International Journal of Engineering Education Vol. 27, No. 2, pp. 217–224, 2011
Printed in Great Britain

0949-149X/91 $3.00+0.00

SPICE as a Fast and Stable Tool for Simulating a Wide Range of Dynamic Systems*

JOEL D. HEWLETT and BOGDAN M. WILAMOWSKI
Department of Electrical and Computer Engineering, Auburn University, Auburn, Alabama, USA. E-mail: hewlejd@auburn.edu

This paper introduces a generalized method for simulating dynamic systems in SPICE. The proposed method is useful for students in that the necessary software is free and only an elementary knowledge of SPICE is required. The method uses a netlist description of the system that comprises little more than a set of state equations. In comparison with many of the more commonly used tools, such as MATLAB/SIMULINK, the presented method is less involved, allowing the student to stay focused on the problem rather than the software. The method is not limited only to electrical systems: it can be applied to virtually any dynamic system, making it accessible to students from other backgrounds.

Keywords: SPICE; system dynamics; state-variable analysis; computer aided analysis

1. Introduction

The modern engineer enjoys the luxury of a powerful and diverse computational toolset to aid in the problem solving process. Unfortunately for the engineering student, most of these tools are cost prohibitive, and they may also involve a rather steep learning curve. This paper offers an alternative approach to analyzing dynamic systems that is straightforward, and can be implemented using tools that are available at no cost to the student. The method uses the powerful numerical solvers designed for Simulation Programs with Integrated Circuit Emphasis (SPICE). While the educational benefits of SPICE for electrical applications are well documented [1–4], here, a more general approach is taken that allows nearly any dynamic system to be simulated in SPICE. Furthermore, the proposed method is presented in a way that does not require the user to have a detailed knowledge of electronics, making it accessible to students from across the various engineering disciplines.

One of the most attractive benefits of SPICE is that there are a number of versions that are available to students free of charge, allowing the student to use the software at home. Free versions include LTSpice [11], SIP [13], and a student version of the widely used PSPICE [12]. Both LTSpice and PSPICE are used professionally, and are based on the open source SPICE3 [14] engine, which was developed at the University of California, Berkeley. SIP is a web based equivalent written in Perl. There are some restrictions on the functionality of the student version of PSPICE; however the classes of problems presented here are not subject to these limitations. That is to say, there is effectively no limit to the size of the problems that can be analyzed.

Another benefit of SPICE is that, while it is quite powerful for the proposed method, its interface and functionality are not overwhelmingly complex. This is because the proposed method bypasses the schematic interface and instead uses a netlist description. The netlist is a text based description of the circuit that can be written in any basic text editor. For students, computer aided examples can illustrate principles in ways that might otherwise be unclear. Unfortunately, it is possible for students to become confused by the simulation process and fail to appreciate the results, which could defeat the purpose of the exercise. The method presented in this paper may be beneficial to students since it is intended to place more emphasis on the results by simplifying the simulation process.

In addition to the instructional benefits, the presented techniques also have practical value. For certain classes of high-order dynamic systems with relatively high degrees of stiffness, SPICE often outperforms many of the most commonly used tools, including MATLAB and SIMULINK. For example, SPICE has been successfully used to simulate a 25th order oil pump-jack [5], for which the SPICE simulation time was over 100 times shorter than MATLAB. An approach similar to the one presented here has also been used to model physical phenomena in semiconductor devices by solving sets of partial differential equations [9]. Such dynamic systems can have orders of as high as several hundred. This leads to an equivalent electrical network containing several hundred capacitors [9], which can then be solved using SPICE [10].

In some instances, SPICE has been shown to produce more accurate results as well. This has even motivated the interfacing of SPICE with other simulation tools [19]. For example, in one study,

* Accepted 24 November 2010.

SPICE was interfaced with SIMULINK in order to produce more accurate results when simulating a magnetic bearing. It was shown that the SIMULINK results alone were unreliable for the particular system [18].

2. Underlying Concepts

As previously noted, the SPICE solver was designed to handle systems that have even the highest degrees of stiffness. In fact, while it may not appear so on the surface, at its core, SPICE is essentially a robust solver for general ODEs. In this paper, a method is presented that taps this capability, allowing SPICE to handle virtually any linear or nonlinear dynamic system. The basic principles behind this technique are the subject of this section.

The proposed method is based on a state space approach that requires minimal effort from the user. As will be shown, the state space formulation of a dynamic system in SPICE is not only intuitive, it is easy to implement. Furthermore, because it is a state space method, the only operation required is integration. This greatly reduces the complexity of the necessary circuitry since, as it will be shown, integration can be performed using only a three element circuit. Therefore, very little knowledge of circuit analysis is needed to understand the basic principles. While a detailed knowledge of SPICE is not necessary, there are a number of SPICE references available if needed [6–7].

The integrator (Fig. 1(a)) is the fundamental building block of the state variable model. While most versions of SPICE do not support this operation directly, it can be indirectly implemented using the circuit in Fig. 1(b). The same operation could be achieved using an inductor based circuit; however for simplicity, only the capacitive version is discussed here.

In general form, the voltage across a capacitor is expressed as

$$v(t) = v(t_0) + \frac{1}{C}\int_0^t i(\tau)d\tau. \tag{1}$$

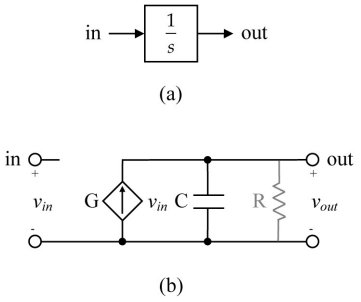

Fig. 1. (a) Ideal integrator block; (b) SPICE equivalent.

Therefore, for the circuit in Fig. 1(b), this results in

$$v_{out} = v(t_0) + \int_{t_0}^t v_{in}(\tau)d\tau. \tag{2}$$

While only the capacitor and current source are required for integration, SPICE will not allow this configuration since the output is considered to be floating. Fortunately, there is a simple way to work around this. By placing a resistor in parallel with the capacitor, the problem is eliminated. The added resistor is shown in grey in Fig. 1(b). Of course, the resulting integrator is no longer ideal, however, for very large resistor values, say $R \geq 100TG\Omega$, the resulting loss is negligible.

The integrator circuit in Fig. 1(b) is implemented in SPICE using the following syntax,

```
1:  C1  0  1  1
2:  R1  0  1  100GIG
3:  G1  0  1  VALUE = {V(in)}
```

Later, when this circuit is used for constructing state-variable models, each integrator will be associated with a unique state variable. Thus the element names and node numbers will match their corresponding state variables.

From here, the remaining details are perhaps best explained through example.

3. Second order dynamic systems

Let us begin with the simulation of a free falling mass. Suppose a plane traveling at altitude h with velocity v releases a payload volume vol and mass m. What will be the time and distance traveled at impact?

The trajectory of the payload is characterized by its acceleration a and velocity v in both the vertical and horizontal directions. That is,

$$a_x = \frac{kv_x^2}{\rho}, \text{and}$$
$$a_y = g - \frac{kv_y^2}{\rho} \tag{3}$$

where ρ is the density of the payload and k is the coefficient of friction. Solving (3) is equivalent to solving a pair of second order systems, one for the x direction and one involving the y direction.

Next, an equivalent state space representation of the systems must be found. Since each system is second order, four state variables are required, which correspond to the displacement in m and velocity in m/s for both dimensions.

Thus the following substitutions are made: $v_1 = x, v_2 = v_x, v_2 = y$ and $v_4 = v_y$. Now the systems are rewritten in terms of the assigned variables,

```
 1:  * Source Payload
 2:  .PARAM k=3.24 g=9.81 m=100 vol=0.1 rho={m/vol}
 3:  * Capacitors for integration
 4:  C1 0 1 1
 5:  C2 0 2 1
 6:  C3 0 3 1
 7:  C4 0 4 1
 8:  * Eliminate floating nodes
 9:  R1 0 1 100GIG
11:  R2 0 2 100GIG
12:  R3 0 3 100GIG
13:  R4 0 4 100GIG
14:  * State equations
15:  G1 1 0 VALUE={-V(2)}
16:  G2 2 0 VALUE={(k*V(2)**2/rho)}
17:  G3 3 0 VALUE={-V(4)}
18:  G4 4 0 VALUE={(g-k*V(4)**2/rho)}
19:  * Initial conditions
20:  .IC V(1)=0 V(2)=100 V(3)=300 V(4)=0
21:  .TRAN 1e-3 10 0 0.1 UIC
22:  .PROBE
23:  .END
```

FOUR VARIABLE HEADER

V(1) = v_1, horizontal position
V(2) = v_2, horizontal velocity
V(3) = v_3, vertical position
V(4) = v_4, vertical velocity

Fig. 2. Listing for the projectile trajectory problem.

$$\begin{aligned}\dot{v}_1 &= -v_2\\ \dot{v}_2 &= \frac{kv_2^2}{\rho}\\ \dot{v}_3 &= -v_4\\ \dot{v}_4 &= g - \frac{kv_4^2}{\rho}.\end{aligned} \tag{4}$$

With the systems in state variable form, the task of generating the listing shown in Fig. 2 is straightforward. The process can be handled in a few simple steps:

1. (Line 2) Define any necessary constants using the .PARAM command.
2. (Lines 4–7) Assign a 1F capacitor for each state variable to be integrated. For this particular system, there are four state variables, which require four capacitors.
3. (Lines 9–13) If necessary, add 100 GΩ resistors in parallel with each capacitor in order to avoid floating node errors.
4. (Lines 14–18) Implement each state equation in Equation (4) using a separate dependant current source.
5. (Line 20) Specify any non-zero initial conditions using the .IC command. Here, an initial state $v = 0$ is assumed, meaning no initial velocity or displacement.
6. (Line 21) Define the parameters for simulation using the .TRAN command.

The syntax for the .TRAN command is

.*TRAN* {print step value}{final time}{initial print time}{step ceiling value}

The print step value determines how often points are stored for plotting. The final time determines the length of the simulation. The initial print time denotes the first value to be stored and the step ceiling value determines the maximum allowable time step for integration.

Because the syntax for the integrating elements is independent of all but the order of the system, they can be viewed as a sort of standardized header, as indicated by the shaded box in Fig. 2. Therefore, this portion of the listing is omitted in the remaining examples due to its trivial nature.

Assuming

$$m = 100\ kg,\ v_0 = 100\frac{m}{s},\ vol = 0.1\ m^3,$$
$$\rho = \frac{m}{vol} = 1000\frac{kg}{m^2},\ h = 300m \text{ and } k = 3.24$$

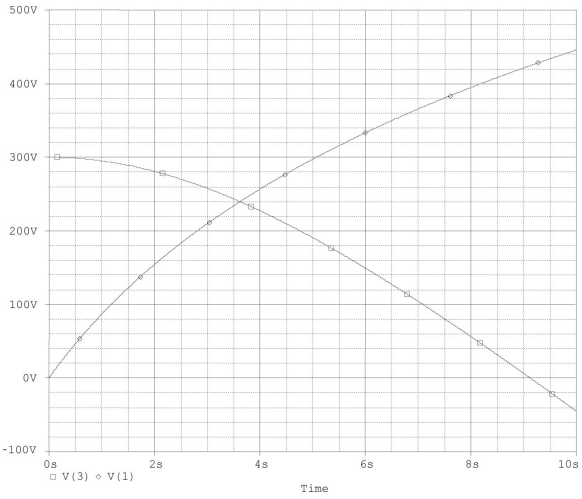

Fig. 3. Simulation of the projectile trajectory. The state variables

the simulation of this system produces the results shown in Fig. 3. Here, the trajectory of the projectile is represented in two parts, with v_1 and v_3 plotted separately vs. time. Note that in the SPICE plot in Fig. 3, the state variables v_1 and v_3 are represented by the voltages V(1) and V(3) respectively. It is apparent from the plot that v_3 reaches 0 at approximately 9.6 s, at which time v_1 is roughly equal to 422 m.

4. Higher order systems

Now, consider the system of coupled mechanical oscillators in Fig. 4. Skipping the details of the derivation, the resulting state equations are found to be

$$\blacksquare(\text{"v"}'_{\downarrow}1\&=\&\text{"v"}_{\downarrow}2,\\ @\text{"v"}'_{\downarrow}2\&=\&-k_{\downarrow}1\ \text{"v"}_{\downarrow}1/m_{\downarrow}1-k_{\downarrow}2\ (\text{"v"}_{\downarrow}1-\text{"v"}_{\downarrow}3)/m_{\downarrow}1-b_{\downarrow}1\ \text{"v"}_{\downarrow}2/m_{\downarrow}1-b_{\downarrow}2\ (\text{"v"}_{\downarrow}2-\text{"v"}_{\downarrow}4)/m_{\downarrow}1,\\ @\text{"v"}'_{\downarrow}3\&=\&\text{"v"}_{\downarrow}4, \qquad (5)$$

where v_1, v_2 and v_5 are the mass positions in meters, and v_2, v_4 and v_6 are the respective velocities in m/s. From here, the netlist is composed in the same manner as in the previous example. The completed listing is shown in Fig. 6. Here, the input stimulus u is implemented using the DC voltage source on line 28. Note that in SPICE a statement can be made to span multiple lines by placing a '+' symbol in front of each subsequent line. This is especially useful when dealing with long lists of parameters or equations like the ones shown here. Note also that the syntax for the integrating capacitors and resistors has been omitted as was explained in the previous example.

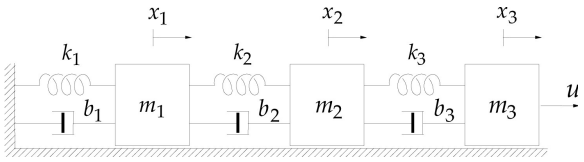

Fig. 4. System of three coupled oscillators.

For this simulation, parameter values of $u = 5$ N, $m_1 = 1$ kg, $m_2 = 500$ kg, $m_3 = 1000$ kg, $k_1 = k_2 = k_3 = 5$ and $b_1 = b_2 = b_3 = 40$ are chosen, with initial conditions $v = 0$. The resulting response is presented in . Though the relatively large size of m_1 results in a correspondingly large time constant, after around 350 s the system appears to settle at $v_1 = 1$, $v_2 = 2$, $v_3 = 3$.

In order to verify the results for this problem, the system was also simulated in MATLAB using the recommended ODE45 solver [15]. The MATLAB results are plotted in Fig. 7. It is evident upon inspection that both methods produce matching results. Although it produces the same result, the MATLAB simulation is more complicated to set up since it requires two separate files to be written. In addition to the main script that initiates the solver, a separate file must be written that defines the system. SPICE, in contrast, handles everything in a single netlist.

It should also be noted that the time required to simulate the system in MATLAB using the ODE45 solver is 7.5 seconds, whereas in SPICE, the same simulation requires only 0.116 seconds. While this may not impose a major inconvenience to the user, it

```
 1:  * Source Coupled Oscillators
 2:  .PARAM m1=1000 m2=500 m3=10
 3:  + k1=5    k2=5   k3=5
 4:  + b1=40   b2=40 b3=40
  ⋮  SIX VARIABLE HEADER
17:  * State equations
18:  G1    0   1   VALUE={V(2)}
19:  G2    0   2   VALUE={-k1*V(1)/m1-k2*(V(1)-V(3))/m1-
20:  + b1*V(2)/m1-b2*(V(2)-V(4))/m1}
21:  G3    0   3   VALUE={V(4)}
22:  G4    0   4   VALUE={-k2*(V(3)-V(1))/m2-k3*(V(3)-
23:  + V(5))/m2-b2*(V(4)-V(2))/m2-b3*(V(4)-V(2))/m2}
24:  G5    0   5   VALUE={V(6)}
25:  G6    0   6   VALUE={-k3*(V(5)-V(3))/m3-b3*(V(6)-
26:  + V(4))/m3+V(u)/m3}
27:  * Stimulus
28:  Vu    u   0   DC 5V
29:  * Initial conditions
30:  .IC  V(1)=0, V(2)=0, V(3)=0,
31:  + V(4)=0, V(5)=0, V(6)=0
32:  .TRAN 1e-3 400s 0ms 10e-3 UIC
33:  .PROBE
34:  .END
```

V(1) = v_1, position of m_1
V(2) = v_2, velocity of m_1
V(3) = v_3, position of m_2
V(4) = v_4, velocity of m_2
V(5) = v_5, position of m_3
V(6) = v_6, velocity of m_3

Fig. 5. Listing for the coupled oscillator problem.

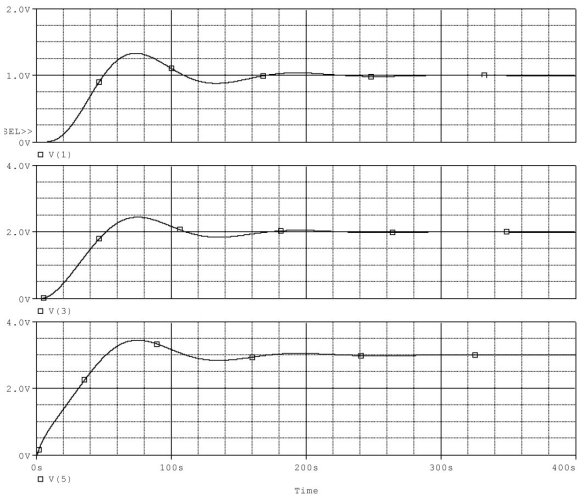

Fig. 6. Simulation of three damped coupled oscillators. The displacements v_1, v_3, and v_5, measured in meters, are represented by the voltages V(1), V(3) and V(5) respectively.

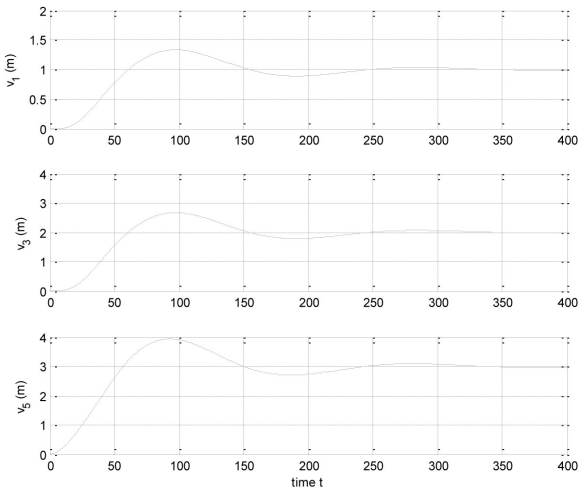

Fig. 7. MATLAB verification for the coupled oscillator problem.

still represents a 6400% difference in simulation time. A more extreme example is discussed in the next section, for which the MATLAB simulation time becomes a significant issue.

5. Stiff systems

One area in which SPICE outperforms MATLAB is the simulation of stiff dynamic systems. This is because of the relatively high degrees of stiffness commonly encountered in the electronic systems for which it was designed. This ability can be extended to more general systems using the proposed method. Here, it is used to simulate a stiff system from the field of chemical kinematics [16]. The problem involves determining the chemical concentrations of four reactants in a long chemical reaction. The form of the system is given by:

$$\begin{aligned}\dot{v}_1 &= -Av_1 - Bv_1v_2\\ \dot{v}_2 &= Av_1 - MCv_2v_2\\ \dot{v}_3 &= Av_1 - Bv_1v_2 - MCv_2v_3 + Cv_4\\ \dot{v}_4 &= Bv_1v_2 - Cv_4\end{aligned} \tag{6}$$

where $A = 7.89 \times 10^{-10}$, $B = 1.1 \times 10^7$, $C = 1.13 \times 10^3$ and $M = 10^6$. The equivalent netlist is shown in Fig. 8.

Because the reaction is slow, the system is simulated for t = 0 to 10^7 s. The SPICE simulation for this system required 2 minutes and 12 seconds to complete. The simulation results are shown in Fig. 9. In contrast, the same system, when simulated in MATLAB using the ODE45 solver, would require more than 5 years to complete. This is estimated by measuring the time required for MATLAB to simulate the system over a 100 s period and then extrapolating to 10^7 s Not only is this simulation time unacceptable, but using the default maximum step size causes the algorithm to become unstable, which produces inaccurate results. In order to eliminate this instability, the maximum step size must be adjusted by trial and error using the 'odeset' function. Of course this effort is futile anyway, since even with a properly selected maximum step, the necessary simulation time remains prohibitively long.

In the interest of fairness, it should be noted that the system can be successfully simulated in MATLAB using one of its more sophisticated solvers. In

```
 1:  * Source Chemical Kinematics
 2:  .PARAM A=7.89e-10 B=1.1e7 C=1.13e3 M=1e6;
  :  FOUR VARIABLE HEADER
11:  * State equations
12:  G1    0   1   VALUE={-A*V(1)-B*V(1)*V(3)}
13:  G2    0   2   VALUE={A*V(1)-M*C*V(2)*V(3)}
14:  G3    0   3   VALUE={A*V(1)-B*V(1)*V(3)-M*C*V(2)*V(3)+C*V(4)}
15:  G4    0   4   VALUE={B*V(1)*V(3)-C*V(4)}
16:  * Initial conditions
17:  .IC  V(1)=1.76e-3, V(2)=0, V(3)=0, V(4)=0
```

V(1) = v_1, concentration of reactant 1

V(2) = v_2, concentration of reactant 2

Fig. 8. SPICE listing for the chemical kinematics problem.

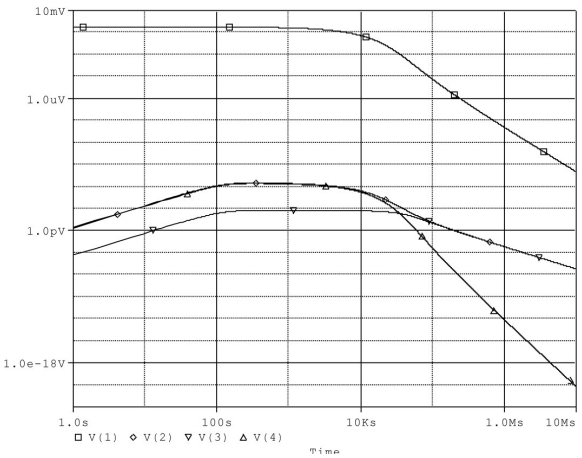

Fig. 9. SPICE results for the chemical kinematics problem.

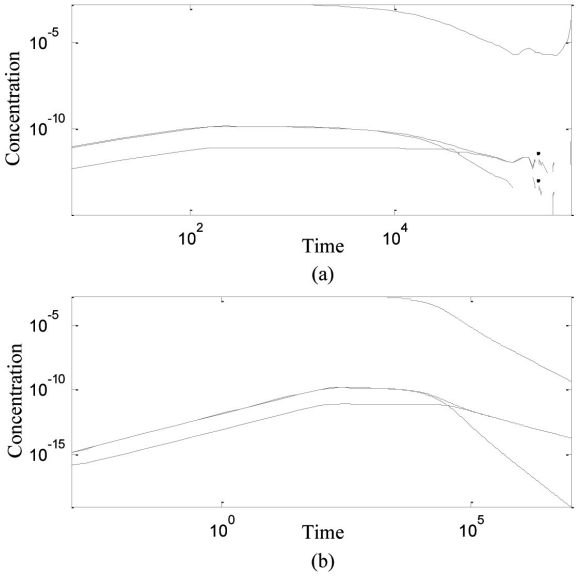

Fig. 10. MATLAB results for the chemical kinematics problem using (a) the default settings and (b) the 2nd order Gear Method.

fact, using the ODE15s solver [15], it is possible to produce results that are nearly identical to SPICE. ODE15s is a variable order solver that can be configured to use the same backward differentiation formula, also known as Gear's method, which is used by SPICE [6]. To do this, however, the student must first recognize that the simulation may be unstable and that the time step has become inordinately small. He must then also recognize that the issue is related to the stiffness of the system, and be familiar enough with the MATLAB solvers to switch to ODE15s. For this particular problem the ODE15s solver is also unstable when using the default settings, as shown in Fig. 10(a). Therefore, the student must know that they must reduce the maximum order of the differentiation formula in order to stabilize the algorithm. Finally, in order to improve the accuracy of the solution, the student would need to turn on the backward differentiation formulas mentioned above. Once these changes have been made, the net result is the same 2nd order Gear method that SPICE uses by default. This is verified by the solution shown in Fig. 10(b), which matches the SPICE simulation from Fig. 9.

6. Simulation of chaotic systems

Now we turn our attention to an example from the field of chaos theory, known as the Lorenz attractor [8]. While this system is deceivingly simple in appearance, in reality, its behavior is surprisingly complex. The term chaos, when applied to dynamic systems, implies an abnormally high level of sensitivity to initial conditions. For these systems, even minimal changes in the initial state can result in wildly different trajectories. Of all such systems, the Lorenz attractor is perhaps the most widely documented, and has become an icon of sorts in the field. Its popularity is due in large part to its simplicity, as well as the aesthetic nature of its solution. In this example SPICE is used to demonstrate its unique behavior [8].

As always, the system must first be expressed in state variable form. Here, we have a relatively simple third order nonlinear system described by the following equations, commonly known as the Lorenz equations:

$$\begin{aligned} \dot{v}_1 &= \sigma(v_2 - v_1) \\ \dot{v}_2 &= v_1(\rho - v_3) - v_2 \\ \dot{v}_3 &= v_1 v_2 - \beta v_3 \end{aligned} \tag{7}$$

The parameters σ, ρ, and β are known respectively as the Prandtl number, the Rayleigh number, and the physical proportion. The classical examples of this system use values of $\sigma = 10$, $\rho = 10$, and $\beta = \frac{8}{3}$. Also, because there is no external stimulus, the system must be excited using a set of nonzero initial conditions. The resulting netlist is shown in Fig. 11.

Since the goal is to observe and verify the system's chaotic nature, the results are simulated and compared using two separate initial conditions: v0 = [–8, 8, 27] and v0 = [–7.99, 8, 27]. The results for both initial conditions are shown in Fig. 12. Notice how quickly the two trajectories deviate. For the first six seconds, the two solutions appear quite similar. However, by the eight second mark, they bear almost no resemblance. This illustrates the sensitive dependence on initial conditions mentioned previously. A phase portrait for the first solution is shown in Fig. 13. Notice also how the state trajectory seems to oscillate randomly between two separate basins of attraction.

```
 1:  * Source Lorenz Attractor
 2:  .PARAM sigma=10, rho=28, beta={8/3}
  ⋮  THREE VARIABLE HEADER
11:  G1 0 1 VALUE={sigma*(V(2)-V(1))}
12:  G2 0 2 VALUE={V(1)*(rho-V(3))-V(2)}
13:  G3 0 3 VALUE={V(1)*V(2)-beta*V(3)}
14:  .IC V(1)=-8, V(2)=8, V(3)=27
15:  .TRAN .01ms 20s 0 .1ms UIC
16:  .PROBE
17:  .END
```

V(1) = v_1
V(2) = v_2
V(3) = v_3

Fig. 11. SPICE netlist for the Lorenz attractor.

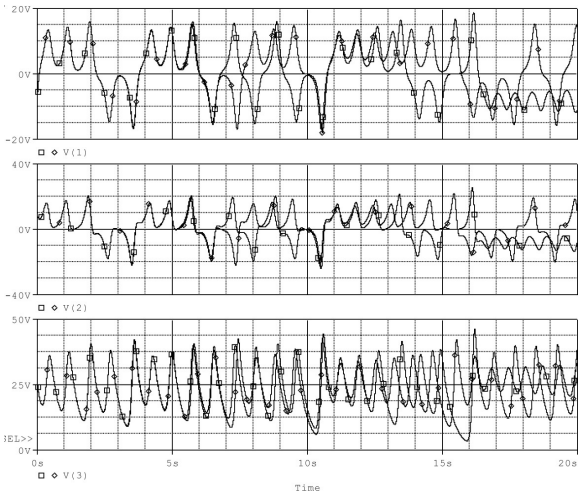

Fig. 12. Transient response of the Lorenz attractor.

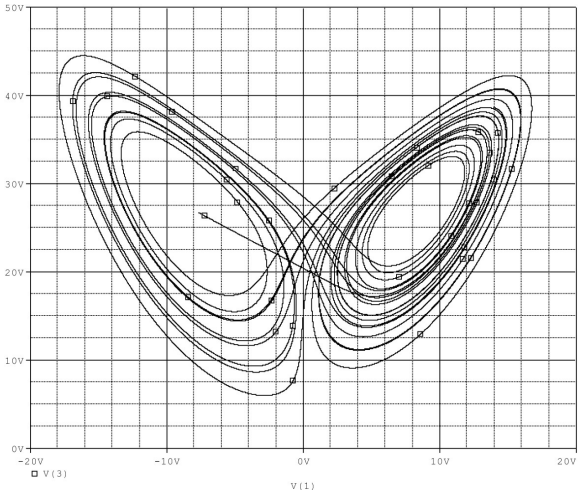

Fig. 13. State trajectory of the Lorenz attractor.

7. Discussion

The proposed method is not intended as a replacement for the more commonly used tools such as MATLAB and Simulink; however, as illustrated in the preceding examples, the SPICE method has some notable advantages that might make it a useful supplement for students.

First, the software is available for free, which means the students can use it on their own computers. This may be useful to students who would like to work at home or when on-campus labs are crowded. This also allows the student to pursue topics further on their own time.

Another advantage of SPICE is stability. Unlike MATLAB, where the stability of a simulation can be highly dependent on the choice of the solver, SPICE uses the robust 2nd order Gear method by default, which is known to be astable [17]. This eliminates the need for any advanced knowledge of numerical methods by the user. This would certainly be useful for students who may be encountering dynamic systems for the first time. This advantage was illustrated in the chemical kinematics example.

In many cases, SPICE may also have the advantage of shorter computation time, as illustrated in both the coupled oscillator example and the chemical kinematics example. This is because, in addition to being highly stable, the SPICE solver was designed to handle very stiff systems. When applied to stiff systems, many common integration schemes can be extremely inefficient due to the reduction of the time step. This is also the case with the ODE45 solver, which is the standard solver used in MATLAB. Although these systems can be successfully simulated in MATLAB using one of the more sophisticated solvers, this too requires a somewhat advanced level of knowledge and intuition from the user, as illustrated in Section 5.

8. Conclusion

It has been shown that the algorithms employed by SPICE for handling differential equations describing electronic circuits are also useful for solving more general classes of systems. Owing to the relatively high degrees of stiffness encountered in many electronic systems, these algorithms are especially powerful and effective over a very wide range of systems. Using the presented method, nearly any dynamic system that can be described as a set of state equations can be simulated in SPICE. Further-

more, while other software packages such as MATLAB may require additional knowledge from the user in selecting the best solver for a given problem, the versatility of the SPICE solver does not require the user to bother with the details of the underlying algorithm. This offers an obvious benefit for students who may be more interested in the meaning of the results than in the numerical methods by which they are generated and, practically speaking, a detailed knowledge of numerical methods may be beyond the scope of many undergraduate engineering programs.

References

1. J. L. Suthar, J. R. Laghari and T. J. Saluzzo, Usefulness of SPICE in high voltage engineering education, *IEEE Trans. Power Systems*, **6**, 1991, pp. 1272–1278.
2. L. V. Hmurcik, M. Hettinger, K. S. Gottschalck and F. C. Fitchen, SPICE applications to an undergraduate electronics program, *IEEE Trans. Educ.*, **33**, 1990, pp. 183–189.
3. W. L. Brown, A. Y. J. Szeto, Reconciling Spice results and hand calculations: Unexpected problems, *IEEE Trans. Educ.*, **43,** 2000, pp. 43–51.
4. L. H. Herbert, Power Electronics Instruction in the USA and Canada: Topics, Curricula, and Trends, *Int. J. Eng Educ.*, **14**, 1998, pp. 282–288.
5. B. M. Wilamowski, O. Kaynak, Oil well diagnosis by sensing terminal characteristics of the induction motor, *IEEE Trans. on Industrial Electronics*, **47**, 2000, pp. 1100–1107.
6. B. M. Wilamowski and R. C. Jaeger. *Computerized Circuit Analysis Using SPICE Programs*, The McGraw-Hill Companies, Inc., Dubuque, Iowa, 1997.
7. A. Vladimirescu, *The Spice Book*, 2nd Edn, John Wiley & Sons, Inc., 2010.
8. E. N. Lorenz, Deterministic nonperiodic flow, *J. Atmos. Sci.*, **20**, 1963, pp. 130–141.
9. B. M. Wilamowski, Z. J. Staszak and R. H. Mattson, An electrical network approach to the analysis of semiconductor devices, *IEEE Trans. Educ.*, **35**, 1992, pp. 144–152.
10. W. R. Zimmerman, Time domain solutions to partial differential equations using SPICE, *IEEE Trans. Educ.*, **39**, 1996, pp. 563–573.
11. LTspice IV Downloads and Updates, http://www.linear.com/designtools/software/ltspice.jsp, Accessed 2 November 2010.
12. PSPICE 9.1 Student Version, http://www.electronics-lab.com/downloads/schematic/013/, Accessed 2 November 2010.
13. B. Wilamowsk , J. Regnier and A. Malinowski, SIP-Spice Intranet Package, *IEEE International Symposium on Industrial Electronics, ISIE '98*, Pretoria, July 1998, (August 2002), pp. 192–195.
14. The SPICE Page, http://bwrc.eecs.berkeley.edu/classes/icbook/spice/, Accessed 4 November 2010.
15. Solve Initial Value Problems for Ordinary Differential Equations, http://www.mathworks.com/help/techdoc/ref/ode23.html, Accessed 4 November 2010.
16. W. H. Enright, T. E. Hull and B. Lindberg. Comparing numerical methods for stiff system of ODEs. *BIT*, **15**, 1975, pp. 10–48.
17. Create or alter options structure for ordinary differential equation solvers, http://www.mathworks.com/help/techdoc/ref/odeset.html, accessed 12 November 2010.
18. C. G. Wilson and J. Y. Hung, A system technique combining SPICE and SIMULINK Tools, *IECON '10. 36th Annual Conference of IEEE Industrial Electronics Society*, Phoenix, AZ, 7–10 November 2010.
19. M. Madbouly, M. Dessouky, M. Zakaria, R. Latif and A. Farid, MATLAB–SPICE interface (MATSPICE) and its applications, *Proceedings of the 15th International Conference on Microelectronics, ICM '03*, (December 2003), pp. 37–40.

Joel D. Hewlett received his BS and MS in electrical engineering from Auburn University, Auburn, AL, where he is currently working toward his PhD in electrical and computer engineering. He is a Research Assistant with the Department of Electrical and Computer Engineering, Auburn University. He was a Systems Analyst with Delta Research, Inc., Huntsville, AL. His research interests include intelligent systems, neural networks, numerical optimization, evolutionary computation and control systems. He is a Reviewer for the *IEEE Transactions on Industrial Electronics*.

Bogdan M. Wilamowski received his MS in computer engineering, his PhD degree in neural computing, and the Dr Habil. degrees in integrated circuit design in 1966, 1970, and 1977, respectively. He is currently the Director of the Alabama Micro/Nano Science and Technology Center and a Professor with the Department of Electrical and Computer Engineering. Professor Wilamowski is the author of four textbooks and about 300 refereed publications and is the holder of 28 patents. He was the Major Professor for over 150 graduate students. His main areas of interest include computational intelligence and soft computing, computer-aided design development, solid-state electronics, mixed- and analog-signal processing, and network programming. Dr Wilamowski was the Vice President of the IEEE Computational Intelligence Society and the President of the IEEE Industrial Electronics Society. He also was the Editor-in-Chief of the *IEEE Trans. on Industrial Electronics* and is currently the Editor-in-Chief of the *IEEE Trans. on Industrial Informatics*.