

II. Newton–Raphson Optimization Using Quattro Pro

Quattro Pro uses the Newton–Raphson method to solve nonlinear equations that may encompass several variables and constraints. The equivalent circuit (EC) parameters of an induction machine, which include stator and rotor resistances, and stator, rotor, and magnetizing reactances, can be obtained from Quattro Pro using its Newton–Raphson-based optimizer function. The Quattro Pro spreadsheet can be set up to include the torque and power factor equations, an initial estimate for each parameter, and relevant nameplate and performance data. The relevant performance data consist of full-load, locked-rotor, and breakdown torque values, full-load power factor, full-load slip, and supply voltage. Quattro Pro begins by using the Newton–Raphson optimizer to adjust each parameter and recalculate the spreadsheet. Based on the new results, the optimizer continues to make adjustments until a solution is reached that meets all of the requirements. The optimizer's recommended solutions appear in the designated cells, but the solutions vary depending on the initial estimates of the EC parameters. In general, the more realistic the starting values are, the closer the results are to the correct optimal solution.

The major drawback to the Newton–Raphson method is that its success depends on the selection of good initial estimates. Although the optimization process may only take a few minutes, a considerable amount of time and effort can be spent selecting the initial estimates, which require familiarity with the particular machine size and parameters. Even when the initial solutions appear reasonable, the optimizer still may not converge to the correct solution.

III. Genetic Algorithms

A. Introduction

The genetic algorithm is another method which may be used to solve a system of nonlinear equations. The genetic algorithm uses objective functions based on some performance criterion to calculate an error. However, the genetic algorithm is based on natural selection using random numbers and does not require a good initial estimate. That is, solutions to complex problems could evolve from poor initial estimates in a game of survival of the fittest. Genetic algorithms manipulate strings of binary digits and measure each string's strength with a fitness value. The stronger strings advance and mate with other strong strings to produce offspring. Eventually, one string emerges as the best. One of the most important advantages of the genetic algorithm over the Newton–Raphson technique is that it is able to find the global minimum, instead of a local minimum, and that the initial estimate need not be close to the actual values. Another advantage is that it does not require the use of the derivative of the function, which is not always easily obtainable or may not even exist, for example, when dealing with real measurements involving noisy data.

B. The Main Operators

The mechanics of the genetic algorithm are elementary, involving nothing more than copying strings, random number generation, and swapping partial strings [12]. A simple genetic algorithm that produces good results in many practical problems is composed of the following three operators:

1) reproduction;
2) crossover;
3) mutation.

Reproduction is a process in which individual strings are selected according to their fitness. The fitness is determined by calculating how well each string fits an objective function. Copying strings according to their fitness value implies that strings that fit the objective function well have a higher probability of contributing one or more offspring in the next generation. This process of reproduction is, of course, an artificial version of natural selection. Here, the objective function is the final arbiter of the string creature's life or death.

Stochastic sampling with replacement is the name given to a simple reproduction scheme. This scheme is based on placing the string probabilities on a weighted roulette wheel and spinning the wheel to select a string. The probabilities on the roulette wheel are determined by the string's fitness as a percentage of the total population fitness. The roulette wheel selection scheme utilizes random numbers to simulate a spin of the wheel. Once a string is selected by the reproduction operator, the string is copied into a mating pool and waits to be selected for further genetic operator action. The roulette wheel scheme does not guarantee that the fittest strings will be selected, although their probability for selection is high. Therefore, this method may not produce the best results, especially for problems with small populations.

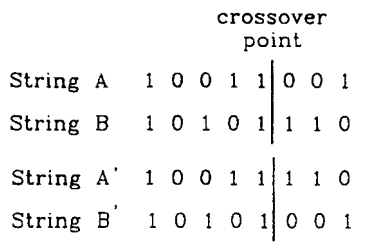

Fig. 1. The crossover operator.

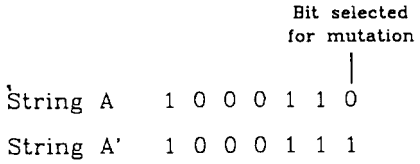

Fig. 2. The mutation operator.

Crossover is a two-step process that involves mating and swapping of partial strings. Each time the crossover operator takes action, two randomly selected strings from the mating pool are mated. Then, in the case of simple crossover, a position along one string is selected at random, and all binary digits following the position are swapped with the second string. The result is two entirely new strings that move on to the next generation. This can be more clearly understood by the following example, in which string 1 and string 2 have already been chosen to mate, as shown in Fig. 1.

Mutation follows crossover and protects against the loss of useful genetic information (1's and 0's). The operator works by randomly selecting one string and one bit location and changing that string's bit from a 1 to a 0 or vice versa, as shown in Fig. 2. The probability for mutation to occur is usually very small, roughly one mutation per 1000 bit transfers.

The three genetic operators, reproduction, crossover, and mutation, provide an effective search technique using natural selection and random number generation. Advanced operators, such as dominance, inversion, and segregation exist, but are generally not essential for good results to many problems. In some cases, the advanced operators can degrade the performance of the genetic algorithm.

C. Implementation of the Genetic Algorithm

The genetic algorithm can be used to calculate the EC parameters of an induction machine, as shown in Fig. 3. The locked-rotor, breakdown, and full-load torque equations form a multiobjective optimization problem, where each equation is a function of three or more machine parameters. The three

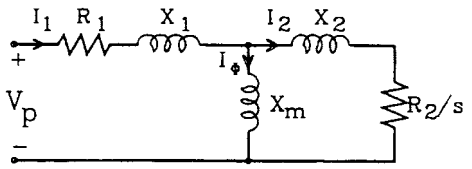

Fig. 3. Induction motor EC.

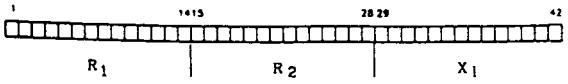

Fig. 4. The genetic algorithm string.

torque functions can be written as follows:

$$F1(R_1,R_2,X_l)=\frac{V^2\dfrac{R_2}{s}}{w_s\left[\left(R_1+\dfrac{R_2}{s}\right)^2+X_l^2\right]}-T_{fl} \quad (3.1)$$

$$F2(R_1,R_2,X_l)=\frac{V^2R_2}{w_s[(R_1+R_2)^2+X_l^2]}-T_{lr} \quad (3.2)$$

$$F3(R_1,X_l)=\frac{V^2}{2w_s[R_1+\sqrt{R_1^2+X_l^2}]}-T_{bd} \quad (3.3)$$

where $F1$ is the error in the full-load torque, $F2$ is the error in the locked-rotor torque, $F3$ is the error in the breakdown torque, R_2 is the rotor resistance, R_1 is the stator resistance, X_2 is the rotor reactance, and X_1 is the stator reactance, $X_l = X_1 + X_2$.

For simplicity, the stator and rotor leakage reactances are combined into one leakage reactance (X_l). The stator and rotor reactances can be extracted after the optimization by knowing the design class of the machine. The magnetizing reactance (X_m) can be calculated using the full-load power factor equation after R_1, R_2, and X_l have been calculated, using the genetic algorithm.

Each parameter is coded as a 14-bit unsigned binary number, and together they form one 42-bit string, as shown in Fig. 4. The maximum value each parameter can have, based on an accuracy of three decimal places, is 16.384 Ω.

In this case, the error function is chosen as the sum of the squares of the torque error functions, while the fitness function is the inverse of the error. The aim of the genetic algorithm is to minimize the error or to maximize the fitness:

$$\varepsilon = F1(\cdot)^2 + F2(\cdot)^2 + F3(\cdot)^2 \quad (3.4)$$

$$\text{Fitness} = \frac{1}{\varepsilon}. \quad (3.5)$$

IV. Results

A. Steady-State Parameter and Torque Results

Three induction motors (Table I) with known EC parameters were used to test four different versions of the genetic algorithm. Each version uses the same population size, random number generator, string size, objective functions, and fitness function, yet each is slightly different in its approach. A description of each version follows.

TABLE I
Actual Machine Parameters

Hp	Rpm	Volt	R_1	R_2	X_1	X_m
5	1750	230	0.434	0.303	2.511	24.61
50	1705	460	0.087	0.228	0.604	13.08
500	1773	2300	0.262	0.187	2.412	54.02

Version 1 (V1) uses stochastic sampling with replacement (weighted roulette wheel), as described earlier for reproduction. Simple crossover and mutation are also used, that is, one randomly selected crossover point and one bit change per thousand bit transfers for each string. This version is the simple genetic algorithm.

Version 2 (V2) is identical to V1, except that deterministic sampling is used, instead of stochastic sampling, for its reproduction scheme. The deterministic sampling scheme calculates the probabilities of selection as usual, i.e., the string fitness divided by the total fitness. Then, each string is assigned an expected number based on its probability of selection. The actual number of times a string is copied into the mating pool is found from the integer part of the expected number. If additional strings are needed to fill the new population, then the fractional parts of the expected number are sorted, and the strings are selected from the top of the sorted list. This selection scheme has proved superior to straight roulette-wheel selection, since it guarantees that fit strings will be copied into the mating pool.

Version 3 (V3) uses the deterministic sampling scheme with two-point crossover. The two-point crossover operator swaps all binary digits between two randomly selected points along the string.

Version 4 (V4) uses the deterministic sampling selection scheme with a crossover operator for each parameter. This means that there is one randomly selected crossover point for each parameter, or three crossover points along the entire string. This algorithm is superior to the other three, since it produces consistently good results.

The results of each version of the genetic algorithm are given in Tables II–V. Comparisons can be made with Table I, which shows the actual EC parameters for each induction machine. In addition, results from Quattro Pro's Newton–Raphson search routine is provided.

The results in using V1 of the genetic algorithm are shown in Table II. The estimated parameters are compared against the actual parameters and the errors calculated. Considerable errors are produced for some parameters, for example, 85% error in R_1 and even larger errors in R_2. This version would be unacceptable. Table III has the parameter results using V2. Although the parameter errors are not as large as for V1, they are still considerable. V3 does not produce substantially better results than V2, as shown in the parameter errors of Table IV. V4 has the best results with acceptable errors in R_2, X_l, and X_m, as shown in Table V. Larger errors were produced in R_1 for small and large motors. However, Table VI shows that errors in R_1 do not affect the torque calculations

TABLE II
RESULTS OF GENETIC ALGORITHM V1

	Motor 1 5 hp	Motor 2 50 hp	Motor 3 500 hp
R_1 (est)	0.063	0.083	0.666
R_1 (act)	0.434	0.087	0.262
% error	85.48	4.59	154.19
R_2 (est)	0.336	1.714	0.180
R_2 (act)	0.303	0.228	0.187
% error	10.89	651.75	3.74
X_1 (est)	2.690	0.642	2.046
X_1 (act)	2.511	0.604	2.412
% error	7.13	6.29	15.17
X_m (est)	28.44	71.77	44.31
X_m (act)	24.61	13.08	54.02
% error	15.56	448.70	17.97

TABLE III
RESULTS OF GENETIC ALGORITHM V2

	Motor 1 5 hp	Motor 2 50 hp	Motor 3 500 hp
R_1 (est)	0.707	0.245	0.334
R_1 (act)	0.434	0.087	0.262
% error	62.90	181.61	27.48
R_2 (est)	0.286	0.184	0.206
R_2 (act)	0.303	0.228	0.187
% error	5.61	19.29	10.16
X_1 (est)	2.038	0.440	2.438
X_1 (act)	2.511	0.604	2.412
% error	18.84	27.15	1.08
X_m (est)	20.39	9.39	55.18
X_m (act)	24.61	13.08	54.02
% error	17.15	28.21	2.15

TABLE IV
RESULTS OF GENETIC ALGORITHM V3

	Motor 1 5 hp	Motor 2 50 hp	Motor 3 500 hp
R_1 (est)	0.194	0.305	0.297
R_1 (act)	0.434	0.087	0.262
% error	55.29	250.57	13.36
R_2 (est)	0.370	0.163	0.179
R_2 (act)	0.303	0.228	0.187
% error	22.11	28.51	4.27
X_1 (est)	2.529	0.332	2.496
X_1 (act)	2.511	0.604	2.412
% error	0.717	45.03	3.48
X_m (est)	29.29	7.81	55.05
X_m (act)	24.61	13.08	54.02
% error	19.02	40.29	1.91

TABLE V
RESULTS OF GENETIC ALGORITHM V4

	Motor 1 5 hp	Motor 2 50 hp	Motor 3 500 hp
R_1 (est)	0.367	0.087	0.325
R_1 (act)	0.434	0.087	0.262
% error	15.44	0.00	24.04
R_2 (est)	0.314	0.239	0.191
R_2 (act)	0.303	0.228	0.187
% error	3.63	4.82	2.14
X_1 (est)	2.386	0.641	2.470
X_1 (act)	2.511	0.604	2.412
% error	4.98	6.13	2.40
X_m (est)	25.39	13.73	54.72
X_m (act)	24.61	13.08	54.02
% error	3.17	4.97	1.29

TABLE VI
TORQUE ERRORS FOR GENETIC ALGORITHM V4

	Motor 1 5 hp	Motor 2 50 hp	Motor 3 500 hp
T_{fl} (est)	22.36	234.17	2023.23
T_{fl} (act)	22.35	234.55	1997.9
% error	0.045	0.162	1.267
T_{lr} (est)	14.31	519.10	841.85
T_{lr} (act)	14.3	529.708	835.57
% error	0.069	2.00	0.752
T_{bd} (est)	50.45	765.39	4982.38
T_{bd} (act)	50.14	773.987	5017.68
% error	0.618	1.111	0.704

significantly. For example, while there were 15% and 24% errors in R_1, for the 5-hp and 500-hp motors, the maximum error produced in the estimation of any torque is 2%. This is not unexpected, since the error function was defined to minimize the torques, not the electrical parameters. Tables VII and VIII show that acceptable results are obtainable using Newton–Raphson techniques, provided good initial estimates of parameters are used. However, Table IX shows that a slight change in the initial estimate of a parameter can cause the Newton–Raphson to converge to an entirely wrong solution, as shown for the leakage and magnetizing reactances. The genetic algorithm is more robust in this regard.

The performance of the error function when each version of the genetic algorithm is used is compared in Fig. 5. The results show that all versions eventually converge, thus producing low errors in the torques, but not necessarily low errors in the parameters. V4, however, converges fastest with acceptable errors in the parameters, as well as the torques. Fig. 6 shows the convergence of machine parameters using the Newton–Raphson method. Two cases are used to test the convergence of the Newton–Raphson method. In case 1, the initial guess of machine parameters was good, and the optimizer converged to the correct machine parameters. In case 2, the parameter Xl was changed from 1.05 to 1.00, while all other parameters remained the same, and the optimizer failed to converge.

B. Transient Torque Results

In order to demonstrate the accuracy during transient operation, the parameters generated by the genetic algorithm are used to predict the motor back-EMF waveforms for a 5-hp induction motor during a power outage. V4 of the genetic algorithm is used and compared against the results using the known original parameters, as shown in Figs. 7 and 8. The

TABLE VII
RESULTS OF NEWTON–RAPHSON SEARCH METHOD

	Motor 1 5 hp	Motor 2 50 hp	Motor 3 500 hp
R_1 (est)	0.392	0.087	0.262
R_1 (act)	0.434	0.087	0.262
% error	9.67	0.00	0.00
R_2 (est)	0.313	0.238	0.195
R_2 (act)	0.303	0.228	0.187
% error	3.30	4.39	4.28
X_1 (est)	2.375	0.631	2.521
X_1 (act)	2.511	0.604	2.412
% error	5.42	4.47	4.52
X_m (est)	24.24	13.73	56.09
X_m (act)	24.61	13.08	54.02
% error	1.50	5.35	3.83

TABLE VIII
TORQUE ERRORS FOR NEWTON-RAPHSON METHOD

	Motor 1 5 hp	Motor 2 50 hp	Motor 3 500 hp
T_{fl} (est)	22.33	235.16	2001.95
T_{fl} (act)	22.35	234.55	1997.9
% error	0.089	0.260	0.203
T_{lr} (est)	14.31	530.32	833.66
T_{lr} (act)	14.3	529.708	835.57
% error	0.069	0.116	0.229
T_{bd} (est)	50.13	775.27	5017.5
T_{bd} (act)	50.14	773.987	5017.68
% error	0.019	0.166	0.004

TABLE IX
DIVERGENCE OF NEWTON-RAPHSON METHOD

	Actual Machine Parameter	Case 1 Starting Parameter	Case 1 Final Parameter	Case 2 Starting Parameter	Case 2 Final Parameter
R_1	0.434	0.10	0.391	0.10	1.400
R_2	0.303	0.10	0.312	0.10	15.590
X_1	2.511	1.05	2.370	1.00	0.000
X_m	24.610	13.00	24.240	13.00	0.000

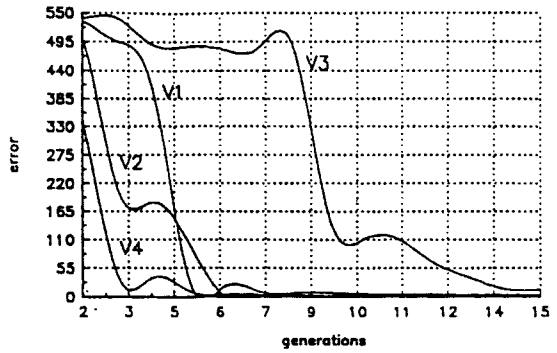

Fig. 5. Convergence of parameters using genetic algorithms.

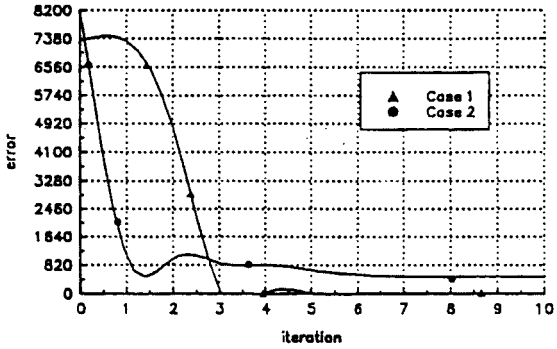

Fig. 6. Convergence of parameters using Newton–Raphson.

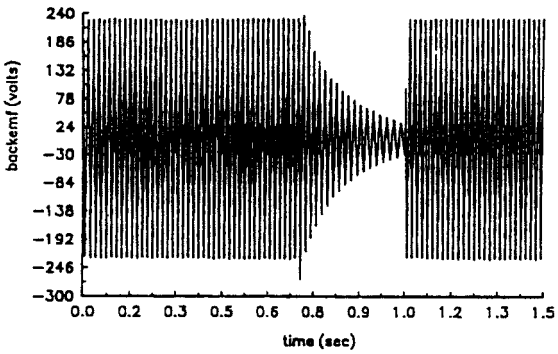

Fig. 7. Back EMF using actual parameters.

waveforms were generated using EMTP [13], [14], and it is clear that there is little difference between the waveforms.

While the back-EMF waveforms are virtually identical, the torque–speed curves are slightly dissimilar, mainly due to the differences in the rotor resistance. This is apparent in Figs. 9 and 10 for the genetic-generated parameters and the actual parameters, respectively.

C. Effect of Population Size and Bit Length on the Algorithm Performance

While 14 bits were used for each parameter and a population size of 250 strings was used in this paper, it is of interest to examine the effect of different bit lengths and population numbers on the ability of the algorithm to converge. The results are presented in Fig. 11, which shows a bit-length variation from 8 to 18 and the number of string and the population size from 100 to 500 for V4. The number of generations is basically an indication of the time to converge. If the number of bits is less than ten, then poor results are obtained, regardless of the population size. If the population is less than 150 strings, then poor results are obtained, regardless of the bit length. Also, if the population is too large, greater than 450 strings, for example, the performance deteriorates, regardless of bit size, indicating that, perhaps, other advanced operators like dominance and segregation may have to be used to procure reasonable results. The graph shows that the bit sizes between 10–18 and population sizes between 150–400 would give consistently good results. A smaller bit size and population size has advantages in computer space requirements and processing time.

D. Manufacturer Performance Data

Different manufacturers may calculate the performance data of a machine using a slightly different method. In addition,

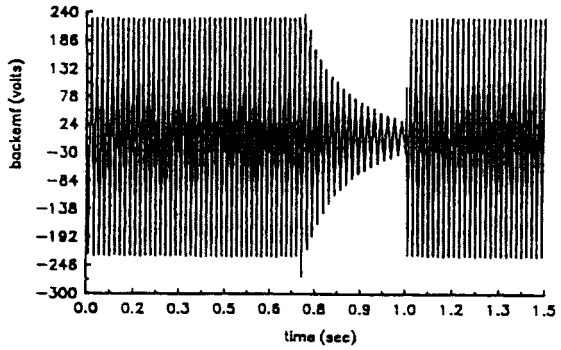

Fig. 8. Back EMF using genetic algorithm V4 parameters.

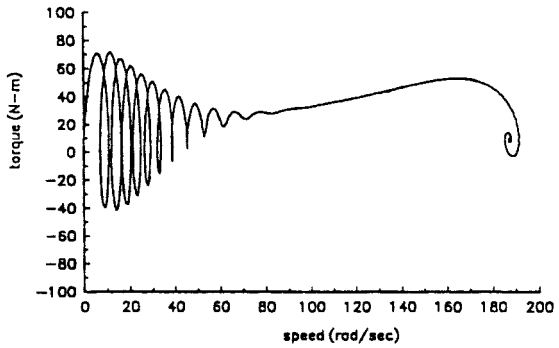

Fig. 9. Torque–speed curve using actual parameters.

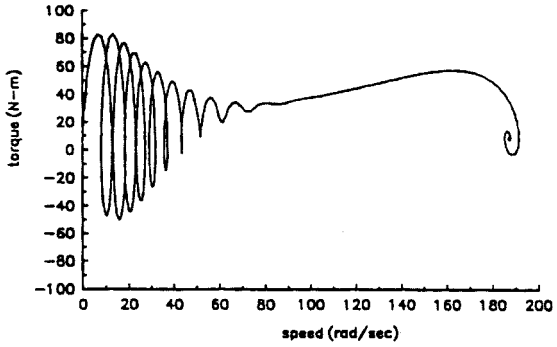

Fig. 10. Torque–speed curve using genetic algorithm V4 parameters.

the designs of different manufacturers can lead to different errors from using the five-parameter model used in this paper. Thus, the performance of the genetic algorithm would differ for different manufacturers, resulting in different errors in full-load, locked-rotor, and breakdown torque. This section examines these torque errors when using manufacturers' data of starting, full-load, and breakdown torques. Thus, no prior knowledge of the parameters was available. This section differs from Section IV-A, in that known parameters were used there to calculate the starting, full-load, and breakdown torques using the induction motor EC. Thus, the equivalent circuit was assumed to be accurate, with the same parameters being applied to starting, as well as running, conditions.

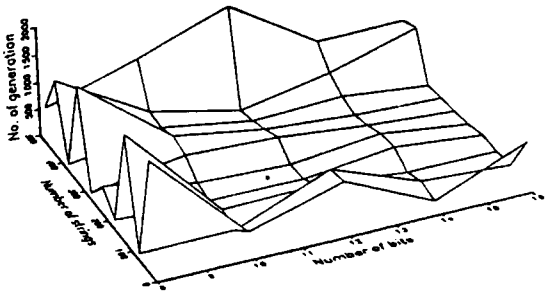

Fig. 11. Effect of bit length and population size on generation number.

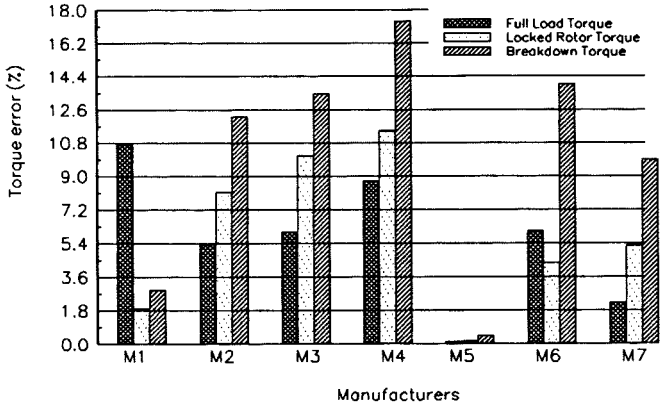

Fig. 12. Genetic performance for 5-hp induction machine.

The performance data from seven different manufacturers was gathered from MotorMaster, a package developed by the Washington State Energy Office to collect motor manufacturers' data. This was used as input for genetic algorithm V4. The parameters for the simple EC were calculated and the torques recalculated and the errors determined. Now, the errors are larger, as shown in Fig. 12 for a commercial 5-hp motor from seven different manufacturers, as a result of inaccuracies in the model (constant parameters with five variables). However, in spite of the simple model, manufacturers $M1, M3, M4, M5$, and $M7$ all produce acceptable results (around 10% error or less), while $M2$ and $M6$ torque calculations have larger errors. This indicates that either a deep-bar model or a five-parameter model with variable parameters must be used to reduce the errors in predicting the torques for these two manufacturers.

V. Deep Bar Induction Motor Model and Results

A. Deep-Bar Model

The genetic algorithm worked well when the single-cage model was accurate. That is, if the single-cage model with known parameters is used to calculate the starting, breakdown and full-load torques and, if these torques are fed into the genetic algorithm, the algorithm is capable of finding accurate single-cage parameters that allow the recalculation of the original torques. The algorithm performs less well when the manufacturer's torque data and a single-cage rotor is assumed. An obvious extension to the above work is to use a deep-bar machine model to allow for parameter variations at different slips.

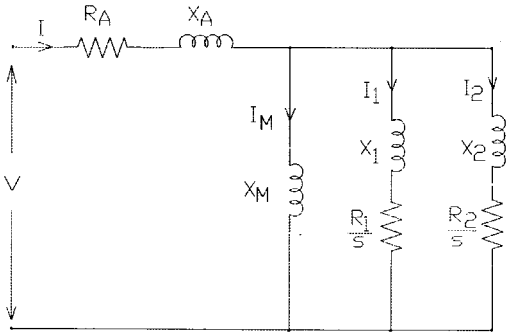

Fig. 13. Deep-bar induction motor EC.

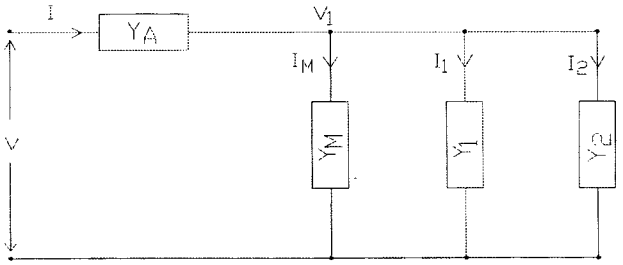

Fig. 14. EC admittance.

A deep-bar model of an induction motor has two separate cages, called the starting and running cage, resulting in seven electrical parameters. Fig. 13 shows the single-phase EC of a deep-bar induction motor, whereas Fig. 14 shows the same with admittance values indicated for each branch. When using a deep-bar model, all seven electrical parameters, stator resistance R_a, running cage resistance R_1, starting cage resistance R_2, stator winding leakage reactance X_s, running cage leakage reactance X_1, starting cage leakage reactance X_2, and magnetizing reactance X_m can be determined simultaneously using full-load torque, locked-rotor torque, breakdown torque, power factor, and full-load current. Motor performance data are used as inputs to determine the electrical parameters. These calculated parameters are then compared to the actual parameters to determine the errors.

The relevant equation used for calculating the deep-bar parameters are as follows:

$$f1''(R_A, X_A, R_1, X_1, R_2, X_2, X_M) = \frac{(T|_{s=\text{rated speed}} - T_{fl})100}{T_{fl}} \tag{5.1}$$

$$f2''(R_A, X_A, R_1, X_1, R_2, X_2, X_M) = \frac{(T|_{s=1} - T_{lr})100}{T_{lr}} \tag{5.2}$$

$$f3''(R_A, X_A, R_1, X_1, R_2, X_2, X_M) = \frac{(T|_{s=\max} - T_{bd})100}{T_{bd}} \tag{5.3}$$

$$f4''(R_A, X_A, R_1, X_1, R_2, X_2, X_M) = \frac{(\cos\theta - pf)100}{pf} \tag{5.4}$$

$$f5''(R_A, X_A, R_1, X_1, R_2, X_2, X_M) = \frac{(I|_{\text{rated speed}} - i_{fl})100}{i_{fl}} \tag{5.5}$$

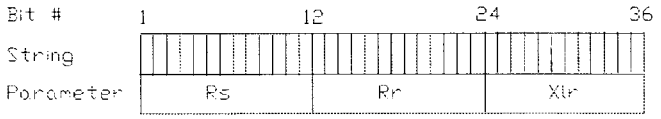

Fig. 15. The genetic algorithm string.

where full-load torque T_{fl}, blocked-rotor torque T_{lr}, breakdown torque T_{bd}, full-load current i_{fl}, and power factor pf are the performance data used in the program as input from the user.

Each parameter is coded as a 14-bit unsigned binary number, and together they form one 42-bit string for a single cage, as shown in Fig. 15, and a 98-bit string for a deep-bar model. The maximum value each parameter can have, based on an accuracy of three decimal places, is 16.384 Ω.

For the single-cage motor, the error function is chosen as the sum of the squares of the torque error functions, while the fitness function is the inverse of the error. The aim of the genetic algorithm is to minimize the error or to maximize the fitness, as shown in (3.4) and (3.5).

Similarly, for the deep-bar model, the error and fitness functions are given by

$$\varepsilon' = (f1'')^2 + (f2'')^2 + (f3'')^2 + (f4'')^2 + (f5'')^2 \tag{5.6}$$

$$\text{Fitness} = \frac{1}{\varepsilon'}. \tag{5.7}$$

B. Breakdown Torque For Deep-Bar Model

An analytical expression for breakdown torque T_{bd} cannot be easily obtained from differentiation of the torque expression. When $dT/ds = 0$ is solved for slip s, a polynomial equation, given in (3.13), of degree six is found:

$$Ms^6 + Ns^4 + Qs^2 + R = 0. \tag{5.8}$$

All the coefficients are in terms of electrical parameters and, hence, they are all constant for a particular motor.

In this section, two different approaches to determine breakdown torque are proposed and implemented. It is known that if a torque-versus-slip relationship of an induction motor is plotted, the breakdown torque is obtained at $s_{\max}$. Hence, one method of solution uses iteration within the main program to calculate the different values of torque as slip is varied over a range where $s_{\max}$ is known to occur. Since the magnitude of the breakdown torque is of main concern and not the slip at which breakdown torque occurs, this method gives a reasonably good result in calculating breakdown torque for the deep-bar induction motor for a given set of parameters obtained from the genetic algorithm. This is called the iterative loop method.

Another simplified approach for determining the breakdown torque for the deep-bar model can be used. The deep-bar model has two cages, a starting cage and a running cage. It has been found that the starting cage contributes a negligible amount of torque during the running condition. Since breakdown torque occurs near the high-speed running region, the starting cage can be neglected in calculating the breakdown torque. With

TABLE X
COMPARISON OF DIFFERENT METHODS OF DETERMINING T_{bd}

Motors	Methods	Calcul a-ted T_{bd} Nm	Actual T_{bd} Nm	Slip	%Error in T_{bd}
4700 HP,6.6KV	Iterative Loop	31943	31982	0.034	0.12
2970 rpm,50Hz	Simplifi-ed Model	30100	31982	0.033	5.6
5 HP,1745 rpm	Iterative Loop	50.93	50.57	0.136	0.71
230 v, 60 Hz	Simplifi-ed Model	49.73	50.57	0.135	1.66

TABLE XI
COMPARISON OF PERFORMANCE USING TWO DIFFERENT METHODS FOR CALCULATING BREAKDOWN TORQUE

Motors	Methods	%Error T_{fl}	%Error T_{lr}	%Error T_{bd}
4700 HP,6.6KV	Iterative Loop	1.8	2.0	3.0
2970 rpm,50Hz	Simplifi-ed Model	2.6	3.1	4.8
5 HP,1745 rpm	Iterative Loop	1.1	0.9	1.3
230 v, 60 Hz	Simplifi-ed Model	1.7	2.7	3.8

this simplification, the seven-parameter or deep-bar model reduces to a five-parameter model when determining the breakdown torque. This is called the simplified model method.

A comparison of different methods for determining the breakdown torque for the deep-bar motor is given in Table X. For small motors, the simplified five-parameter model gives almost the same result for breakdown torque as the seven-parameter model. The iterative loop method produces the most accurate result. In both methods, the slip at which breakdown torque occurs is approximately equal.

Table XI shows the comparison in the performance of genetic algorithms using the iterative loop and simplified model methods in determining parameters for the deep-bar model. For both motors, the simplified model method produces higher errors in full-load torque T_{fl}, blocked-rotor torque T_{lr}, and breakdown torque T_{bd} than those produced by the iterative loop method. However, all the errors in torque in the simplified model method are within an acceptable range.

C. Manufacturer Performance Data

This section examines the ability of the genetic algorithm to determine motor parameters for the calculation of torque from the readily available manufacturer's data. Thus, no prior knowledge of the parameters was available. This section differs from Section V-B, in that known parameters were used there to calculate the starting torque, locked-rotor torque, and breakdown torque of the motor using single-cage and deep-bar EC models.

The performance data of six different-sized motors from seven different manufacturers were gathered from MotorMaster, a package developed by the Washington State Energy Office to collect motor manufacturers' data. Out of these six motors, initially, the performance data of the 5-, 50-, and 100-hp motors were used as input to the genetic algorithm

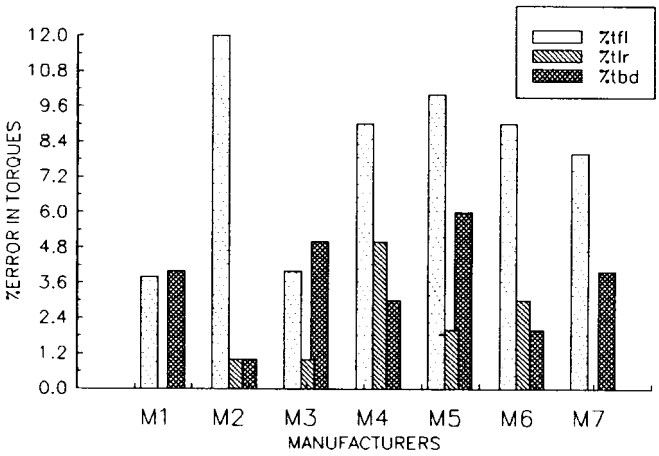

Fig. 16. Genetic performance using single-cage model for 5-hp motor.

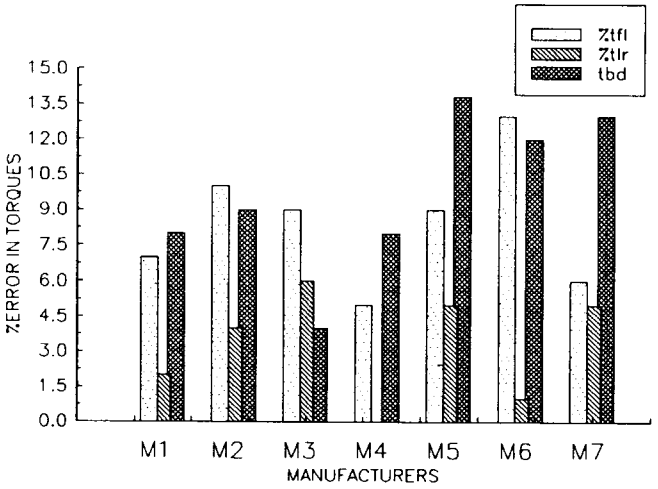

Fig. 17. Genetic performance using single-cage model for 50-hp motor.

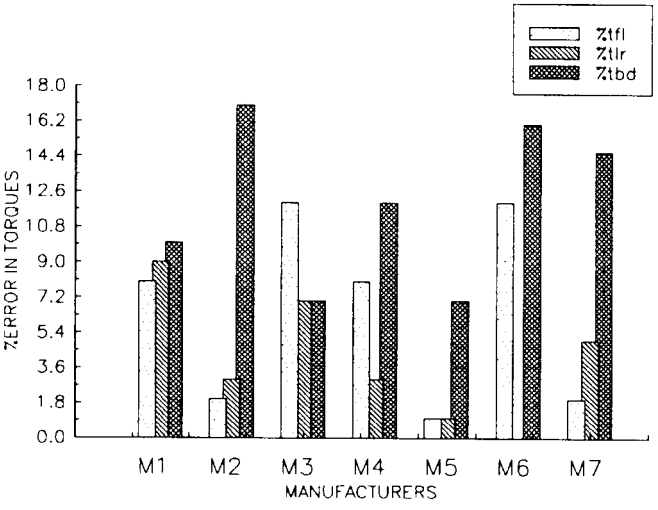

Fig. 18. Genetic performance using single-cage model for 100-hp motor.

using the single-cage model and deep-bar model. The parameters for the single-cage and deep-bar EC were calculated and the torques recalculated and the errors determined. The results are shown in Figs. 16–21. The highest errors in 5-, 50-,

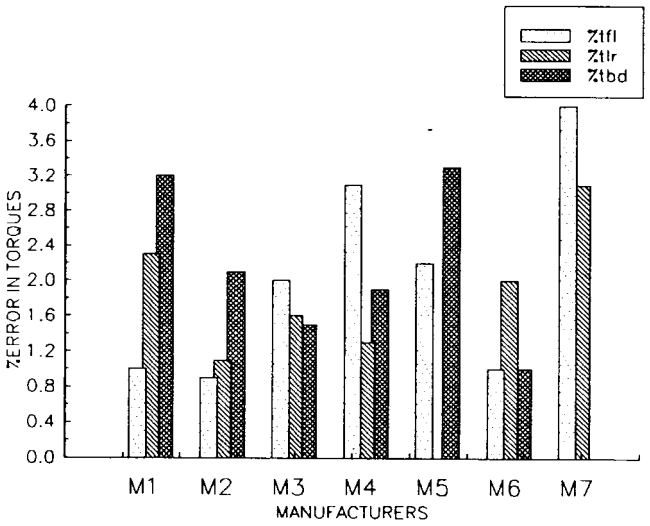

Fig. 19. Genetic performance using deep-bar model for 5-hp motor.

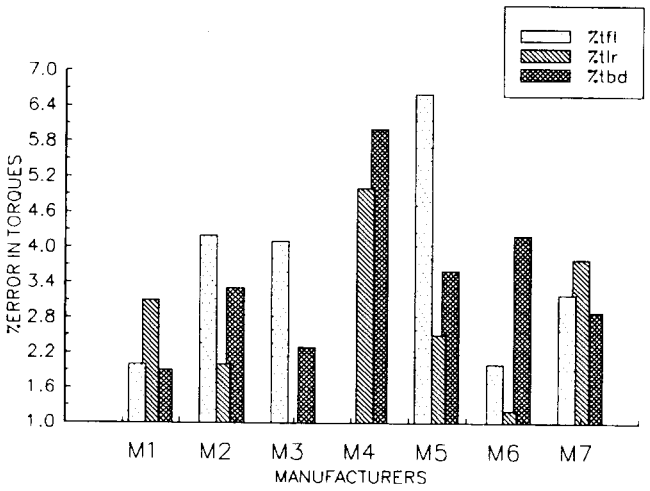

Fig. 20. Genetic performance using deep-bar model for 50-hp motor.

and 100-hp motors using the single-cage model are 12.0%, 13.8%, and 17.0%, while for the deep-bar model, the errors are 4.0%, 6.6%, and 9.4%, respectively. The seven-parameter model produces lower errors in torques than those of the five-parameter model, and all the errors are within 10%.

VI. Conclusion

This paper has applied the genetic algorithm to the problem of motor parameter determination to allow the calculation of torque transients. The input data set is generic in nature, and the parameters obtained are suitable for system-type studies, for protection, for example, but may not be suitable for precise applications like vector-controlled drives. Several different versions of the genetic algorithm were examined by calculating the parameters for a small (5-hp), medium (50-hp), and a large (500-hp) induction motor. V4 produces extremely good results when the torques were generated from the EC with known parameters. Larger errors were produced when using the single-cage model and the actual data from several manufacturers, due to the neglect of parameter variations and

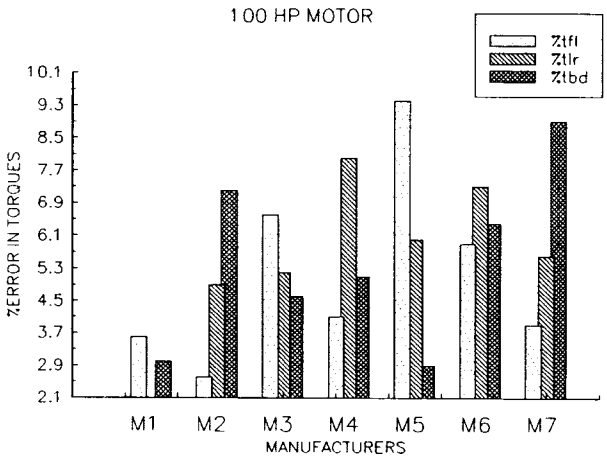

Fig. 21. Genetic performance using deep-bar model for 100-hp motor.

deep-bar effects in the model. A deep-bar model was then used with improved ability to predict the parameters. The use of the Newton–Raphson method was also demonstrated, and its sensitivity to the initial starting values was highlighted.

References

[1] Z. Zhang, G. E. Dawson, and T. R. Eastham, "Evaluation of dynamic parameters and performance of deep-bar induction machines," in *Proc. IEEE-IAS Annu. Meeting*, 1993, pp. 62–66.

[2] S. I. Moon and A. Keyhani, "Estimation of induction machine parameters from standstill time domain data," in *Proc. IEEE-IAS Annu. Meeting*, 1993, pp. 336–342.

[3] J. R. Willis, G. J. Brock, and J. S. Edmonds, "Derivation of induction motor models from standstill frequency-response tests," *IEEE Trans. Energy Conversion*, vol. 4, pp. 608–615, Dec. 1989.

[4] T. A. Lipo and A. Consoli, "Modeling and simulation of induction motors with saturable leakage reactances," *IEEE Trans. Ind. Applicat.*, vol. IA-21, pp. 180–189, Jan./Feb. 1984.

[5] A. Keyani and H. Tsai, "IGSPICE simulation of induction machines with saturable inductances," *IEEE Trans. Energy Conversion*, vol. 4, pp. 118–125, Mar. 1989.

[6] L. Zai, C. L. de Marco, and T. A. Lipo, "An extended Kalman filter approach to rotor time constant measurement in PWM induction motor drives," *IEEE Trans. Ind. Applicat.*, vol. 28, pp. 96–104, Jan./Feb. 1992.

[7] J. Holtz and T. Thim, "Identification of the machine parameters in a vector-controlled induction motor drive," *IEEE Trans. Ind. Applicat.*, vol. 27, pp. 1111–1118, Nov./Dec. 1991.

[8] J. A. de Koch, F. S. van der Merwe, and H. J. Vermeuler, "Induction motor parameter estimation through an output error technique," presented at the IEEE-PES Winter Power Meeting, Columbus, OH, 1993, Paper 93-WM 019-9 EC.

[9] T. A Higgins, W. L. Snider, P. L. Young, and H. J. Holley, "Report on bus transfer: Part I—Assessment and application," *IEEE Trans. Energy Conversion*, vol. 5, pp. 462–469, Sept. 1990.

[10] S. S. Mulukutla and E. M. Gulachenski, "A critical survey of considerations in maintaining process continuity during voltage dips while protecting motors with reclosing and bus-transfer practices," *IEEE Trans. Power Syst.*, vol. 7, pp. 266–272, Aug. 1992.

[11] B. K. Johnson and J. R. Willis, "Tailoring induction motor analytical models to fit known motor performance characteristics and satisfy particular study needs," *IEEE Trans. Power Syst.*, vol. 6, pp. 959–965, Aug. 1991.

[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[13] *EMTP Revised Rule Book, Version 2.0*, EPRI EL-6421-L, vol. 1, Version 2.0, Electric Power Research Institute, Palo Alto, CA, June 1989.

[14] G. J. Rogers and D. Shirmohammadi, "Induction machine modeling for electromagnetic transient program," *IEEE Trans. Energy Conversion*, vol. EC-2, pp. 622–628, Dec. 1987.

Pragasen Pillay (S'84–M'87–SM'92) received the Bachelor's degree from the University of Durban-Westville, Durban, South Africa, in 1981, the Master's degree from the University of Natal, Durban, in 1983, and the Ph.D. degree from Virginia Polytechnic Institute and State University, Blacksburg, in 1987, while funded by a Fulbright Scholarship.

From January 1988 to August 1990, he was with the University of Newcastle upon Tyne, Newcastle upon Tyne, U.K. From August 1990 to August 1995, he was with the Electrical Engineering Department, University of New Orleans, New Orleans, LA. He is currently a Professor in the Department of Electrical and Computer Engineering, Clarkson University, Potsdam, NY, where he holds the J. Newell Distinguished Professorship in Engineering. His research and teaching interests are in modeling, design, and control of electric motors and drives.

Dr. Pillay is a Member of the IEEE Power Engineering, IEEE Industry Applications, IEEE Industrial Electronics, and IEEE Power Electronics Societies. He is a member of the Electric Machines Committee, Vice Chairman (Programs) of the Industrial Drives Committee, and Vice Chairman of the Continuing Education Subcommittee within the IEEE Industry Applications Society. He has organized and taught short courses in electric drives at the Annual Meeting of the Industry Applications Society. He is a member of the Institution of Electrical Engineers, U.K., and a Chartered Electrical Engineer in the U.K.

Ray Nolan received the B.S. and M.S. degrees in electrical engineering from the University of New Orleans, New Orleans, LA, in 1991 and 1994, respectively.

He is currently an Electrical Engineer with Marrero, Couvillon & Associates, Metairie, LA.

Towhidul Haque received the B.S.E.E. degree from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 1992 and the M.S.E.E. degree from the University of New Orleans, New Orleans, LA, in 1995.

He is currently a Software Engineer with Denro, Inc., Gaithersburg, MD, where he has designed and developed real-time embedded software for integrated voice and data communications switching systems.