A New Public-Domain Simulator for Power Electronic Circuits

Mahesh B. Patil, Senior Member, IEEE, Shyama P. Das, Member, IEEE, Avinash Joshi, and Mukul Chandorkar, Member, IEEE

Abstract—A new public-domain simulator for power electronic circuits is described. The organization of the simulator is briefly discussed. The most important feature of the simulator is that the user can define new elements in a flexible manner. The differences between the new simulator and other simulators are enumerated. Some simulation examples are discussed to demonstrate the applications of the simulator. It is pointed out that the new simulator is particularly attractive for engineering institutes in developing countries where access to expensive commercial packages with similar capabilities may be difficult.

Index Terms—Circuit simulation, power electronics, public-domain package.

I. INTRODUCTION

C IRCUIT simulation is a powerful tool for circuit design and evaluation, and, over the years, many computer programs have been developed for this purpose [1]–[10]. Most of these programs are designed so that the commonly used circuit elements are incorporated within the program, and the user can call them in the circuit file following certain syntax rules and keywords specified by the program designer. However, incorporation of a new element by the user is generally impossible, or only allowed in a limited manner by modification of the model equations of an existing element. This situation reduces the flexibility offered to the user for solving new problems.

The simulation package SABER [11] was developed to give the user the capability to add new elements to the simulation package with ease and flexibility. In SABER, a new element could be added by creating a template following certain syntax rules, and the model equations could either be written in a specially designed language called MAST or supplied using a C or FORTRAN routine. The user does not need to know the details about how the templates are interpreted by the program; she or he must simply describe the element behavior in a high-level language. Because this is a relatively straightforward exercise, it is not surprising that SABER has now become popular and is used for many applications.

The purpose of this paper is to present a <u>Solver</u> for circuit <u>EQ</u>uations with <u>U</u>ser-defined <u>EL</u>ements (SEQUEL), which, like SABER, has been developed with the emphasis on *user-defined* elements. Apart from some major differences in terms of

S. P. Das and A. Joshi are with the Department of Electrical Engineering, Indian Institute of Technology (IIT), Kanpur 208016, India.

Publisher Item Identifier S 0018-9359(02)01296-7.

technical capabilities, SEQUEL is a public-domain program, as opposed to SABER. SEQUEL can be downloaded from the Internet [12], and it can run on a personal computer (PC) with Linux 6.0. From the educational point of view, SEQUEL offers the following advantages. First, the program can be installed on any number of computers, making it simultaneously accessible to a large number of students. In fact, with this advantage of the program in mind, we recently conducted a course for teachers of engineering colleges in India, where the idea of using SEQUEL for teaching purposes was well appreciated by the participants. Second, vendors of commercial packages usually tend to keep the implementation concealed from the user so that other vendors do not benefit from their implementation. For example, one finds that most of the templates (excluding the trivial ones like a resistor) in SABER are coded so that the user has no idea of how a particular element has been implemented. This concealment means that the user is not fully aware of the equations that are being solved, which is clearly an impediment for academic work. On the other hand, all of the templates in SEQUEL are available in source code form so that the user can easily see which equations are being solved and how they have been implemented, making the package much more interesting for teaching.

The paper is organized as follows. Section II describes the organization of SEQUEL from the user's point of view. Some simulation examples are presented in Section III, each bringing out some unique feature of the program. Because a comprehensive manual is available for SEQUEL [12], some aspects of the package, such as syntax rules, have not been described here. However, some representative templates and a circuit file have been included in the Appendix.

II. SEQUEL ORGANIZATION

SEQUEL has been developed with the emphasis on model development by the user. A simplified block diagram of the program [12] is shown in Fig. 1. Here, the block on the left, which includes basic element templates (BETs), compound element templates (CETs), and circuit file, is completely user definable. Basic elements are individual elements at the lowest level. The model equations describing the behavior of the element are to be included in the BET. From the BETs, the user can make up a CET; however, a CET can also include another CET, thus making it a recursive definition (this is similar to the subcircuit idea in SPICE [9]). In other words, BETs describe the equations, and CETs may be thought of simply as connections. The actual organization of SEQUEL is a little more complicated because

0018-9359/02\$17.00 © 2002 IEEE

Manuscript received December 20, 1999; revised October 22, 2001.

M. B. Patil and M. C. Chandorkar are with the Department of Electrical Engineering, Indian Institute of Technology (IIT), Bombay, Powai, Mumbai 400076, India.

Fig. 1. Simplified block diagram of SEQUEL.

it distinguishes between electrical, thermal, general, and digital elements, and correspondingly, there are BETs and CETs of four different types. However, those details are described in [12].

The circuit file appearing in Fig. 1 is a description of the circuit being simulated. The circuit may have elements of one or more of the four types, electrical, thermal, general, and digital. The main program reads the circuit file and picks out the required CETs and BETs from the library. This information is used in parsing [8], i.e., decomposition of the entire circuit into basic elements. The SEQUEL preprocessor also reads the circuit file and prepares the required basic element (BE) routines (in FORTRAN), which are to be called by the main program in the course of the simulation. In doing so, the preprocessor transfers the user's description of model behavior *verbatim* from the BET to the BE routine. The important issue of how the equations are to be entered by the user will now be addressed, but first, the type of equations being solved and the method for solving them will be briefly discussed.

SEQUEL is intended for solving a generally nonlinear system of equations of the form

$$f_i(x_1, x_2, \ldots, x_N) = g_i$$

where $g_i = 0$ or $g_i = dx_j/dt$. The method employed by SEQUEL to solve the nonlinear equations is the well-known Newton–Raphson (NR) iterative method [6]. To go from one NR iteration to the next, each of the N functions and its derivatives must be evaluated with respect to each of the unknowns involved. Thus, if the user supplies routines (in the form of BETs) to compute the functions and derivatives, SEQUEL can assemble the equations together and find the solution. In the case of an electrical circuit, some of the equations (Kirchhoff's current law and Kirchholff's voltage law) depend only on the circuit topology. These equations are assembled automatically by SEQUEL like any other circuit simulation program. The rest of the equations are element-dependent and will be called the basic element constitutive equations (BECE). The sparse tableau [13] approach is used in SEQUEL for writing the BECEs.

A resistor is an example. The BECE in this case is $v_1 - Ri_1 = 0$ where v_1 and i_1 are the basic element voltage (BEV) and basic element current (BEC), respectively. For a resistor, there is only one BEV and BEC; however, the subscripts on v and i will be required in general. In Appendix A, a BET for the resistor is shown. This BE has two nodes term1 and

term2, and one real parameter \mathbf{r} ($\mathbf{r} = 0$ indicates that its default value is 0). The BEV (v1) is defined as the difference between the node voltages at term1 and term2. This BE has one function (i.e., one BECE) that depends upon the BEV v1 and the current (from term1 to term2). The square brackets after v1 indicate that the derivative of this function with respect to v1 is *always* a constant (=1), independent of any parameter or variable. The FORTRAN lines are used to calculate the function and the derivative with respect to the current. The logical flags (1_dc, 1_trns, 1_function, 1_jacobian) need some explanation.

As previously mentioned, the SEQUEL preprocessor prepares a BE routine, where the FORTRAN section of the BET is copied *as is* to the BE routine. Now, this routine, in the course of the simulation, may be called by the main program in a variety of situations such as direct current (dc), transient, function evaluation, Jacobian calculation, and sensitivity calculation. The flags, such as l_function, are passed down from the main program to indicate the simulation stage. The user, while writing the FORTRAN section of the BET, anticipates the different situations and includes suitable calculations as required. When these are understood, the functioning of the resistor template should be self-explanatory. A more complicated BET, for an induction motor, is given in the Appendix; it will be discussed in Section III. There are a large number of other templates available to the user [12].

Described above is the basic idea behind SEQUEL, and it differs from SABER in the following major aspects. First, in SE-QUEL, the user is asked to provide the function values and the derivatives in the BE template. If the number of equations is large (about 10), the exercise of writing the derivatives is somewhat cumbersome. However, from the computational point of view, a routine that provides function values and their derivatives in a single call is far more efficient. Imagine a routine that only gives the function values. To compute the derivatives required by the Newton-Raphson method, this routine would have to be called repeatedly (with slightly different solution vectors) to get approximate derivative values. This is clearly a computationally expensive procedure. Second, in SEQUEL, the concept of through and across variables [11] is not required because it is based on the sparse tableau approach [13]. A given function is simply a function of N variables x_1 through x_N , which are all treated in the same manner. The user only has to specify how the functions and the nonzero derivatives are to be evaluated, given the solution vector. This approach gives the user greater flexibility in writing the BE equations. Note that the NR method does not require the equations to be written in terms of through and across variables; the need to cast the equations in that form arises because of the modified nodal analysis [6] approach used in SABER. Third, if the user is interested in developing a new template, SABER requires the knowledge of the hardware description language called MAST [11], which can prove to be an impediment. On the other hand, writing a template in SEQUEL requires knowledge of FORTRAN. Because FORTRAN is a general-purpose computer language, students are likely to have learned it in the first year of their engineering curriculum. Finally, SEQUEL provides very efficient computation of steady-state quantities, as is illustrated in Section III.

Fig. 2. Three-phase rectifier circuit.

This feature is not available in any other general-purpose circuit simulation program. There are, of course, many other aspects of SEQUEL which deserve explanation; however, those aspects can be found in [12].

III. SIMULATION EXAMPLES

In this section, simulation examples of typical power electronic circuits are presented. The circuit files for these examples are available, and the reader can simply run the program on these to get the results that are described here; also, the reader can simulate other circuits of interest by modifying the circuit files or writing new files. The detailed syntax rules for writing circuit files can be found in the SEQUEL manual [12].

A. Three-Phase Rectifier

Fig. 2 shows a six-diode bridge connected to a three-phase alternating current (ac) supply with a source inductance L_s . The bridge supplies an RL load. The output voltage V_{out} has a dc component equal to $1.35V_{LL}$ [V_{LL} being the root mean square (rms) value of V_{ab}] if ideal operation of the bridge is assumed. However, because of the inductance L_s , the conducting intervals may overlap, resulting in modes when more than two diodes are conducting at the same time. The number of devices conducting at a time and the average output voltage depend on the load current i, which can be adjusted by changing R while keeping L constant. From a simulation point of view, this circuit may be considered difficult due to the presence of highly nonlinear switching elements such as diodes. For simulation of this circuit, the diodes were represented as hard switches with ON and OFF resistances $R_{\rm on} = 0.1 \ {\rm m}\Omega$ and $R_{\rm off} = 1 \ {\rm M}\Omega$, respectively. Transient simulation is performed for a sufficiently long time interval in order to reach a steady state. The parameters used in the simulation are $\hat{V} = 325$ V, f = 50 Hz, $L_1 = L_2 =$ $L_3 = 500 \ \mu\text{H}, R = 0.45 \ \Omega$, and $L = 200 \ \text{mH}$. The source voltages shown in Fig. 2 are given by $V_1 = V \sin(\omega t), V_2 =$ $\hat{V}\sin(\omega t - 2\pi/3)$, and $V_3 = \hat{V}\sin(\omega t + 2\pi/3)$. Fig. 3 shows the steady-state current through D1, and Fig. 4 shows the load current as a function of time.

Fig. 4 illustrates very effectively how computation of steadystate quantities may take several cycles of transient simulation; in this example, it has taken four seconds or 200 cycles. In reality, only the steady-state results (i.e., *one* cycle) may be of

Fig. 3. Steady-state current through D1 (amperes) for the circuit of Fig. 2.

Fig. 4. Load current for the three-phase diode bridge rectifier.

interest. SEQUEL provides an efficient way to do this computation. Although the details of the procedure will be described separately, the basic idea can be explained as follows [14]. The state of a circuit is completely determined if the values of all state variables (in our example, the inductor currents) are known at a given time. Thus, the steady-state computation problem reduces to finding a state (i.e., a state *vector*) such that, after integrating for one cycle, the system is back in the same state. This can be achieved very effectively by employing the NR method [14]. In the three-phase rectifier example, the steady-state results were obtained in only four NR iterations with this method, which is equivalent to four cycles of the source voltage. This method will be referred to as the steady-state waveform (SSW) method.

Because the SSW method is essentially an NR method, it exhibits quadratic convergence. To appreciate this property, a norm must first be defined. The equations being solved in the SSW method are $i_k(T) = i_k(0)$, where i_k is the current for inductor k and T represents the period. A norm is now defined as $\sum (i_k(T) - i_k(0))^2$. The dramatic reduction in the norm with respect to the iteration number for this example can be observed in Fig. 5. Here and in many other examples, a computational advantage of an order of magnitude or more is easily obtained with the SSW method over the conventional transient simulation method.

Fig. 5. Steady-state computation: norm versus iteration number for the diode-bridge rectifier example.

B. Induction Motor Examples

Induction motor model in the stationary d-q reference frame is a highly nonlinear system. The system behavior is described by the following equations [15]:

$$i_{ds} = \frac{l_r}{l_m l_e} \psi_{ds} - \frac{1}{l_e} \psi_{dr} \tag{1}$$

$$i_{dr} = \frac{1}{l_m} \psi_{ds} - \left(\frac{l_{ls}}{l_m} + 1\right) i_{ds} \tag{2}$$

$$i_{qs} = \frac{l_r}{l_m l_e} \psi_{qs} - \frac{1}{l_e} \psi_{qr} \tag{3}$$

$$i_{qr} = \frac{1}{l_m} \psi_{qs} - \left(\frac{l_{ls}}{l_m} + 1\right) i_{qs} \tag{4}$$

$$T_{em} = \frac{3}{4} l_m (i_{qs} i_{dr} + i_{ds} i_{qr})$$
(5)

$$\omega_r = \frac{P}{2} \,\omega_{rm} \tag{6}$$

$$\psi_{ds} = v_{ds} - r_s i_{ds} \tag{7}$$

$$\dot{\psi}_{qs} = v_{qs} - r_s i_{qs} \tag{8}$$

$$\dot{\psi}_{dr} = -\omega_r \psi_{qr} - r_r i_{dr} \tag{9}$$

$$\psi_{qr} = \omega_r \psi_{dr} - r_r i_{qr} \tag{10}$$

$$\dot{\omega}_r = \frac{P}{2} \frac{T_{em} - T_L}{J}.$$
(11)

Equations (7)–(10) correspond to a variation of electrical quantities, and (11) relates the electromagnetic torque developed by the motor with the motor speed. Equations (1)–(11) must be solved simultaneously to yield the currents and speed of the induction motor. A basic element template that implements these equations is shown in Appendix B. The induction motor is treated as a general element for which the derivatives are supplied with respect to the general variables (listed under main_var) rather than electrical currents and voltages. In the template, the variables involved in each equation must be indicated, and the evaluation of the function and Jacobian values must be incorporated in the form of FORTRAN statements. Note that dfdvr(m, n) denotes the derivative of function mwith respect to variable n. In the course of the simulation, the

Fig. 6. Transient torque-speed characteristics of induction motor under free acceleration.

main program passes down the values of the different variables and parameters to the template, and the template conveys to the main program function values and the corresponding derivatives. There are no restrictions on the variables. This situation must be contrasted with SABER templates, where it is the user's responsibility to identify certain variables as the *through* variables and others as the *across* variables, an exercise which can turn out to be rather tedious and unnatural.

Two examples will now be described to illustrate the applications of the induction machine template. In these examples, other templates are also required; these are available [12] as source code. It would be instructive to go through the concerned templates for each example and see how the various quantities are *mapped* at different levels.

As the first example, free acceleration of an induction motor is considered. When an induction motor is accelerated from standstill by applying a set of rated sinusoidal voltages, it goes through a transient state and then reaches the steady state. Fig. 6 shows the simulated torque-speed transient. The machine parameters were taken from [15, Table 4.10-1, 3-hp machine]. The result matches that given in [15, Fig. 4.10-1].

As the second example, indirect vector control of an induction motor is considered. Application of the well-established vector control theory enables an induction motor to be controlled like a separately excited dc drive with torque and flux decoupled. Fig. 7 shows the schematic diagram of an indirect vector controlled induction motor drive. The inverter is operated under current controlled mode using three hysterisis current controllers for the three phases. The complete closed loop control system was simulated. The following parameters were used: $r_s = 6.3456\Omega$; $l_{ls} = 0.0162$ H; $l_m = 0.2628$ H; $l_{lr} = 0.0162$ H; $r_r = 4.346\Omega$; J = 0.0093 Kg-m²; number of poles = 4; $i_{ds}^{e\star} = 3.8$ A; h (hysterisis band) = 0.1 A; $V_{dc} = 300$ V; $K_P = 0.05$; and $K_I = 0.02/s$. Fig. 8 shows the response of the speed and the phase a motor current to a step change of the reference speed followed by a reversal.

In Appendix C, the circuit file used in the vector-controlled induction motor example is reproduced. A close examination of the block diagram (Fig. 7) and this circuit file will reveal the correspondence between the variables and elements in the two descriptions. Most of these are self-explanatory. The

Fig. 7. Schematic diagram of an indirect vector-controlled induction motor drive.

Fig. 8. Speed (rad/s) and i_a (amperes) versus time (seconds) for the induction motor drive of Fig. 7. $K_P = 0.05$, $K_I = 0.02/s$.

element vctrrttr (a vector rotator) relates the variables $i_a, i_b, i_c, i_{qs}, i_{ds}$, and θ as

$$\begin{split} i_a &= \cos \theta i_{qs} + \sin \theta i_{ds} \\ i_b &= \cos \left(\theta - \frac{2\pi}{3} \right) i_{qs} + \sin \left(\theta - \frac{2\pi}{3} \right) i_{ds} \\ i_c &= \cos \left(\theta + \frac{2\pi}{3} \right) i_{qs} + \sin \left(\theta + \frac{2\pi}{3} \right) i_{ds}. \end{split}$$

The element vctrrttr1 is like vctrrttr, for the special case $\theta = 0$. Again, as has been noted earlier, the corresponding templates clearly indicate the implementation of these equations.

The purpose of the two examples described here was primarily to demonstrate the capabilities of SEQUEL. The reader should note, in particular, that a fairly complex model, such as an induction motor, is available as a library element, and it can be simply called from the user's circuit file. Such a facility is not available in other public-domain packages. Also, the reader will find that the induction motor template involves a straightforward translation of the model equations into FORTRAN statements. If new templates must be written for teaching or research work, the user could develop them easily, with the help of the manual.

In conclusion, a new *public-domain* simulator for simulation of power electronic circuits has been presented. Some examples have been presented to describe the capabilities of the new simulator. SEQUEL has many other interesting features and applications, such as electrothermal circuits, electronic circuits, small-signal analysis, mixed-signal simulation, sensitivity calculation, and switched-capacitor circuit simulation. Details regarding these features and about the syntax used in SEQUEL can be found in the manual [12]. The reader can reproduce all of the results presented in the paper (and more) with the circuit files provided in [12].

Although a limited number of examples have been described here, SEQUEL already has a reasonably complete library for power electronics applications. Circuit files for many other examples including converters, closed-loop control, inverters, and power systems can be obtained from the authors. Because SE-QUEL can be installed on any number of PCs, homework problems based on simulation can be assigned to students even in large classes. Simulation has been found to help the students greatly even for fairly simple circuits. With the appropriate circuit files available, the effort required on the part of the student is greatly reduced. Apart from being completely free, SEQUEL also allows the users to closely look at all of the templates involved in a given circuit. This enables the reader to understand all aspects of the implementation, something that is not possible with commercial packages. In our opinion, this is also of great educational value.

APPENDIX

Described here are two basic element templates and a circuit file. Only the statements directly relevant to the discussion in the paper are given below; the complete templates and files can be obtained from the author [12].

A. Basic Element Template for a Resistor

```
ebe name = resistor
nodes: term1 term2
rparms: r = 0
be_v: v1 = term1 - term2
no_functions = 1
f_1: v1[1] current
fortran:
variables:
source:
      if (l_dc.or.l_trns) then
         r = rparm(1)
         v1 = v(1)
         if (1 \text{-function}) f(1) = v1 - r * cur(1)
         if (l_jacobian) then
           dfdi(1,1) = -r
         end if
      end if
endfortran
endebe
```

B. Basic Element Template for an Induction Machine

gbe name = indmc $main_var =$ + ids idr iqs iqr psids psidr + psiqs psiqr wr wrm tem + vds vqs rparms: rs = 0 lls = 0 lm = 0llr = 0 rr = 0j=0 tl=0 poles =4+ls = 0 lr = 0 le = 011 = 0 12 = 0+13 = 0 x1 = 0 x2 = 0+ $no_functions = 11$ f_1: ids[1] psids psidr f_2: idr[1] psids ids f_3: iqs[1] psiqs psiqr f_4: iqr[1] psiqs iqs f_5: tem[1] iqs idr ids iqr f_6: wr[1] wrm f_7: d_dt(psids) vds ids f_8: d_dt(psiqs) vqs iqs f_9: d_dt(psidr) wr psiqr idr f_10: d_dt(psiqr) wr psidr iqr f_11: d_dt(wrm) tem fortran: source: if (l_trns) then if (l_function) then f(1) = ids - (l1 * psids) + (psidr/le)f(2) = idr - (psids/lm) + (l2 * ids)f(3) = iqs - (l1 * psiqs) + (psiqr/le)

f(4) = iqr - (psiqs/lm) + (l2 * iqs)f(5) = tem - x1 * (iqs * idr - ids * iqr)f(6) = wr - x2 * wrmf(7) = vds - rs * idsf(8) = vqs - rs * iqsf(9) = -wr * psiqr - rr * idrf(10) = wr * psidr - rr * iqrf(11) = (tem - tl)/jend if if (l_jacobian) then dfdvr(1, 5) = -11dfdvr(1, 6) = 1.0/ledfdvr(2, 1) = 12dfdvr(2, 5) = -1.0/lmdfdvr(3, 7) = -11dfdvr(3, 8) = 1.0/ledfdvr(4, 3) = 12dfdvr(4, 7) = -1.0/lmdfdvr(5, 3) = -x1 * idrdfdvr(5, 2) = -x1 * iqsdfdvr(5, 1) = x1 * iqrdfdvr(5, 4) = x1 * idsdfdvr(6,10) = -x2dfdvr(7, 12) = 1.0dfdvr(7, 1) = -rsdfdvr(8, 13) = 1.0dfdvr(8, 3) = -rsdfdvr(9, 9) = -psiqrdfdvr(9, 8) = -wrdfdvr(9, 2) = -rrdfdvr(10, 9) = psidrdfdvr(10, 6) = wrdfdvr(10, 4) = -rrdfdvr(11, 11) = 1.0/jend if end if endfortran endgbe

title: imvc # vector controlled ind mc with step input begin_circuit gelement name = im1 type = indmc + ids = ids idr = idr iqs = iqs iqr = iqr + psids = psids psidr = psidr psiqs = psiqs

C. Circuit File for Vector-Controlled Induction Motor Example

- + psiqr = psiqr wr = wr wrm = wrm tem = tem
- + vds = vds vqs = vqs
- + rs = 6.3456 lls = 0.0162 lm = 0.2628
- + llr = 0.0162 rr = 4.346 j = 0.0093
- + tl = 0 poles = 4
 gelement type = pulse1 y = wrmstar
 + t1 = 0.5 t2 = 1.5 val1 = 0 val2 = 60 val3 = -60
- gelement type = const y = idsestar c = 3.8
 gelement type = diff_2 x1 = wrmstar x2 = wrm
 + y = e
- gelement type = picntrl x = e y = iqsestar

- $+ kp = 0.05 ki = 0.02 y2_ig = 0 y2_sv = 0$ gelement type = lmtr x = iqsestar+ y = iqsestar1 xmin = -13 xmax = 13gelement type = $div_2 x1 = iqsestar1$ + x2 = idsestar y = rgelement type = multscl x = r y = wsl+ a = 15.577gelement type = $sum_2 x1 = wsl x2 = wr y = we$ gelement type = intgrtr x = we y = thetaegelement type = vctrrttr ia = iastar ib = ibstar ic = icstariqs = iqsestar1 ids = idsestar++ theta = thetae gelement type = vctrrttr1 ia = ia ib = ib ic = ic iqs = iqs ids = idsgelement type = invrtr vao = vao vbo = vbo vco = vco+ia = ia ib = ib ic = iciastar = iastar ibstar = ibstar+icstar = icstar++ h = 0.1 vdc = 300gelement type = vaotovanvan = van vbn = vbn vcn = vcn+vao = vao vbo = vbo vco = vco+gelement type = abctodqvan = van vbn = vbn vcn = vcn
- + vqs = vqs vds = vds
- end_circuit

REFERENCES

- S. W. Director, Ed., Computer-Aided Circuit Design: Simulation and Optimization. Stroudsburg, PA: Dowden, Hutchinson & Ross, 1973.
- [2] L. O. Chua and P. M. Lin, Computer-Aided Analysis of Electronic Circuits. Englewood Cliffs, NJ: Prentice-Hall, 1975.
- [3] R. W. Jensen and L. P. McNamee, Eds., Handbook of Circuit Analysis Languages and Techniques. Englewood Cliffs, NJ: Prentice-Hall, 1976.
- [4] J. Vlach and K. Singhal, Computer Methods for Circuit Analysis and Design. New York: Van Nostrand Reinhold, 1983.
- [5] D. O. Pederson, "A historical review of circuit simulation," *IEEE Trans. Circuits Syst.*, vol. CAS-31, Jan. 1984.
- [6] W. J. McCalla, Fundamentals of Computer-Aided Circuit Simulation. Norwell, MA: Kluwer , 1987.
- [7] A. F. Schwarz, Computer-Aided Design of Microelectronic Circuits and Systems. London: Academic, 1987.
- [8] R. Raghuram, Computer Simulation of Electronic Circuits. New York: Wiley, 1989.
- [9] M. H. Rashid, SPICE for Circuits and Electronics Using PSPICE, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1995.
- [10] K. S. Kundert, Designer's Guide to SPICE and SPECTRE. Norwell, MA: Kluwer, 1995.
- [11] SABER Reference Manual, Analogy, Inc., Beaverton, OR, 1996.
- [12] M. B. Patil. SEQUEL users' manual. [Online]. Available: http://www.ee.iitb.ernet.in/~microel/faculty/download.html.

- [13] C. D. Hachtel, R. K. Brayton, and F. G. Gustavson, "The sparse tableau approach to network analysis and design," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 101–113, Jan. 1971.
- [14] T. J. Aprille and T. N. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," *Proc. IEEE*, vol. 60, pp. 108–114, 1972.
- [15] P. C. Krause, *Analysis of Electric Machinery*. New York: McGraw-Hill, 1986.
- [16] I. Takahashi and T. Noguchi, "A new quick-response and high-efficiency control of an induction motor," *IEEE Trans. Ind. Applicat.*, pp. 820–827, Sept./Oct. 1986.

Mahesh B. Patil (S'91–M'95–SM'01) received the B.Tech. degree from the Indian Institute of Technology (IIT), Bombay, India, in 1984, the M.S. degree from the University of Southern California, Los Angeles, CA, in 1987, and the Ph.D. degree from the University of Illinois, Urbana-Champaign, in 1992, all in electrical engineering.

He was a Visiting Researcher with the Central Research Laboratories, Hitachi, Tokyo, Japan, in 1993. From 1994 to 1999, he was a faculty member with the Electrical Engineering Department at IIT, Kanpur, India. He is currently on the faculty of the Electrical Engineering Department at IIT Bombay. His research interests include device modeling and simulation, and circuit simulation.

Shyama P. Das (M'97) was born in Cuttack, India, in 1968. He received the B.Tech. (Hons.) degree in electrical engineering and the M.Tech. and Ph.D. degrees in electrical engineering from the Indian Institute of Technology (IIT), Kharagpur, India, in 1990, 1992, and 1997, respectively.

From 1996 to 1997, he was a Lecturer in the Electrical Engineering Department, Regional Engineering College, Rourkela, India. In 1997, he joined the Electrical Engineering Department, IIT, Kanpur, India, where he is currently working as an Assistant Professor. He has published several papers in national and international journals and conferences. His research interests include power electronics, high performance industrial drives, and microprocessor-based control and instrumentation.

Dr. Das is a life member of Institute of Electronics and Telecom Engineers (IETE), India. He received second prize for best M.Tech. thesis by the Indian Society for Technical Education in 1992.

Avinash Joshi received the Ph.D. degree in EE from University of Toronto, Canada in 1979.

He is a Professor of Electrical Engineering at the Indian Institute of Technology (IIT), Kanpur, India. He also worked for General Electric Company in India. His research interests include power electronics and applications.

Mukul Chandorkar (M'85) received the B.Tech. degree from the Indian Institute of Technology (IIT), Bombay, India, the M.Tech. degree from the Indian Institute of Technology, Madras, India, and the Ph.D. degree from the University of Wisconsin, Madison, in 1984, 1987, and 1995, respectively, all in electrical engineering.

He has several years of experience in the power electronics industry in India, Europe and the United States. From 1996 to 1999, he was with ABB Corporate Research, Ltd., Baden, Switzerland. He is currently an Associate Professor in the Electrical Engineering Department of IIT Bombay. His technical interests include uninterruptible power supplies, electric drives, and the measurement and analysis of power quality.