Efficient Steady-State Analysis based on Matrix-Free Krylov-Subspace Methods

Ricardo Telichevesky
Cadence Design Systems
San Jose, California

Kenneth S. Kundert
Cadence Design Systems
San Jose, California

Jacob K. White
Massachusetts Institute of Technology
Cambridge, Massachusetts

Presented at DAC, June 1995

Abstract

Gaussian-elimination based shooting-Newton methods, a commonly used approach for computing steady-state solutions, grow in computational complexity like N^3, where N is the number of circuit equations. Just using iterative methods to solve the shooting-Newton equations results in an algorithm which is still order N^2 because of the cost of calculating the dense sensitivity matrix. Below, a *matrix-free* Krylov-subspace approach is presented, and the method is shown to reduce shooting-Newton computational complexity to that of ordinary transient analysis. Results from several examples are given to demonstrate that the matrix-free approach is more than ten times faster than using iterative methods alone for circuits with as few as 400 equations.

1 Introduction

The growing importance of integrated circuits for communication systems has renewed interest in steady-state methods for distortion analysis of large analog circuits. Finding fast algorithms for accurately computing steady-state solutions is particularly important because many steady-state solutions are needed to determine a single circuit's behavior. For example, to fully characterize a given analog circuit's distortion, it is necessary to sweep both the frequency and amplitude of the applied signal. This implies that an accurate steady-state solution must be computed at each of at least several hundred points in the two sweeps.

Since analog circuits are relatively stable problems, finite-difference methods are not commonly used to compute steady-state solutions. Instead, the two most popular approaches are the harmonic balance algorithm for mildly nonlinear circuits, and the shooting-Newton methods for more drastically nonlinear circuits [Kundert90]. It is also possible to simulate the circuit for a time interval long enough to insure steady-state has been achieved, but for many circuits this time interval may be prohibitively long.

In this paper we focus on shooting-Newton methods. For such methods, if Gaussian-elimination is used to solve the dense shooting-Newton update equations, the method grows in computational complexity like N^3, where N is the number of circuit equations. It is possible to use iterative methods to solve the shooting-Newton update equations, but this results in an algorithm which is still order N^2. In this paper, a *matrix-free* Krylov-subspace approach is presented, and is shown to reduce shooting-Newton computational complexity to that of ordinary transient analysis. In the following background section, we describe the shooting-Newton method, and then describe the matrix-free Krylov-subspace approach in section 3. In section 4, computational results on a several examples are examined, and the results used to show that the matrix-free approach substantially reduces simulation time, particularly for larger circuits. For example, the matrix-free method is more than ten times faster than using iterative methods alone for circuits with as few as 400 equations.

2 Shooting Methods

Finding the periodic steady-state solution of a circuit involves finding the initial condition for the circuit's associated system of differential equations such that the solution at the end of the period matches the initial condition.

More precisely, finding the steady-state solution means finding a particular solution to the circuit equations, as in

$$f(v(t),t) = i(v(t)) + \dot{q}(v(t)) + u(t) = 0, \tag{1}$$

where $u(t) \in \Re^N$ is the vector of input sources, $v(t) \in \Re^N$ is the vector of node voltages, and $i(v(t)), q(v(t)) \in \Re^N$ are the vectors of resistive node currents and node charges or fluxes. The periodic steady-state solution is the solution of (1) that also satisfies the two-point constraint

$$v(T) - v(0) = 0. \tag{2}$$

Generally, shooting methods reformulate (1) and (2) as

$$\phi(v(0),0,T) - v(0) = 0, \tag{3}$$

where ϕ is the state-transition function for (1). As (3) is a nonlinear algebraic problem, standard Newton methods can be used to solve for $v(0)$. We refer to the combination of the Newton and shooting methods as the shooting-Newton algorithm.

When applying Newton's method directly to (3), it is necessary to compute both the response of the circuit over one period and the sensitivity of the final state with respect to changes in the initial state $v(0)$. The sensitivity is used to determine how to correct the initial state to reduce the difference between the initial and final state [Aprille72].

Applying Newton's method to (3) results in the iteration

$$v_0^j = v_0^{j-1} - \left[J_\phi(v_0^{j-1},0,T) - I\right]^{-1}\left[\phi(v_0^{j-1},0,T) - v_0^{j-1}\right] \tag{4}$$

where j is the iteration number, $v_0 = v(0)$, I is the identity matrix, and

$$\begin{aligned} J_\phi(v(0),0,T) &= \frac{d}{dv(0)}\left(\phi(v(0),0,T)\right) \\ &= \frac{dv(T)}{dv(0)}. \end{aligned} \tag{5}$$

There are two important pieces to the computation of the Newton iteration given in (4): factoring the matrix $J_\phi(v(0),0,T) - I$, which is a dense matrix in general, and evaluating the state-transition function $\phi(v(0),0,T)$ and its derivative $J_\phi(v(0),0,T)$.

The state-transition function is computed by integrating (1) numerically over the shooting interval. The derivative of the state-transition function, referred to as the sensitivity matrix, is computed simultaneously because there are several quantities that are common to both computations. To see this, consider solving (1) using the backward-Euler integration method. The resulting discretized equation for v at the m^{th} time-step is then

$$f(v_m) = \frac{1}{h_m}[q(v_m) - q(v_{m-1})] + i(v_m) + u_m = 0 \tag{6}$$

where v_m is an approximation to $v(t_m)$, $u_m = u(t_m)$, $m = 1,2,\ldots,M$, $h_m = t_m - t_{m-1}$ is the time-step, $t_0 = 0$, and $t_M = T$.

Using Newton's method to solve the implicit relation in (6) leads to the iteration

$$\begin{aligned} \left[\frac{1}{h_m}\frac{dq(v_m^{\ell-1})}{dv_m} + \frac{di(v_m^{\ell-1})}{dv_m}\right](v_m^\ell - v_m^{\ell-1}) = \\ -\frac{1}{h_m}\left(q(v_m^{\ell-1}) - q(v_{m-1})\right) - i(v_m^{\ell-1}) - u_m, \end{aligned} \tag{7}$$

where ℓ is the Newton iteration index.

Using the notation $di(v)/dv = G(v)$ and $dq(v)/dv = C(v)$ results in

$$\begin{aligned} \left[\frac{C(v_m^{\ell-1})}{h_m} + G(v_m^{\ell-1})\right](v_m^\ell - v_m^{\ell-1}) = \\ -\frac{1}{h_m}\left(q(v_m^{\ell-1}) - q(v_{m-1})\right) - i(v_m^{\ell-1}) - u_m \end{aligned} \tag{8}$$

The sensitivity matrix, $J_\phi = dv_M/dv_0$, can be computed by differentiating both sides of (6) with respect to v_0,

$$\frac{1}{h_m}\frac{d}{dv_0}\left(q(v_m) - q(v_{m-1})\right) + \frac{d}{dv_0}i(v_m) = 0, \tag{9}$$

which, after applying the chain rule, can be written as

$$\left[\frac{C(v_m)}{h_m} + G(v_m)\right]\frac{dv_m}{dv_0} = \frac{C(v_{m-1})}{h_m}\frac{dv_{m-1}}{dv_0} \tag{10}$$

or

$$J_f(v_m)\frac{dv_m}{dv_0} = \frac{C(v_{m-1})}{h_m}\frac{dv_{m-1}}{dv_0} \tag{11}$$

where $J_f(v_m) = C(v_m)/h_m + G(v_m)$.

The Jacobian $J_\phi(v_0,0,T) = dv_M/dv_0$ is computed by repeated application of (11) starting from the initial condition $dv_0/dv_0 = I$. Note that for each time-step the derivatives $J_f(v_m)$ and $C(v_{m-1})$ in (11) are already available, as they are required in (8), and $J_f(v_m)$ will already be factored. Then to compute the dv_m/dv_0 matrix from the dv_{m-1}/dv_0 matrix requires that each column of the dv_{m-1}/dv_0 matrix be multiplied by $C(v_{m-1})/h_m$ and then solved using the sparse LU factored $J_f(v_m)$. Since dv_{m-1}/dv_0 is dense, the computational work per time-step for computing the sensitivity matrix is at least order N^2; there are at least order N computations for each of the N solves. The size of $J_\phi(v_0,0,T)$ can be reduced somewhat by eliminating columns associated with rapidly decaying states [Kakizaki85].

Algorithm I
(GMRES algorithm for solving $Ax = b$)

Guess at a solution, x^0.
Initialize the search direction $p^0 = b - Ax^0$.
Set $k = 1$.
do {
  Compute the new search direction, $p^k = Ap^{k-1}$.
  Orthogonalize, $p^k = p^k - \sum_{j=0}^{k-1} \beta_{k,j} p^j$.
  Choose α_k in
    $x^k = x^{k-1} + \alpha_k p^k$
  to minimize $\|r^k\| = \|b - Ax^k\|$.

  If $\|r^k\| < tolerance_{\text{gmres}}$, return v^k as the solution.
  else Set $k = k + 1$.
}

3 The Matrix-Free Krylov-Subspace Approach

Each iteration of the shooting-Newton method requires solving the dense linear system given in (4). If Gaussian elimination is used to solve (4), the number of floating point operations required will grow proportionly with the *cube* of the number of unknowns. Clearly, the Gaussian elimination approach will become computationally intractable if the number of circuit equations exceeds several hundred. Instead, consider solving the linear system (4) using an iterative method like the Krylov-subspace based GMRES algorithm [Saad86]. A simplified version of GMRES is given in Algorithm I.

The dominant costs of Algorithm I are in calculating the N^2 entries of $A = J_\phi - I$ using (11) before the iterations begin, and performing N^2 operations to compute Ap^{k-1} on each GMRES iteration. Below, we describe a matrix-free approach which represents $J_\phi - I$ in a sparse form. The approach avoids forming most of A and, for typical circuit problems, reduces the cost of computing $J_\phi p^{k-1}$ to nearly order N operations.

3.1 The Matrix Free Approach

To derive an approach which avoids forming $A = J_\phi - I$, recall that GMRES does not require an explicit representation of A, only the ability to compute Ap^{k-1} is necessary. Note that

$$Ap^{k-1} = (J_\phi - I)p^{k-1}$$

Algorithm II
(Backward-Euler Matrix-Free Shooting-Newton)

Guess a solution, v_0^0.
For $j = 1$ to Newton Limit {

  Integrate (1) from 0 to T with $v_0 = v_0^{j-1}$:
  For $m = 1$ to M {
    Solve (6) for v_m.
    Store the factored $J_f(v_m)$ and $C(v_m)$).
  }

  Solve $(J_\phi(v^{j-1}) - I)\delta v^j = v^{j-1} - \phi(v^{j-1})$:
    (with GMRES)
    In GMRES Compute $p^{k+1} = J_\phi(v^{j-1})p^k$ using:
    $p^{k+1} = p^k$.
    For $m = 1$ to M solve $J_f(v_m)p^{k+1} = C(v_{m-1})p^{k+1}$.

  Upate $v^j = v^{j-1} + \delta v^j$.
  If $\|\delta v^j\| < tolerance_{\text{Newton}}$, return.
}

$$\approx \frac{\phi(v_0 + \epsilon p^{k-1}, 0, T) - \phi(v(0))}{\epsilon} - p^{k-1}. \quad (12)$$

Therefore, Ap^{k-1} can be computed just by perturbing $v(0)$ in the direction of p^k, then integrating for one period to evaluate $\phi(v_0 + \epsilon p^{k-1}, 0, T)$, and finally applying the relation in (12) [Skelboe80].

Even though (12) can be used directly as a matrix-free approach to forming the matrix-vector products required when GMRES is applied to solving (4), a computationally more efficient and numerically more robust approach is to save $C(v_m)$ and the LU factorization of $J_f(v_m)$ at each time-step, and use (11) to compute $J_\phi p^{k-1}$. This leads to the Algorithm II for computing the steady-state.

Note that computing the GMRES matrix-vector product in Algorithm II requires nearly the same work as computing one column of J_ϕ using (11). Therefore, if the GMRES algorithm converges in many fewer than N iterations, the net computational saving over computing all N columns of J_ϕ needed in a standard shooting-Newton method will be considerable. In the results section below, we will demonstrate by example that on practical circuits the GMRES algorithm converges very rapidly.

It may seem to be a misnomer to refer to Algorithm II as a matrix-free shooting-Newton method, given that so many $J_f(v_m)$ and $C(v_m)$ matrices are being computed and stored. However, $A = J_\phi - I$ is *not* being computed

explicitly, and therefore the method is formally referred to as being matrix-free.

3.2 Saving $J_f(v_m)$ and $C(v_m)$ at Each Time-Step

As mentioned above, (12) can be used directly to approximately compute matrix-vector products, but such an approach would be somewhat inefficient. Each matrix-vector product would require an entire single-period time integration. Instead, substantial additional storage can be allocated so that $C(v_m)$ and the factored version of $J_f(v_m)$ can be stored at each time-step. Although this matrix storage can be significant, for very large circuits the required storage is less than the N^2 storage which would be needed for the dense representation of the sensitivity matrix. In addition, since the nonzero pattern of $C(v_m)$ and the factored version of $J_f(v_m)$ does not change with m, matrix structure information can be stored just once.

3.3 Higher Order Integration Methods

Certainly, to achieve efficiently the high accuracy needed in distortion computation, it is necessary to use a higher order integration method in the transient simulation than backward-Euler. If any of the backward-difference formulas are used, which have the general form

$$i(v_m) + u_m + \sum_{p=0}^{p=P} \alpha_p q(v_{m-p}) = 0, \tag{13}$$

where P is the integration method order, they can be adapted for use in matrix-free steady-state method without further increasing the storage over what is required for backward-Euler. As is clear from the derivation in Section 2, the equation for dv_m/dv_0 is

$$G(v_m)\frac{dv_m}{dv_0} + \sum_{p=0}^{p=P} \alpha_p C(v_{m-p})\frac{dv_{m-p}}{dv_0} = 0, \tag{14}$$

and therefore no new capacitance matrices must be stored when changing from backward-Euler to higher order methods, it is only necessary to access the already stored C matrices from previous time-steps.

4 Results

In this section we experimentally examine the performance of three shooting-Newton schemes: Gaussian elimination, explicit GMRES, and matrix-free GMRES.

circuit	eqns	it	GE	GMRES	MF	GE/MF
xtal	29	3	0.50	0.50	0.39	1.28
mixer	24	4	1.85	1.74	1.20	1.54
dbmixer	100	4	4.15	4.07	1.34	3.09
lmixer	126	3	3.72	3.63	1.03	3.61
cheby	237	4	23.39	21.97	3.01	7.96
scf	377	6	2962	2954	281.4	10.52

Table 1: Comparison of different shooting method schemes

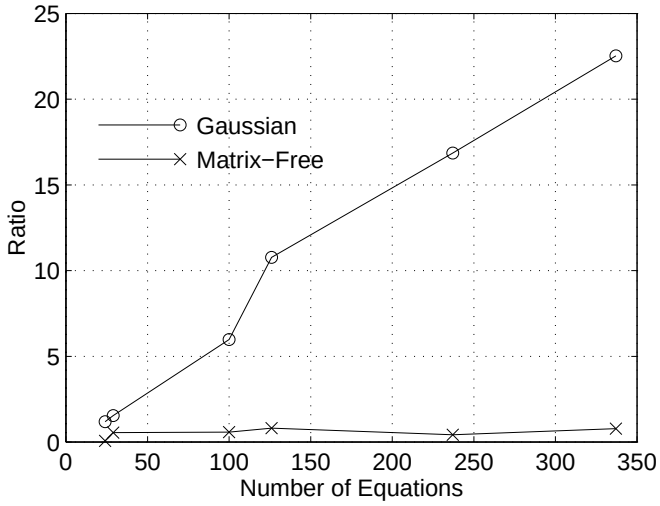

Figure 1: Cost ratio of shooting method overhead to transient analysis

Table 1 compares the performance of the various shooting-Newton methods as implemented in the Spectre™ circuit simulator available from Cadence Design Systems. The test suite includes *xtal*, a crystal filter; *mixer* is a small GaAs mixer; *dbmixer* is a double balanced mixer; *lmixer* is a large bipolar mixer; *cheby* is an active filter; and *scf* is a relatively large switched capacitor filter. The second column in Table 1 lists the number of equations in each circuit. The third column represents the number of one-period transient analyses that were necessary to achieve steady-state using the shooting-Newton method. The fourth, fifth, and sixth columns represent, respectively, the time in seconds to achieve steady-state using Gaussian elimination (GE), explicit GMRES, and the matrix-free (MF) algorithm. All the results were obtained on a HP712/80 workstation. The seventh column demonstrates the effectiveness of the matrix-free approach, listing the speedup obtained with respect to the Gaussian-elimination method. Note that the speed-up

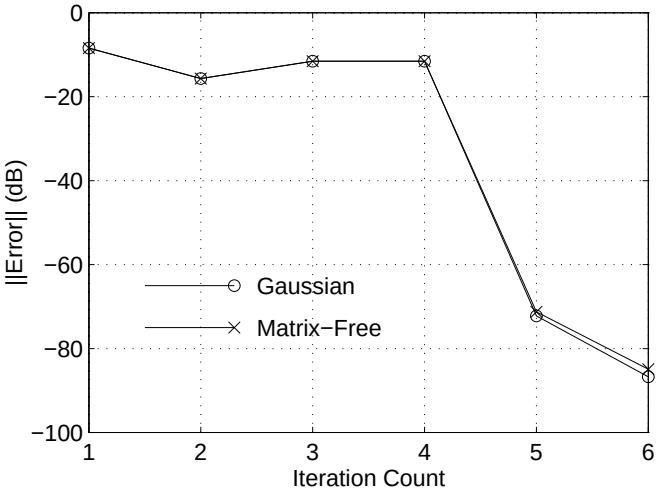

Figure 2: Convergence rate behavior for *scf*.

over the explicit GMRES algorithm would be similar for the examples examined.

Figure 1 compares the shooting update time relative to the costs of performing a one-period transient analysis for the different methods. The matrix-free method exhibits a relatively constant ratio, i.e. the time for computing the update remains comparable to the transient analysis time regardless of the number of equations, while the other methods exhibit nearly a linear growth in this ratio. In circuits with as few as 100 nodes, the costs of computing the shooting update are an order of magnitude larger than the transient simulation costs.

Figure 2 shows the convergence rate of the shooting-Newton method for the relatively large switched-capacitor filter example. The figure clearly indicates that using matrix-free GMRES to compute the shooting-Newton update instead of using conventional Gaussian elimination has little effect on the Newton method's convergence.

5 Conclusions

In this paper we described how to use a matrix-free Krylov-subspace based matrix solution method to reduce the shooting method computational complexity to nearly order N in practice, and gave computational results on a variety of examples to demonstrate the effectiveness of this approach. In particular, we demonstrated the method reduces simulation time by more than a factor of ten for circuits with as few as four hundred equations.

Algorithms based on matrix-free Krylov-subspace methods are likely to have impact on other matrix solution problems in circuit simulation, such as harmonic balance and mixed frequency-time methods [Kundert90, Heikkila92].

References

[Aprille72] Thomas J. Aprille and Timothy N. Trick. "Steady-state analysis of nonlinear circuits with periodic inputs." *Proceedings of the IEEE*, vol. 60, no. 1, pp. 108–114, January 1972.

[Heikkila92] Pauli Heikkilä. *Object-Oriented Approach to Numerical Circuit Analysis.* Ph. D. dissertation, Helsinki University of Technology, January 1992.

[Kakizaki85] Makiko Kakizaki and Tsutomu Sugawara. "A modified Newton method for the steady-state analysis." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-4, no. 4, pp. 662–667, October 1985.

[Kundert90] Kenneth S. Kundert, Jacob K. White, and Alberto Sangiovanni-Vincentelli. *Steady-State Methods for Simulating Analog And Microwave Circuits.* Kluwer Academic Publishers, Boston 1990.

[Nagel75] L. W. Nagel. *SPICE2: A Computer Program to Simulate Semiconductor Circuits.* Electronics Research Lab Report, ERL M520, University of California, Berkeley, May 1975.

[Saad86] Y. Saad and M. H. Schultz. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems." *SIAM Journal on Scientific and Statistical Computing*, vol. 7, pp. 856–869, July 1986.

[Skelboe80] Stig Skelboe. "Computation of the periodic steady-state response of nonlinear networks by extrapolation methods." *IEEE Transactions on Circuits and Systems*, vol. CAS-27, no. 3, pp. 161-175, March 1980.