Overview of SPICE-like circuit simulation algorithms

K.G. Nichols
T.J. Kazmierski
M. Zwolinski
A.D. Brown

Indexing terms: SPICE, Simulation algorithms, Circuit level simulation

Abstract: A tutorially orientated account of the principles underlying circuit level simulation forms the basis for discussing reliability of circuit level simulation. An outline of equation formulation and time discretisation of the equations is followed by consideration of stability of the numerical integration methods. The precision of simulation, which also depends on the integration methods used, is then discussed, with particular reference to the analysis parameters of the SPICE simulator. A descriptive account of the Newton–Raphson method of solving the nonlinear algebraic equations, derived from the discretised differential equations, is then presented again with relation to SPICE parameters. The reasons for convergence failure in both transient and DC simulation are related to the underlying principles of simulation and possible ill conditioning of the circuit. The paper includes some simple simulation examples of problematic situations.

1 Introduction

Four principal issues affect the reliability of circuit level simulation. They are numerical stability of integration, precision of results at each time step, convergence at each time step, and adequacy of device modelling. The latter is not a topic considered in this paper. Any discussion of circuit level simulation needs to be related to the ubiquitous SPICE program and so, throughout this paper, upper case strings in parentheses refer to SPICE control parameters and any textual reference is to SPICE version 3E2 [1]. Earlier versions might not necessarily be algorithmically equivalent to 3E2 in all details. The authors are aware that not all copies of version 2G6 are identical.

The method underlying simulation is first, the formulation of a coupled set of nonlinear first-order differential algebraic equations representing the behaviour of the interconnected set of devices comprising the circuit [2, 3].

The second step is the replacement of the time derivatives in the differential equations by finite difference approximations (known as integration formulas) which discretise time, in general, in a nonuniform way. This step transforms the nonlinear differential equations, at each discretised time point, into a time independent set of nonlinear equations.

The third step is to solve the nonlinear equations, at each discretised time, by the Newton–Raphson technique which approximates them with a linear set of equations based on an initial estimate of the solution. Repeated solution of the linear set, with relinearisation of the equation set after each iteration, is used to refine the solution estimate until it is deemed that adequate precision has been achived (RELTOL, ABSTOL, VNTOL).

The converged solution of the Newton–Raphson process is then tested (TRTOL) as an adequate solution of the differential equation set for this time point. Action, as described later, is necessary when the Newton–Raphson iterations fail to converge to a solution, or convergence is attained but the circuit solution is too imprecise or is numerically unstable. The action taken normally involves cutting the time increment and repeating the solution process at the same time point.

Having arrived at a satisfactory solution at the time point, a trial time increment to advance to a new point is selected, and a prediction of the solution at the new point is made. This prediction is used as the initial estimate of the solution for the Newton–Raphson process at the new time point. Thus simulation proceeds as a march-in-time through a sequence of discretised time points selected to achieve both convergence of the Newton–Raphson process and adequate precision of simulation. This implies that a satisfactory initial estimate of the circuit state is available at the start of the simulation. Obtaining this is known as the DC solution. Often, this is the most demanding problem in circuit simulation, and methods of solving it are discussed in this paper.

For a given circuit, numerical stability of integration depends on the integration formula being used and the size of the discretised time step; in adverse situations, convergence failure can occur. Precision problems can arise because of too coarse granularity of time discretisation, for a given integration formula, or because of limited floating-point number precision in determining convergence of the Newton–Raphson process, or because of near singularity of the coefficient matrix of the linear equation set. The last of these can also contribute to convergence problems, but the main cause of convergence

The authors are indebted to Dr. J.T. Lin who illuminated some of the more opaque sections of SPICE code by painstaking reverse engineering.

Paper 1246G (E10), first received 23rd November 1993 and in revised form 9th March 1993

The authors are with the Department of Electronics and Computer Science, University of Southampton, United Kingdom

failure is an inappropriate estimate used to initiate the Newton–Raphson iterative process. These are the issues addressed in this contribution.

2 Formulating the discrete time nonlinear equations

Different simulators use different formulations of the circuit equations. For example, SPICE uses the modified nodal [4] formulation and ASTAP the sparse tableau [5] formulation. Whichever formulation is used, the generic result is a coupled set of nonlinear first-order differential algebraic equations of the form

$$f(x(t), x'(t), t) = 0 \tag{1}$$

Here, $x(t)$ is the vector of circuit variables (e.g. node voltages and charges or branch currents), $x'(t)$ is the derivative of this vector, with respect to time t, and $f(\cdot)$ is a nonlinear vector function. The finite difference expression (differentiation formula [6, 7] to be used to approximate the time derivatives has the form

$$x'(t_n) \equiv x'_n = (1/h_n) \sum_{r=0}^{k} a_r x_{n-r} \tag{2}$$

where h_n is the time step in advancing to discretised time t_n from discretised time t_{n-1}, and the coefficients a_r are rational functions of h_n and up to k of the preceding values of time step. Substitution for x' in eqn. 1 yields

$$f\left(x_n, (1/h_n) \sum_{r=0}^{k} a_r x_{n-r}, t_n\right) = 0 \tag{3}$$

Thus, the nonlinear differential equations, at each discretised time point, have been transformed into a time independent set of nonlinear equations:

$$F(x) = 0 \tag{4}$$

where $x \equiv x_n$ and $F(\cdot)$ is a nonlinear function readily derived from $f(\cdot)$. Before considering the numerical solution of this last equation, in the next Section, the stability of the numerical integration of the discrete time point approximation to the differential equation will be reviewed.

3 Numerical integration

The stability of a nonlinear differential system is most easily addressed by considering its state equation $\xi' = S(\xi, t)$, where the ξ is the state vector, and $S(\cdot)$ is a nonlinear function [8]. In an electric circuit, the components of the state vector are an independent, but complete, subset of its capacitor charges and inductor fluxes. The circuit equation set eqn. 1 can always be translated [9] into a state equation form but, for a nonlinear circuit, possibly only on a small neighbourhood of any value of x.

When linearised on a small neighbourhood of ξ, the state equation is $\xi' = A\xi + Bu(t)$, where A and B are constant matrices and $u(t)$ represents the external excitations. Deductions concerning the stability of the system represented by the state equation, and the stability of numerical integration of discrete time approximations of the equation, are normally obtained by transforming [10, 12] the state vector ξ to the normal mode vector η. When the normal modes (eigenvalues of A) of the system are distinct, this effects a corresponding transformation of the matrix A to a diagonal form with elements equal to the eigenvalues of A. The ξ to η transformation matrix in this case comprises the set of corresponding eigenvectors. When some normal modes are not distinct (i.e. degenerate with multiplicity), the transformed A matrix is not quite diagonal [11] but its diagonal elements are still the eigenvalues, with each nondistinct one appearing with its degenerate multiplicity. These normal forms simplify the system differential equation, by decoupling its component equations and thus allowing their separate integration, which makes the numerical stability analysis of integration formulas easier. For more details of this analysis, the reader is referred elsewhere [8, 10].

In the absence of external excitations, for a stable dissipative system, both ξ and $\eta \to 0$ as time increases. Analysis of the kind described in the preceding paragraph then shows that, for system stability of $\xi' = A\xi$, the eigenvalues of A must lie in the left half complex plane. The real part of an eigenvalue is related to a reciprocal time constant of the system, and the imaginary part is a normal mode frequency; complex eigenvalues occur in complex conjugate pairs.

The use of a differentiation formula replaces ξ' with a finite difference approximation, effectively discretising time, which alters the region of stability in the complex plane so that, in general, it is no longer the left half plane and, further, in a way that depends on the time step h. This is illustrated in Fig. 1 by the stable (shaded) regions for the three one-step formulas

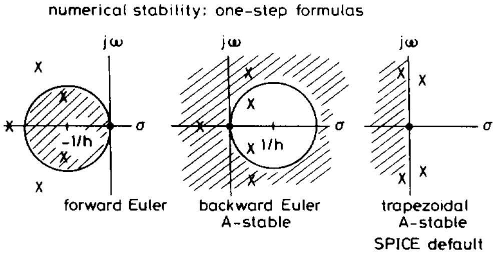


Fig. 1 *Stability, one-step integration formulas*

$$x_n = x_{n-1} + h_n x'_{n-1} \tag{5}$$

$$x_n = x_{n-1} + h_n x'_n \tag{6}$$

$$x_n = x_{n-1} + \tfrac{1}{2} h_n (x'_n + x'_{n-1}) \tag{7}$$

known as the forward Euler, backward Euler and trapezoidal rule, respectively. The first of these is an explicit formula and the last two are implicit formulas in that the expression for x_n does or does not depend on x'_n. In Fig. 1, a real eigenvalue (X) or a complex conjugate eigenvalue pair (X, X) in a shaded region corresponds to stable integration but in a nonshaded region to numerical instability. Thus, for stable numerical integration, all eigenvalues of A must lie in the shaded region.

A differentiation formula which provides for numerically stable integration whenever the system is stable, regardless of the time step h, is said to be A-stable; thus, in Fig. 1, the backward Euler and trapezoidal rule are A-stable formulas, but the forward Euler formula has a very limited stability region unless the time step h is kept so small throughout the simulation as to always include all the eigenvalues of A within the region. For electronic circuits, which normally have eigenvalue magnitudes spanning at least several decades, this leads to impractically long simulations [9]. Systems with a wide spread of eigenvalues are said to be stiff.

In the foregoing, we could have referred to integration formulae rather than differentiation formulas. If eqn. 2 is transposed so that x_n is expressed in terms of $x_{n-r}(0 < r \leqslant s)$ and x'_n, it is called an inegration formula [10]. More generally, a pth-order integration formula is an expression for x_n as a sum of terms in $x_{n-r}(0 < r \leqslant s)$ and in $x'_{n-r}(0 \leqslant r \leqslant s)$ in which a total of just $p + 1$ of the coefficients of the x_{n-r} and x'_{n-r} are nonzero. The formula is explicit if the coefficient of x'_n is zero and implicit otherwise. The coefficients of any particular integration formula are chosed to give the best approximation [6] to the solution of the equation set. The use of a first-order differentiation formula or the corresponding first-order integration formula for the forward and backward Euler methods is equivalent; substituting the differentiation formula for the derivative in the state equation gives the same result as substituting the state equation into the integration formula. The trapezoidal rule, however, is an integration formula with no corresponding form of differentiation formula, since it contains two derivative terms. In practice, eqn. 2 is not substituted into eqn. 1 but rather integration formulas are used at the device level in companion models [8, 12], in which nonlinear characteristics of devices are also linearised (effectively a Newton–Raphson process step), and which are used to build the linear equation set directly. This approach to building the linear equation set has advantages in that convergence and simulation precision can be monitored at the device level.

The default integration formula (METHOD) used in SPICE is the trapezoidal rule. Although, ideally, this rule is A-stable and the region of stability is the left half complex plane, because of linearisation and numerical rounding errors, the boundary of this region becomes somewhat fuzzy. This causes problems in simulation of stable high-Q circuits for which a complex conjugate pair of eigenvalues of A lie close to, but to the left of, the imaginary axis in the complex plane (see Fig. 1). The usual manifestation of the problem is a ringing perturbation, often of some magnitude, on an otherwise correct waveform. Excessive computation can result as a consequence of the reduction in the time step h needed to follow the spurious oscillaton.

SPICE allows an implicit multistep integration formula (METHOD = GEAR)

$$x_n = \sum_{r=1}^{k} \alpha_r x_{n-r} + h_n \beta x'_n \qquad (8)$$

to be used instead of the trapezoidal rule. Here, β is another coefficient like a_r. This formula is one of the general class described in eqn. 2. The region of stability of the two-step second-order Gear formula always includes the whole of the left half complex plane provided [13] the ratio $r \equiv h_n/h_{n-1}$ does not exceed unity; see Fig. 2 which shows the stability regions for r values of 1.0, 1.5 and 2.0. In SPICE, the Gear second-order formula is used with $r = 1$ (when the step size is changed, an initial backward Euler step is taken and then the Gear formula is restarted) and so is A-stable (note [14], there is no A-stable formula of order greater than two). Stable high-Q circuits can therefore be simulated without producing the spurious ringing effect, provided the formula order does not exceed two and the time step expansion ratio is kept less than unity (the latter in simulators other than SPICE). The region of stability of this formula, however, includes some of the right half complex plane. The boundary of the region, while still touching the origin, is bent away from the imaginary axis increasingly as h_n is made larger. In simulation, genuine oscillations are damped out when the eigenvalues of A fall into the region of stability (as indicated by the crosses to the right of the imaginary axis in Fig. 1). Real difficulty arises when it is not known in advance whether a high-Q circuit is a stable amplifier or an oscillatory.

In Fig. 3 the results of simulation of a simple circuit are shown; (a) using the SPICE default trapezoidal rule and (b) using the second-order Gear formula, otherwise

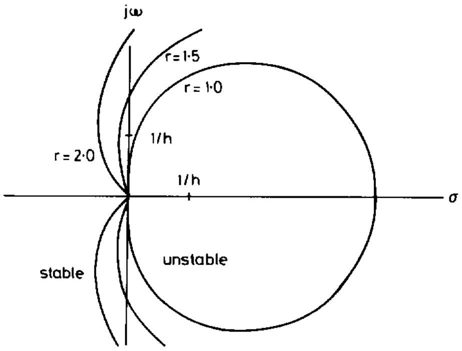
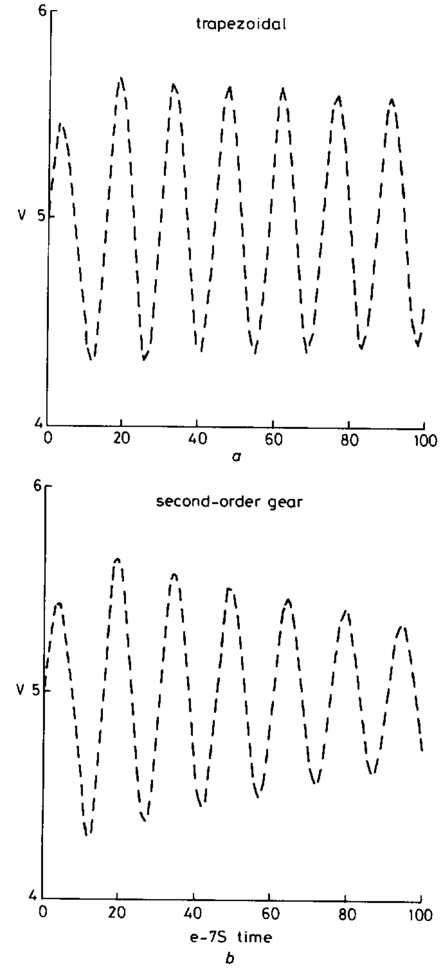


Fig. 2 *Stability, Shichman–Gear second-order formula*



a

b

Fig. 3 *High-Q circuit simulation waveforms*

the circuit and the simulation parameters are identical. The circuit comprised a JFET device in series with a high-Q (about 100) tuned circuit, initially quiescent, and a DC supply. A 10 ns pulse to the gate of the JFET energises the tuned circuit, which rings after the pulse is removed. Note, the tuned circuit is always shunted by the output conductance of the JFET which changes when it is pulsed. Whether the trace in (*a*) is exhibiting some numerical instability or whether the trace in (*b*) is being damped by the Gear formula is a matter of some speculation! Nonlinearity of the JFET model precludes exact analysis of the circuit and hence resolution of this speculation.

4 Precision in simulation

An integration formula is, in effect, a limited number of terms of a Taylor expansion of the solution function with respect to time. The more terms used, the higher the order of approximation and this order is also identified with the integration formula. The remainder of the expansion is referred to as the local truncation error which is expressed as a complexity in terms of a power of the time step h. For example, by performing Taylor expansions of x_n about t_{n-1} and of x_{n-1} about t_n, and with a little manipulation to eliminate the terms in x'_n, the following expression can be obtained

$$x_n = x_{n-1} + \tfrac{1}{2}h_n(x'_n + x'_{n-1}) - \tfrac{1}{12}h_n^3 x'''_n + O(h_n^4) \qquad (9)$$

This is the trapezoidal rule including the local truncation error represented by the last two terms on the right-hand side. The complexity of the error for this rule is therefore $O(h^3)$. The Gear formula of order two also has local truncation error of this order. Higher-order Gear formulas with less truncation error are seldom used because of their inherent numerical instability.

In SPICE, the required maximum local truncation error is specified as, in effect, a multiple (TRTOL) of the appropriate Newton–Raphson convergence requirement (RELTOL, ABSTOL, VNTOL) and also a charge tolerance parameter (CHGTOL) for capacitive elements (SPICE does not appear to have a corresponding parameter for inductive elements). When the test fails for any circuit element, the time step h is cut and the solution at the time point is repeated. When the test is satisfied, the parameters (TRTOL, CHGTOL, RELTOL, ABSTOL and possibly others) are used to calculate a precision-compatible estimate of the next time step for the element. Each calculation is based on a measure of the local truncation error for that element. The actual next time step h used is then the minimum of the calculated time steps of all the elements. This is intended to be the main mechanism of time step control. But note, subsequently, this time step might be rejected if a convergence or precision problem arises at the time point which would then be reworked with a smaller step h.

The initial estimate of the solution at a time point can be obtained by prediction from the solutions at prior time points. Provided the prediction is within the region of convergence of the Newton–Raphson process, iterations of the latter then converge the prediction to the correction (i.e. to an acceptable solution of the differential equation set). A predictor formula is similar to an integration formula but is explicit (i.e. the expression for x_n does not involve a term in x'_n). Provided the predictor is of the same order as that of the integration formula, the difference between the corrected and predicted values can be shown, again by Taylor expansion manipulations, to have the same complexity as the local truncation error (e.g. $O(h^3)$), and can therefore be used as an estimate of the latter. Such an estimate is used both to control the size of the next time step, and also to check the precision of simulation after the step.

The mechanism described in the preceding paragraph is illustrated in Fig. 4 for a one-dimensional example using a first-order predictor (e.g. the forward Euler) and, by implication, a first-order integrator (e.g. the backward Euler). In this Figure, it is assumed, for simplicity of explanation, that an exact solution has been obtained at point 1 on the time axis and that a next time step h has already been selected by some means. The slope of the solution at the point is used to predict the solution P at the next time point 2. The prediction, which is within the region of convergence of the Newton–Raphson process (indicated by range bars in the figure), is used as the initial estimate, and so the solution converges to the correction C. The difference between the actual solution (on the curve) and C is the local truncation error, but the curve is not known, so, that between C and P is used as the measure of local truncation error and, this being within the prescribed simulation precision (TRTOL), the solution is accepted. Had the error been too large, the time step h would have been cut, and the solution at a revised time point (after point 1) re-evaluated, repeatedly if necessary. In either event, the measure of truncation error and the prescribed simulation precision are used to evaluate a suitable value for the next time step h. In some simulators, the next time step is not reevaluated unless either the time step must be cut or the solution at the point is deemed far too precise. In Fig. 4, h is unchanged for the next step. Having decided h, the prediction P at time point 3 is made and is within the region of convergence, and so the correction C is obtained.

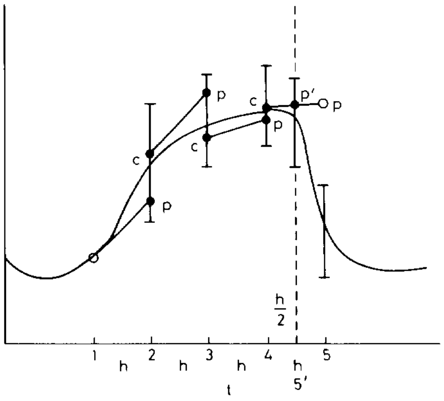


Fig. 4 *March-in-time simulation, predictor–corrector*

The procedure just described is repeated to obtain the prediction P and correction C at time point 4 but at point 5, because the solution curve has dropped markedly, the prediction P is outside the region of convergence, so the time step is halved and a new prediction P' made, which happens to be within the region of convergence at the new, closer, time point 5'. In actuality, the solution is a multidimensional vector, and so some magnitude or norm of the difference between P and C must be used.

In SPICE, the implementation of this method is an option (by conditional compilation) available only for the

Gear formula, otherwise the estimation of error is based on divided differences (to estimate x'''; see Fig. 5 which indicates the first few steps in this process) of solutions at prior time points. Also, without the option, the initial estimate of the solution is taken as the converged solution at the last time point. There is some experimental evidence [15] that the latter strategy requires fewer iterations in total, in the simulation, for predictions can be way off target when, with the larger time steps, a fast changing waveform suddenly levels out. Fig. 6 stresses this point; it might be better to use the solution x_{n-1} to start the Newton–Raphson process at t_n rather than the prediction at P.

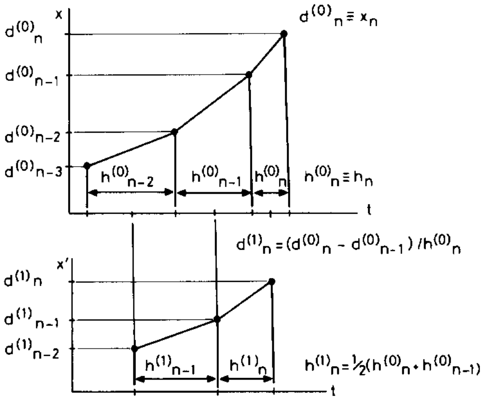


Fig. 5 *Differential coefficients by divided difference method*

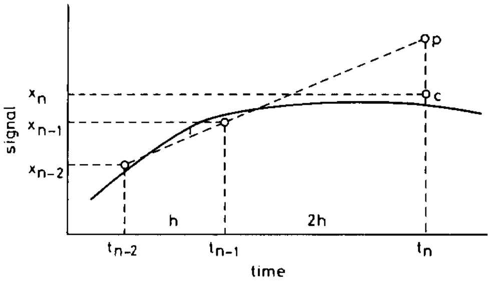


Fig. 6 *Initial estimate, prediction versus last point*

Truncation errors can accumulate over present and past time steps; in Fig. 4, the error at time point 3 appears to be the sum of the local truncation errors at points 2 and 3. With stable numerical integration, however, the local truncation error generated at a step decays with further steps. The rate of decay is related to circuit time constants associated with its acquisition. The local truncation error propagates along with the solution in a way comparable to the response of the circuit to a pulse. When solution waveforms are not changing rapidly, and the time step is long, the local truncation errors more or less decay away completely between adjacent time points, and the total error at a time point is reasonably represented by the local truncation error. When, however, a waveform changes rapidly, the time step shortens and local truncation errors from previous time points have not decayed before more such error is accumulated at the present time point. In fast transients, the total error in a waveform can be as much as an order of magnitude larger than the local truncation error. In logic circuits this is manifested as an error in propagation delay; see Fig. 7. Repeated simulations, at higher precision, are needed to check that simulated delays have not changed significantly.

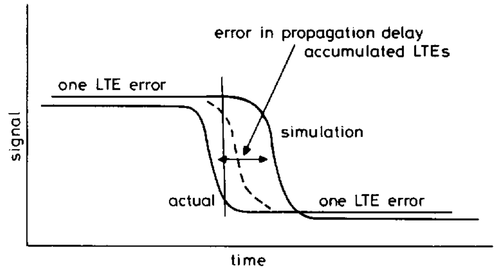


Fig. 7 *Error in propagation delay from accumulated LTEs*

SPICE disguises the foregoing problem to some extent by ignoring decay of local truncation error, and by using an error rate [14] of complexity $O(h^3)/h = O(h^2)$ to control the time step h. Thus, for the trapezoidal rule, the local truncation error rate is taken as

$$LTE = (\tfrac{1}{12})h^2 x_n''' < e_{max} \tag{10}$$

where e_{max} is the prescribed precision derived from the corresponding parameters (see later below), and so the next time step is chosen as

$$h = (12e_{max}/x_n''')^{1/2} \tag{11}$$

This expression is used also to check if the step just taken satisfies the precision requirement; if the step size is greater than h, it is rejected and the time point is reworked. The equivalent expression used in SPICE for the Gear method is

$$h = (e_{max}/a_k\, x^{(k+1)})^{1/k} \tag{12}$$

where k is the order of the formula and a_k is a constant depending on k.

The evaluation of e_{max} for a capacitor element in terms of SPICE parameters is as follows. First, an internal current tolerance parameter

$$\text{itol} = \text{ABSTOL} + \text{RELTOL} \times \max(\text{last or new } |i_c|) \tag{13}$$

is defined where i_c is the capacitor current, and last and new refer to the previous and present values, respectively. Similarly, an internal charge tolerance parameter

$$\text{iqtol} = \text{RELTOL} \times \max(\text{qtol or CHGTOL})/h_n \quad \text{with qtol} = \max(\text{last or new } |q_c|) \tag{14}$$

is defined where q_c is the capacitor charge. (Note that CHGTOL is scaled by RELTOL in this expression.) These two parameters are then used to specify

$$e_{max} = \text{TRTOL} \times \max(\text{itol or iqtol})/(12 \times 0.5) \tag{15}$$

in this case for the trapezoidal rule or the second-order Gear formula. SPICE uses the 0.5 heuristically as a multiplier of the third derivative of the capacitor charge.

The default values of ABSTOL, RELTOL, TRTOL and CHGTOL in SPICE are 1 pA, 0.001, 7 and 0.01 pC, respectively. To illustrate the implications of these values, we consider the decay of 1 pC of charge from a 1 pF capacitor in a situation where the effective time constant of discharge is 1 μs. We assume an initial time step of 0.2 μs. The parameters itol and iqtol then evaluate to 1 nA and 5 nA, respectively, and e_{max} to about 5.5 nA which represent an error of about 0.5% in the current discharging from the capacitor. Evaluating analytically q_c''' for the discharging capacitor, and using eqn. 11, the prediction for the next step size h then becomes 0.26 μs which is compatible with our assumed initial step size, bearing in mind that the current is decaying. This example highlights the heuristic nature of the precision and time step control mechanism in SPICE and also shows how difficult it is to relate the parameters ABSTOL, RELTOL, TRTOL, CHGTOL and VNTOL (default value 1 μV) to the actual precision obtained in simulation.

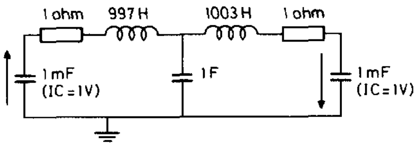


Fig. 8 *Circuit for precision test simulations*

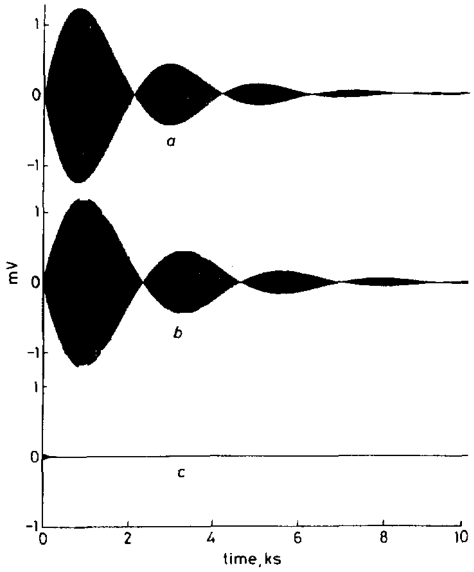


Fig. 9 *Simulation envelopes from precision tests*

The error rate methods of determining step size and estimating local truncation error lead to more realistic, but still heuristic, control of precision for fast transients in simulated waveforms, but are conservative when waveforms are more steady and cause, relatively, more time points in the simulation.

As was mentioned in Section 2, spurious oscillations caused by numerical instability in high Q circuits can be difficult to distinguish from genuine instability of the circuit. Here, we discuss another problem involving ringing effects in high Q circuits not caused by instability but by lack of precision. The energy dissipated per sinusoidal cycle in a high Q circuit is very small and, if less than the allowed local truncation error rate, genuine sinusoidal features in the response can be damped in simulation. A circuit which has been used to examine this [16, 17] is shown in Fig. 8; the 1 F capacitor lightly couples two very high Q loops having marginally distinct normal frequencies. When initially the two 1 mF capacitors are charged to 1 V, in opposite sense, an analytical solution for the time response of the voltage across the 1 F capacitor is as indicated in Fig. 9, trace a. The initial energy in the circuit gradually dissipates by oscillation between the two halves of the circuit at the envelope (beat) frequency. The results of two simulations, with default parameters, using the trapezoidal rule and the second-order Gear method are shown in traces b and c, respectively. The trapezoidal rule result is imprecise (even the beat frequency is wrong!) and the Gear method has damped the oscillation almost to the point of nonexistence. Adjustment of the parameters to tighten up the precision does, however, allow the Gear method, but not the trapezoidal rule, to achieve a result much closer to the analytical solution. In summary, for high Q circuits, the Gear method can numerically damp oscillating circuit behaviour, unless great care is taken with the choice of simulation parameters, while the trapezoidal rule can be imprecise because of a tendency towards instability in the presence of numerical noise. Generally, however, the trapezoidal rule is the preferred method for such circuits, because of the difficulty of defining suitable simulation parameters for the Gear method.

5 The Newton–Raphson method and the convergence domain

The Newton–Raphson process [19] is illustrated in Fig. 10 for a scalar function of a scalar variable. The required solution is the intersection of the function curve with the x-axis. The process starts with an initial estimate $x^{(0)}$ of the solution. The intersection $x^{(1)}$ of the tangent to the curve at $x^{(0)}$ with the x-axis is a refined estimate of the solution. Repeating the procedure starting with $x^{(1)}$ refines the solution further to $x^{(2)}$ which is closer to the desired solution. This iterative procedure is continued until the final x-axis intersection $x^{(k+1)}$, differs by less than some small specified amount from its predecessor $x^{(k)}$. The final iterate is then deemed to be sufficiently close to the solution. Note that the process only converges when the initial estimate of the solution is within a range of convergence; in this case when $x^{(0)}$ is in the interval between the points of zero slope of $F(x)$. The

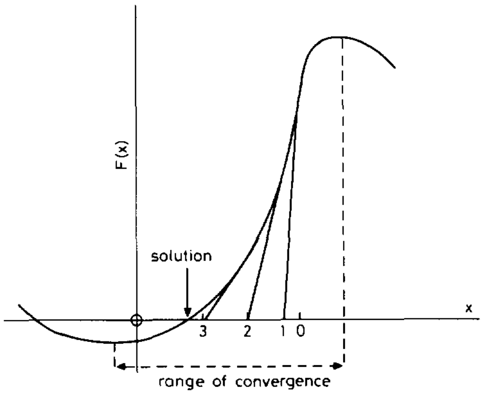


Fig. 10 *Newton–Raphson process in one dimension*

recursive relationship determining successive iterates is

$$x^{(k+1)} = x^{(k)} - [F'(x^{(k)})]^{-1}F(x^{(k)}) \qquad (16)$$

as is evident from simple geometric inspection of Fig. 10 or , alternatively, from the first two terms of the Taylor expansion of $F(x)$, about the value $x^{(k)}$, equated to zero to define $x^{(k+1)}$. Since $[F'(x^{(k)})]^{-1}$ is the reciprocal of a gradient value, no equations have to be solved in this one-dimensional, scalar, case; it is now even more obvious why $F'(x)$ must not be zero.

The vector Newton–Raphson method is illustrated in Fig. 11 for the two-dimensional case. The function F has components F_1 and F_2 which are surfaces intersecting the $F = 0$ plane (defined by the x_1 and x_2 axes) in curved lines as shown in the Figure. The intersection of these lines is the required solution. Given an estimate $x^{(k)}((p_1, p_2)$ in Fig. 12) of the solution, a line through this point in the $F = 0$ plane, is drawn parallel to the F axis to intersect the two surfaces. Tangent planes to the surfaces at these intersection points then intersect the $F = 0$ plane in two straight lines. Finally, the intersection of the two lines yields the revised estimate $x^{(k+1)}$ of the solution. The tangent planes depend on the partial derivatives of the components of F with respect to the components of x and, now using the vector form of Taylor's expansion, the recursive relationship of eqn. 16 becomes

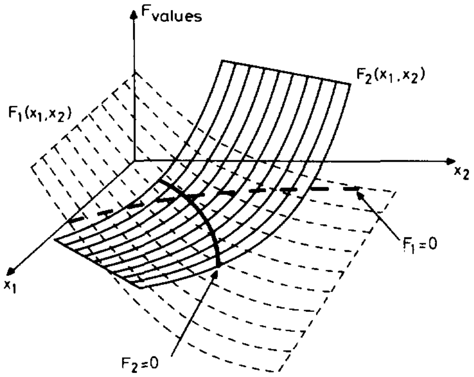


Fig. 11 *Solution of two-dimensional nonlinear equation*

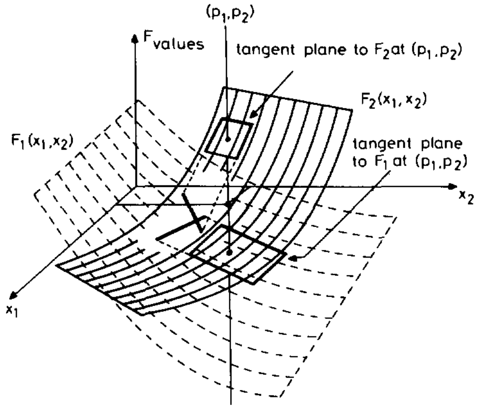

(p1,p2)
F values
tangent plane to F2 at (p1,p2)
F2(x1,x2)
F1(x1,x2)
tangent plane to F1 at (p1,p2)
x2
x1

Fig. 12 *Newton–Raphson process in two dimensions*

$$x^{(p+1)} = x^{(p)} - [J(x^{(p)})]^{-1}F(x^{(p)}) \qquad (17)$$

where $J(x)$ is the Jacobian of $F(x)$ (i.e. the multidimensional equivalent of the gradient $F'(x)$ in eqn. 16). Now $J(x)$ is a square matrix, of size equal to the dimension of x, and so the evaluation of $x^{(p+1)}$ in eqn. 17 requires a numerically costly matrix inversion operation. It is preferable, therefore, to solve the alternative form

$$J(x^{(p+1)})x^{(p+1)} = J(x^{(p)})x^{(p)} - F(x^{(p)}) \qquad (18)$$

which is a linear equation set with coefficient matrix $J(x^{(p)})$ and with the right-hand side of eqn. 18 as the constant vector. As mentioned earlier, the linear equation set is solved repeatedly, updating the coefficient matrix and constant vector each time, until $x^{(p+1)}$ has converged sufficiently.

In some simulators (not SPICE), the linear set

$$J(x^{(p)})\,\Delta x^{(p+1)} = -F(x^{(p)}) \qquad (19)$$

where $\Delta x^{(p+1)} \equiv x^{(p+1)} - x^{(p)}$, is used instead because it is believed that rounding errors, associated with limited precision of floating point number repesentation, arising from cancellation of terms on the right-hand side of the linear equation set, during its solution, are reduced. In this case, the equation set is repeatedly solved until both $\Delta x^{(p+1)}$ and $F(x^{(p)})$ are sufficiently small where the latter plays the role of an error vector since at the true solution $F(x) = 0$.

As briefly explained earlier, the estimate of solution used to initiate the Newton–Raphson iterations must lie within a domain of convergence which includes the real solution [19]. Note that, in multidimensional space, this domain is associated with nonsingularity of $J(x)$, which is equivalent to $F'(x) \neq 0$ in the scalar case. Further, each refinement of the solution produced by the Newton–Raphson process must also lie in this domain. There is no computationally competitive way of determining this domain, and so heuristic methods to ensure convergence are used. Since the solution at the last time point is available, cutting the time step, repeatedly if necessary, is a method of ensuring that the prediction at the new present time-step is within the convergence domain. Given the prediction, it (or one of its successive refinements) is assumed to be outside the domain when convergence is not achieved within a limited number of iterations (ITL4, default value 10). Then, the time step h is cut and the new prediction made. This is a secondary time control mechanism.

6 DC solution

The problem of finding an initial DC solution is more difficult than for the transient simulation case, because no reasonable initial estimate of the DC solution is normally available. A totally quiescent circuit with all sources set to zero values has a known, trivial solution. Performing transient analysis while slowly ramping on the sources to their final values is an almost certain way of obtaining a DC solution, but is not implemented in SPICE explicitly. It can, however, be used by defining the sources appropriately in the circuit description.

The Newton–Raphson process, performed from zero-valued initial estimates, and with inductances shorted, capacitances open circuited, and sources applied, will mostly converge to the correct DC solution. Otherwise, a convergence failure is reported. This method is invoked in SPICE by means of the .OP command.

More reliably [15] in SPICE, a sufficiently large conductance (GMIN, default value 1 pS) connected between each node of a quiescent circuit and ground will ensure convergence of the Newton–Raphson process. The DC solution obtained can then be refined repeatedly with reducing conductance value (until it is finally zero) to yield the correct DC solution. This method is invoked automatically when DC convergence fails, and has proved to be the most effective method of obtaining a solution. Also in SPICE, the initial conductance (GMIN), unreduced, is used to shunt each circuit element diode permanently as an aid to convergence. Further, the limit on the number of Newton–Raphson iterations is higher (ITL1, default value 100) than for transient analysis (ITL4). Convergence failure occurs if this limit is exceeded at any step in the DC solution process.

7 Convergence failure

Apart from the failure modes of the last Section, near singularity of the coefficient matrix of the linear equation set can give rise to numerical instability [11] in the solution. This occurs when the magnitude of the pivot, in the equation reduction by Gaussian elimination, becomes too small (PIVTOL default value 1.0e–12, PIVREL default value 1.0e–3). The elimination pivot is rejected if its magnitude is less than the larger of PIVTOL and of PIVREL times the maximum magnitude element in the rest of the pivot column. This implies that partial (column) pivoting will be used if necessary which compromises a sparse matrix method [19, 20] of solving the linear equation set normally used with consequential significant increase in simulation time.

In a DC solution, failure to find a suitable pivot is akin to a convergence failure and probably implies an unrealistic circuit. In transient simulation, such a failure mostly arises as a time step too small situation following repeated cutting of the time step at a time point in an endeavour to achieve convergence of the Newton–Raphson process or, more probably, in an effort to meet the specified precision (TRTOL). Capacitive and inductive contributions to the coefficient matrix entries, involve terms having a time step *h* divisor which become large as *h* becomes smaller, causing the equation set to become ill conditioned. The convergence domain is thus related to *h* and, in extreme cases, resists attempts to predict into the domain by shrinking *h*. This form of convergence failure is usually indicative of a lack of realism in the circuit or in the modelling of its devices.

8 Floating point precision and the Newton–Raphson process

The precision problem caused by limited floating-point number precision arises when the value of the nonlinear function, in the Newton–Raphson scheme, changes very rapidly near its root (the solution). This can happen in circuits, for example, when very small changes of voltage in a device result in very large changes of current in the device. The effect manifests itself by a negligible (relative to the floating-point precision) change in the estimated solution values, between successive Newton–Raphson iterations, but wild, sign inversion, changes in the function (vector component values). In Fig. 13, the true solution lies in the unresolvable interval marked by the broken lines. The ends of this interval have the function values marked as error. The specified Newton–Raphson convergence precision, less than the magnitude of error, cannot be attained. It is thought (without much confidence) that SPICE incorrectly treats this situation as a convergence failure whereas the negligibly changing estimated solution should be accepted as the solution. In any event, the situation should only arise for somewhat extreme values of circuit components and/or device parameters.

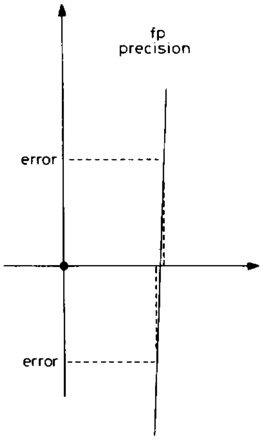


Fig. 13 *Floating point precision limit in Newton–Raphson*

9 Conclusions

We have presented a largely tutorial review of the principles of circuit level simulation, and have done so with the intention of identifying the underlying causes of problems that arise in the practical use of circuit simulation. The authors believe that users of simulation programs need to be aware of how they work, so as to avoid the many difficulties that arise in what often should be straightforward simulations of what appear to be unpretentious circuits. It is the experience of the authors that it is frequently the oversimplified circuits that fail in simulation. It is also their experience that, quite often, apparently successful simulations provide results that are less precise than expected and, in some cases, even patently incorrect. An old adage in the building of simulators business is, 'if it converges and gives them a result, they'll be happy'; much of the aim of this paper is to say to users, 'don't believe it'. The message is that a user should first try to make sure that the results are correct (i.e. they have converged and are not significantly changed by resimulation at higher precision). The next step is to assess them for accuracy in the light of the simulation parameters used, and with knowledge of the limitations of the device models incorporated in the simulator.

Principally, we have addressed and explained the problems of numerical instability, lack of precision in simulation, convergence failure and equation ill conditioning. One important area not covered is simulation problems arising from inappropriate device modelling. We omitted this topic because the generic principles of the topic warrent several papers, and also the many types of device modelled each warrent detailed consideration.

10 References

1 JOHNSON, B., QUARLES, T., NEWTON, A.R., PEDERSON, D.O., and SANGIOVANI-VINCENTELLI, A.: 'SPICE3 Version

3e User's Manual'. Department of EE and CS, Berkeley, University of California, 1991

2 NAGEL, L.W.: 'SPICE2: A computer program to simulate semiconductor circuits'. Electronics research laboratory report UCB/ERL M520, University of California, Berkeley, May 1975

3 McCALLA, W.J.: 'Fundamentals of computer-aided circuit simulation techniques' (Kluwer Academic, Boston, MA, 1975), most chapters

4 HO, C.W., RUEHLI, A.E., and BRENNEN, P.A.: 'The modified nodal approach to network analysis', *Trans. IEEE*, 1975, **CAS-22**, pp. 504–509

5 HACHTEL, G.D., BRAYTON, R.K., and GUSTAVSON, F.G.: 'The sparse tableau approach to network analysis and design', *Trans. IEEE*, 1971, **CT-18**, pp. 101–113

6 GEAR, C.W.: 'The automatic integration of ordinary differential equations', *Commun. ACM*, 1971, **14**, pp. 176–179

7 BRAYTON, R.K., GUSTAVSON, F.G., and HACHTEL, G.D.: 'A new efficient algorithm for solving differential-algebraic systems using implicit backward differential formulas', *Proc. IEEE*, 1972, **60**, pp. 98–108

8 CHUA, L.O., and LIN, P.M.: 'Computer-aided analysis of electronic circuits' (Prentice-Hall, 1975), Chap. 13

9 CHUA, L.O., and LIN, P.M.: 'Computer-aided analysis of electronic circuits' (Prentice-Hall, 1975), Chap. 10

10 SANDBERG, I.W., and SHICHMAN, H.: 'Numerical integration of stiff nonlinear differential equations', *Bell Syst. Tech. J.*, 1968, pp. 511–527

11 WILKINSON, J.H.: 'The algebraic eigenvalue problem' (Clarendon Press, 1965), Chap. 1

12 COSTON, W.T.: 'A transistor companion model including junction capacitances to be used in nonlinear periodic steady-state analysis', *IEEE J.*, 1975, **SC-10**, pp. 40–60

13 KAZMIERSKI, T.J.: 'Remarks on variable-order variable-step backward differentiation formula properties'. Internal Report Department of Electronics, Southampton University, 1983

14 DAHLQUIST, G.C.: 'A special stability problem for linear multistep methods', *BIT*, 1963, **3**, pp. 27–43

15 QUARLES, T.: 'Analysis of performance and convergence issues for circuit simulation'. Memorandum UCB/ERl M89/42, Berkeley, University of California, 1989

16 BOWERS, J.C., and ZOBRIST, G.W.: 'Survey of computer-aided design and analysis programs'. Report of University of South Florida, April, 1976

17 LOURENCO-FERNANDES, J.C.R., and NICHOLS, K.G.: 'Comparitive experience with time-domain nonlinear network integration methods'. Proceedings of the IEEE conference on CAD of electronic and mirowave circuits, Hull, UK, 1977, pp. 17–22

18 ORTEGA, J.M., and RHEINBOLDT, W.C.: 'Iterative solution of nonlinear equations in several variables' (Academic Press, 1970), all chapters, particularly 10

19 BUNCH, J.R., and ROSE, D.J. (Eds.): 'Sparse matrix computations' (Academic Press, 1976), most chapters

20 DUFF, I.S., ERISMAN, A.M., and REID, J.K.: 'Direct methods for sparse matrices' (Clarendon Press, 1986), most chapters