INVITED PAPER

Fundamentals of Fast Simulation Algorithms for RF Circuits

The newest generation of circuit simulators perform periodic steady-state analysis of RF circuits containing thousands of devices using a variety of matrix-implicit techniques which share a common analytical framework.

By Ognen Nastov, Ricardo Telichevesky, *Member IEEE*, Ken Kundert, and Jacob White, *Member IEEE*

ABSTRACT | Designers of RF circuits such as power amplifiers, mixers, and filters make extensive use of simulation tools which perform periodic steady-state analysis and its extensions, but until the mid 1990s, the computational costs of these simulation tools restricted designers from simulating the behavior of complete RF subsystems. The introduction of fast matrix-implicit iterative algorithms completely changed this situation, and extensions of these fast methods are providing tools which can perform periodic, quasi-periodic, and periodic noise analysis of circuits with thousands of devices. Even though there are a number of research groups continuing to develop extensions of matrix-implicit methods, there is still no compact characterization which introduces the novice researcher to the fundamental issues. In this paper, we examine the basic periodic steady-state problem and provide both examples and linear algebra abstractions to demonstrate connections between seemingly dissimilar methods and to try to provide a more general framework for fast methods than the standard time-versus-frequency domain characterization of finite-difference, basis-collocation, and shooting methods.

KEYWORDS | Circuit simulation; computer-aided analysis; design automation; frequency-domain analysis; numerical analysis

I. INTRODUCTION

The intensifying demand for very high performance portable communication systems has greatly expanded the need for simulation algorithms that can be used to efficiently and accurately analyze frequency response, distortion, and noise of RF communication circuits such as mixers, switched-capacitor filters, and amplifiers. Although methods like multitone harmonic balance, linear time-varying, and mixed frequency-time techniques [4], [6]–[8], [26], [37] can perform these analyses, the computation cost of the earliest implementations of these techniques grew so rapidly with increasing circuit size that they were too computationally expensive to use for more complicated circuits. Over the past decade, algorithmic developments based on preconditioned matrix-implicit Krylov-subspace iterative methods have dramatically changed the situation, and there are now tools which can easily analyze circuits with thousands of devices. Preconditioned iterative techniques have been used to accelerate periodic steady-state analysis based on harmonic balance methods [5], [11], [30], time-domain shooting methods [13], and basis-collocation schemes [41]. Additional results for more general analyses appear constantly.

Though there are numerous excellent surveys on analysis technques for RF circuits [23], [35], [36], [42], the literature analyzing the fundmentals of fast methods is limited [40], making it difficult for novice researchers to contribute to the field. In this paper, we try to provide a comprehensive yet approachable presentation of fast

Manuscript received May 23, 2006; revised August 27, 2006. This work was originally supported by the DARPA MAFET program, and subsequently supported by grants from the National Science Foundation, in part by the MARCO Interconnect Focus Center, and in part by the Semiconductor Research Center.
O. Nastov is with Agilent Technologies, Inc., Westlake Village, CA 91362 USA (e-mail: ognen_nastov@agilent.com).
R. Telichevesky is with Kineret Design Automation, Inc., Santa Clara, CA 95054 USA (e-mail: ricardo@teli.org).
K. Kundert is with Designer's Guide Consulting, Inc., Los Altos, CA 94022 USA (e-mail: ken@designers-guide.com).
J. White is with the Research Laboratory of Electronics and Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: white@mit.edu).
Digital Object Identifier: 10.1109/JPROC.2006.889366

methods for periodic steady-state analysis by combining specific examples with clarifying linear algebra abstractions. Using these abstractions, we can demonstrate the clear connections between finite-difference, shooting, harmonic balance, and basis-collocation methods for solving steady-state problems. For example, among users of circuit simulation programs, it is common to categorize numerical techniques for computing periodic steady-state as either time-domain (finite-difference) or frequency-domain (harmonic balance), but the use of Kronecker product representations in this paper will make clear that none of the fast methods for periodic steady-state fundamentally rely on properties of a Fourier series.

We start, in the next section, by describing the different approaches for fomulating steady-state problems and then present the standard numerical techniques in a more general framework that relies heavily on the Kronecker product representation. Fast methods are described in Section IV, along with some analysis and computational results. And as is common practice, we end with conclusions and acknowledgements.

II. PERIODIC STEADY STATE

As an example of periodic steady state analysis, consider the *RLC* circuit shown in Fig. 1. If the current source in Fig. 1 is a sinusoid, and the initial capacitor voltage and inductor current are both zero, then the capacitor voltage will behave as shown in Fig. 2. As the figure shows, the response of the circuit is a sinusoid whose amplitude grows until achieving a periodically repeating steady state. The solution plotted in Fig. 2 is the result of a numerical simulation, and each of the many small circles in the plot corresponds to a simulation timestep. Notice that a very large number of timesteps are needed to compute this solution because of the many oscillation cycles before the solution builds up to a steady state.

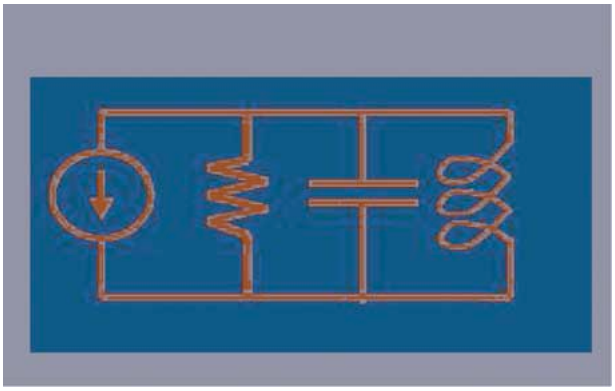

Fig. 1. *Parallel resistor, capacitor, and inductor (RLC) circuit with current source input.*

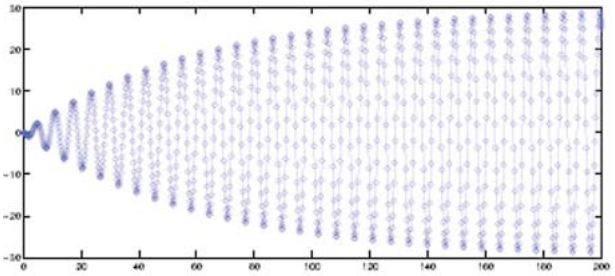

Fig. 2. *Transient behavior of RLC circuit with $R = 30$, $C = 1$, $L = 1$, and $i_s(t) = \cos t$.*

For this simple *RLC* circuit, it is possible to avoid the many timestep simulation and directly compute the sinusoidal steady state using what is sometimes referred to as phasor analysis [2]. To use phasor analysis, first recall that the time behavior of the capacitor voltage for such an *RLC* circuit satisfies the differential equation

$$\frac{d^2 v_c(t)}{dt^2} + \frac{1}{rc}\frac{dv(t)}{dt} + \frac{1}{lc}v(t) + \frac{di(t)}{dt} = 0. \tag{1}$$

Then, since a cosinusoidal input current is the real part of a complex exponential, $I_o e^{j\omega t}$ where $j = \sqrt{-1}$, in sinusoidal steady state the voltage must also be the real part of a complex exponential given by

$$v(t) = \text{Real}\left(\frac{j\omega}{-\omega^2 + \frac{1}{rc}\omega + \frac{1}{lc}} I_o e^{j\omega t}\right) \tag{2}$$

as can be seen by substituting $v(t) = V_o e^{j\omega t}$ in (1) and solving for V_o.

The simple phasor analysis used above for computing sinusoidal steady is not easily generalized to nonlinear circuits, such as those with diodes and transistors. The behavior of such nonlinear circuits may repeat periodically in time given a periodic input, but that periodic response will almost certainly not be a sinusoid. However, there are approaches for formulating systems of equations that can be used *to compute directly the periodic steady state* of a given nonlinear circuit, avoiding the many cycle time integration shown in Fig. 2. In the next subsection, we briefly describe one standard form for generating systems of differential equations from circuits, and the subsections that follow describe two periodic steady-state equation formulations, one based on replacing the differential equation initial condition with a boundary condition and the second based on using the abstract idea of a state transition function. In later sections, we will describe

several numerical techniques for solving these two formulations of the periodic steady-state problem.

A. Circuit Differential Equations

In order to describe a circuit to a simulation program, one must specify both the topology, how the circuit elements are interconnected, and the element constitutive equations, how the element terminal currents and terminal voltages are related. The interconnection can be specified by labeling $n+1$ connection points of element terminals, referred to as the nodes, and then listing which of b element terminals are connected to which node. A system of n equations in b unknowns is then generated by insisting that the terminal currents incident at each node, except for a reference or "ground" node, sum to zero. This conservation law equation is usually referred to as the Kirchhoff current law (KCL). In order to generate a complete system, the element constitutive equations are used to relate the n node voltages with respect to ground to b element terminal currents. The result is a system of $n+b$ equations in $n+b$ variables, the variables being ground-referenced node voltages and terminal currents. For most circuit elements, the constitutive equations are written in a voltage-controlled form, meaning that the terminal currents are explicit functions of terminal voltages. The voltage-controlled constitutive equations can be used to eliminate most of the unknown branch currents in the KCL equation, generating a system of equations with mostly ground-referenced node voltages as unknowns. This approach to generating a system of equations is referred to as modified nodal analysis and is the equation formulation most commonly used in circuit simulation programs [1], [3].

For circuits with energy storage elements, such as inductors and capacitors, the constitutive equations include time derivatives, so modified nodal analysis generates a system of N differential equations in N variables of the form

$$\frac{d}{dt}q(v(t)) + i(v(t)) + u(t) = 0 \tag{3}$$

where t denotes time, $u(t)$ is a N-length vector of given inputs, v is an N-length vector of ground-referenced node voltages and possibly several terminal currents, $i(\cdot)$ is a function that maps the vector of N mostly node voltages to a vector of N entries most of which are sums of resistive currents at a node, and $q(\cdot)$ is a function which maps the vector of N mostly node voltages to a vector of N entries that are mostly sums of capacitive charges or inductive fluxes at a node.

If the element constitutive equations are linear, or are linearized, then (3) can be rewritten in matrix form

$$C\frac{d}{dt}v(t) + Gv(t) + u(t) = 0 \tag{4}$$

where C and G are each $N \times N$ matrices whose elements are given by

$$C_{j,k} = \frac{\partial q_j}{\partial v_k} \quad G_{j,k} = \frac{\partial i_j}{\partial v_k}. \tag{5}$$

The basic forms given in (3) and (4) will be used extensively throughout the rest of this paper, so to make the ideas clearer, consider the example of the current-source driven *RLC* circuit given in Fig. 1. The differential equation system generated by modified nodal analysis is given by

$$\begin{bmatrix} c & 0 \\ 0 & l \end{bmatrix}\frac{d}{dt}\begin{bmatrix} v_c(t) \\ i_l(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{r} & 1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} v_c(t) \\ i_l(t) \end{bmatrix} + \begin{bmatrix} i_s(t) \\ 0 \end{bmatrix} = 0 \tag{6}$$

where $v_c(t)$ is the voltage across the capacitor, $i_l(t)$ is the current through the inductor, and $i_s(t)$ is the source current.

When the circuit of interest contains only capacitive and resistive elements, and all the sources are current sources, v is precisely a set of ground-referenced node voltages, q is a vector of sums of charges at a node, i is a vector of sums of currents at a node, C is a capacitance matrix, and G is a conductance matrix. As an example of this common special case, consider the N node *RC* line example in Fig. 3. The differential equation system is given by

$$\begin{bmatrix} c & 0 & 0 & \dots & 0 \\ 0 & c & 0 & \dots & 0 \\ & & \ddots & & \\ & & & \ddots & \\ 0 & 0 & \dots & 0 & c \end{bmatrix}\frac{d}{dt}\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} + \begin{bmatrix} 2g & -g & 0 & \dots & 0 \\ -g & 2g & -g & \dots & 0 \\ & & \ddots & & \\ & & & \ddots & \\ 0 & 0 & \dots & -g & g \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} + \begin{bmatrix} i_s(t) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{7}$$

where $1/g = r$ from (6).

B. Boundary Condition Formulation

A given function of time, $x(t)$, is said to be periodic with period T if

$$x(t) = x(t+T) \tag{8}$$

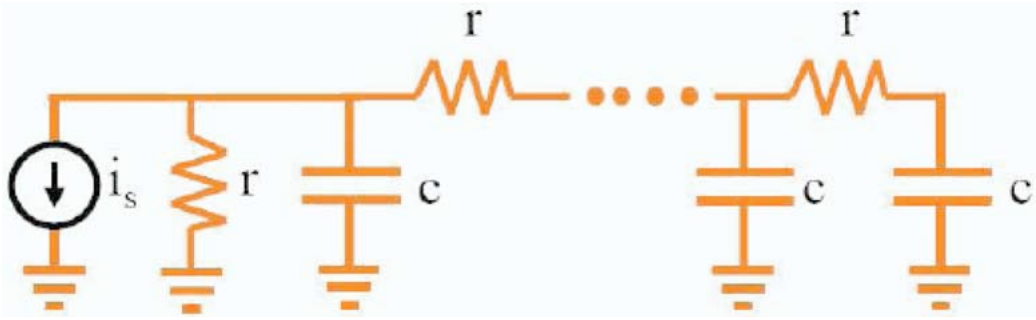

Fig. 3. *Resistor-capacitor (RC) line circuit with current source input.*

for all t. The circuit differential equation system has a periodic solution if the input $u(t)$ is periodic and there exists a periodic $v(t)$ that satisfies (3).

The above condition for a periodic solution suggests that it is necessary to verify periodicity at every time instance t, but under certain very mild conditions this is not the case. If the $q(\cdot)$ and $i(\cdot)$ satisfy certain smoothness conditions, then given a particular intial condition and input, the solution to (3) will exist and be unique. This uniqueness implies that if $v(0) = v(T)$, and $u(t) = u(t+T)$ for all t, then $v(t) = v(t+T)$ for all t. To see this, consider that at time T, the input and state are such that it is identical to restarting the differential equation at $t = 0$. Therefore, uniqueness requires that the solution on $t \in [T, 2T]$ replicates the solution on $t \in [0, T]$.

The above analysis suggests that a system of equations whose solution is periodic can be generated by appending the differential equation system (3) with what is often referred as a two-point boundary constraint, as in

$$\frac{d}{dt}q(v(t)) + i(v(t)) + u(t) = 0 \quad v(T) - v(0) = 0. \tag{9}$$

The situation is shown diagrammatically in Fig. 4.

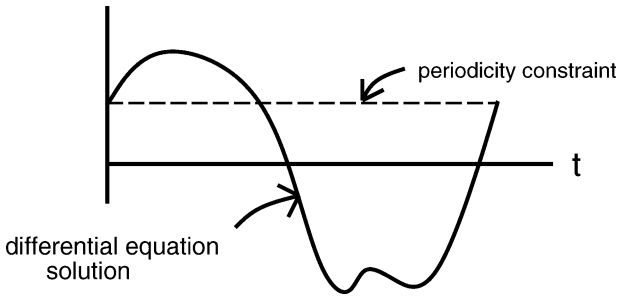

Fig. 4. *Pictorial representation of periodic steady-state condition.*

As an example, consider the *RLC* example in Fig. 1, whose associated differential equation system is given in (6) and whose response to a sinusoidal current source from zero initial conditions is plotted in Fig. 2. As is easily verified, if the initial condition on the inductor current is zero, and the initial voltage on the capacitor $v(0) = 30.0$, then a single period simulation will produce one of the last cycles in Fig. 2.

C. State Transistion Function Formulation

An alternative point of view of the differential equation system in (3) is to treat the system as implicitly defining an algebraic function which maps an initial condition, an N-length vector v_o, and a time, τ, to the solution of the system at time τ, an N-length vector v_τ. The result of applying this implicitly defined state transition function to a given initial condition and time is,

$$v_\tau = \Phi(v_o, \tau) \tag{10}$$

and Φ can be evaluated by solving (3) for $v(t)$ with the initial condition $v(0) = v_o$, and then setting $v_\tau = v(\tau)$.

Rephrasing the result from above, given a differential equation system whose nonlinearities satisfy smoothness conditions and whose input is periodic with period T, if the solution to that system satisfies $v(0) = v(T)$, then the $v(t)$ computed from the initial condition $v_T = v(T) = v(0)$ will be the periodic steady state. The state transition function, though implicitly defined, yields an elegant way of expressing a nonlinear algebraic equation for such a v_T, as in

$$v_T - \Phi(v_T, T) = 0. \tag{11}$$

1) State Transition Function Examples: The state transition function is a straightforward but abstract construction best made clear by examples. As a very simple example, consider the *RLC* circuit in Fig. 1 with no inductor. The

example is then an *RC* circuit described by the scalar differential equation

$$c\frac{d}{dt}v(t)+\frac{1}{r}v(t)+u(t)=0. \tag{12}$$

The analytic solution of the scalar differential equation given an initial condition $v(0)=v_o$ and a nonzero c is

$$v(t)=e^{-\frac{t}{rc}}v_o-\int_0^t e^{-\frac{t-\tau}{rc}}\frac{u(\tau)}{c}d\tau=\Phi(v_o,t). \tag{13}$$

If $u(t)$ is periodic with period T, then (11) can be combined with (13) resulting in a formula for v_T

$$v_T=-\frac{1}{1-e^{-\frac{T}{rc}}}\int_0^T e^{-\frac{T-\tau}{rc}}\frac{u(\tau)}{c}d\tau. \tag{14}$$

As a second more general example, consider the linear differential equation system given in (4). If the C matrix is invertible, then the system can be recast as

$$\frac{d}{dt}v(t)=-Av(t)-C^{-1}u(t) \tag{15}$$

where A is an $N\times N$ matrix with $A=C^{-1}G$. The solution to (15) can be written explicitly using the matrix exponential [2]

$$v(t)=e^{-At}v_o-\int_0^t e^{-A(t-\tau)}C^{-1}u(\tau)d\tau \tag{16}$$

where e^{-At} is the $N\times N$ matrix exponential. Combining (11) with (16) results in a linear system of equations for the vector v_T

$$(I_N-e^{-AT})v_T=-\int_0^t e^{-A(t-\tau)}C^{-1}u(\tau)d\tau \tag{17}$$

where I_N is the $N\times N$ identity matrix.

For nonlinear systems, there is generally no explicit form for the state transition function $\Phi(\cdot)$; instead, $\Phi(\cdot)$ is usually evaluated numerically. This issue will reappear frequently in the material that follows.

III. STANDARD NUMERICAL METHODS

In this section we describe the finite-difference and basis collocation techniques used to compute periodic steady states from the differential equation plus boundary condition formulation, and then we describe the shooting methods used to compute periodic steady-state solutions from the state transition function-based formulation. The main goal will be to establish connections between methods that will make the application of fast methods in the next section more transparent. To that end, we will introduce two general techniques. First, we review the multidimensional Newton's method which will be used to solve the nonlinear algebraic system of equations generated by each of the approaches to computing steady states. Second, we will introduce the Kronecker product. The Kronecker product abstraction is used in this section to demonstrate the close connection between finite-difference and basis collocation techniques, and is used in the next section to describe several of the fast algorithms.

A. Newton's Method

The steady-state methods described as follows all generate systems of Q nonlinear algebraic equations in Q unknowns in the form

$$F(x)\equiv\begin{bmatrix}f_1(x_1,\ldots,x_Q)\\ f_2(x_1,\ldots,x_Q)\\ \vdots\\ f_Q(x_1,\ldots,x_Q)\end{bmatrix}=0 \tag{18}$$

where each $f_i(\cdot)$ is a scalar nonlinear function of a q-length vector variable.

The most commonly used class of methods for numerically solving (18) are variants of the iterative multidimensional Newton's method [18]. The basic Newton method can be derived by examining the first terms in a Taylor series expansion about a guess at the solution to (18)

$$0=F(x^*)\approx F(x)+J(x)(x^*-x) \tag{19}$$

where x and x^* are the guess and the exact solution to (18), respectively, and $J(x)$ is the $Q\times Q$ Jacobian matrix whose elements are given by

$$J_{i,j}(x)=\frac{\partial f_i(x)}{\partial x_j}. \tag{20}$$

The expansion in (19) suggests that given x^k, the estimate generated by the kth step of an iterative algorithm, it is possible to improve this estimate by solving the linear system

$$J(x^k)(x^{k+1} - x^k) = -F(x^k) \tag{21}$$

where x^{k+1} is the improved estimate of x^*.

The errors generated by the multidimensional Newton method satisfy

$$\|x^* - x^{k+1}\| \leq \gamma \|x^* - x^k\|^2 \tag{22}$$

where κ is proportional to bounds on $\|J(x)^{-1}\|$ and the ratio $\|F(x) - F(y)\|/\|x - y\|$. Roughly, (22) implies that if $F(x)$ and $J(x)$ are well behaved, Newton's method will converge very rapidly given a good initial guess. Variants of Newton's method are often used to improve the convergence properties when the initial guess is far from the solution [18], but the basic Newton method is sufficent for the purposes of this paper.

B. Finite-Difference Methods

Perhaps the most straightforward approach to numerically solving (9) is to introduce a time discretization, meaning that $v(t)$ is represented over the period T by a sequence of M numerically computed discrete points

$$\begin{bmatrix} v(t_1) \\ v(t_2) \\ \vdots \\ v(t_M) \end{bmatrix} \approx \begin{bmatrix} \hat{v}(t_1) \\ \hat{v}(t_2) \\ \vdots \\ \hat{v}(t_M) \end{bmatrix} \equiv \hat{\boldsymbol{v}} \tag{23}$$

where $t_M = T$, the hat is used to denote numerical approximation, and $\hat{\boldsymbol{v}} \in \Re^{MN}$ is introduced for notational convenience. Note that $\hat{\boldsymbol{v}}$ does not include $\hat{v}(t_0)$, as the boundary condition in (9) implies $\hat{v}(t_0) = \hat{v}(t_M)$.

1) Backward-Euler Example: A variety of methods can be used to derive a system of nonlinear equations from which to compute $\hat{\boldsymbol{v}}$. For example, if the backward-Euler method is used to approximate the derivative in (3), then $\hat{\boldsymbol{v}}$ must satisfy a sequence of M systems of nonlinear equations

$$F_m(\hat{\boldsymbol{v}}) \equiv \frac{q(\hat{v}(t_m)) - q(\hat{v}(t_{m-1}))}{h_m} + i(v(t_m)) + u(t_m) = 0 \tag{24}$$

for $m \in \{1, \ldots, M\}$. Here, $h_m \equiv t_m - t_{m-1}$, $F_m(\cdot)$ is a nonlinear function which maps an MN-length vector to an N-length vector and represents the jth backward-Euler timestep equation, and periodicity is invoked to replace $\hat{v}(t_0)$ with $\hat{v}(t_M)$ in the $j = 1$ equation.

The system of equations is diagrammed in Fig. 5.

It is perhaps informative to rewrite (24) in matrix form, as in

$$\begin{bmatrix} \frac{1}{h_1} I_N & & & -\frac{1}{h_1} I_N \\ -\frac{1}{h_2} I_N & \frac{1}{h_2} I_N & & \\ & \ddots & \ddots & \\ & & -\frac{1}{h_M} I_N & \frac{1}{h_M} I_N \end{bmatrix} \begin{bmatrix} q(v(t_1)) \\ q(v(t_2)) \\ \cdots \\ q(v(t_M)) \end{bmatrix} + \begin{bmatrix} i(v(t_1)) \\ i(v(t_2)) \\ \cdots \\ i(v(t_M)) \end{bmatrix} + \begin{bmatrix} u(t_1) \\ u(t_2) \\ \cdots \\ u(t_M) \end{bmatrix} = 0 \tag{25}$$

where I_N is used to denote the $N \times N$ identity matrix.

2) Matrix Representation Using Kronecker Products: The backward-Euler algorithm applied to an M-point discretization of a periodic problem can be more elegantly summarized using the Kronecker product. The Kronecker product of two matrices, an $n \times p$ matrix A and $m \times l$ matrix B, is achieved by replacing each of the np elements of matrix A with a scaled copy of matrix B. The result is the $(n + m) \times (p + l)$ matrix

$$A \otimes B \equiv \begin{bmatrix} A_{1,1}B & A_{1,2}B & \ldots & A_{1,p}B \\ A_{2,1}B & A_{2,2}B & \ldots & A_{2,p}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1}B & A_{n,2}B & \ldots & A_{n,p}B \end{bmatrix}. \tag{26}$$

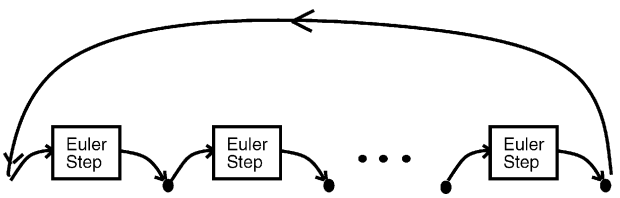

Fig. 5. ***Graphical representation of (24). Note that there is no large dot before the first Euler-step block, indicating that*** $\hat{v}(t_0) = \hat{v}(t_M)$.

The Kronecker product notation makes it possible to summarize the backward-Euler M-step periodic discretization with an $M \times M$ differentiation matrix

$$D_{\text{be}} = \begin{bmatrix} \frac{1}{h_1} & & & -\frac{1}{h_1} \\ -\frac{1}{h_2} & \frac{1}{h_2} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{h_M} & \frac{1}{h_M}, \end{bmatrix} \tag{27}$$

and then apply the backward-Euler periodic discretization to an N-dimensional differential equation system using the Kronecker product. For example, (25) becomes

$$F(\hat{v}) = D_{\text{be}} \otimes I_N \boldsymbol{q}(\hat{\boldsymbol{v}}) + \boldsymbol{i}(\hat{\boldsymbol{v}}) + \boldsymbol{u} = 0 \tag{28}$$

where I_N is the $N \times N$ identity matrix and

$$\boldsymbol{q}(\hat{\boldsymbol{v}}) \equiv \begin{bmatrix} q(\hat{v}(t_1)) \\ q(\hat{v}(t_2)) \\ \vdots \\ q(\hat{v}(t_m)) \end{bmatrix}, \quad \boldsymbol{i}(\hat{\boldsymbol{v}}) \equiv \begin{bmatrix} i(\hat{v}(t_1)) \\ i(\hat{v}(t_2)) \\ \vdots \\ i(\hat{v}(t_m)) \end{bmatrix},$$
$$\boldsymbol{u} \equiv \begin{bmatrix} u(t_1) \\ u(t_2) \\ \vdots \\ u(t_M)) \end{bmatrix}. \tag{29}$$

One elegant aspect of the matrix form in (28) is the ease with which it is possible to substitute more accurate backward or forward discretization methods to replace backward-Euler. It is only necessary to replace D_{be} in (28). For example, the L-step backward difference methods [32] estimate a time derivative using several backward timepoints, as in

$$\frac{d}{dt} q(v(t_m)) \approx \sum_{j=0}^{L} \alpha_j^m q\big(v(t_{m-j})\big). \tag{30}$$

Note that for backward-Euler, $L = 1$ and

$$\alpha_0^m = \alpha_1^m = \frac{1}{h_m} \tag{31}$$

and note also that α_j^m will be independent of m if all the timesteps are equal. To substitute a two-step backward-difference formula in (28), D_{be} is replaced by D_{bd2} where

$$D_{bd2} = \begin{bmatrix} \alpha_0^1 & 0 & \dots & 0 & \alpha_2^1 & \alpha_1^1 \\ \alpha_1^2 & \alpha_0^2 & 0 & \dots & 0 & \alpha_2^2 \\ \alpha_2^3 & \alpha_1^3 & \alpha_0^3 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & & \\ 0 & \dots & 0 & \alpha_2^M & \alpha_1^M & \alpha_0^M \end{bmatrix}. \tag{32}$$

3) Trapezoidal Rule RC Example: The Trapezoidal rule is not a backward or forward difference formula but can still be used to compute steady-state solutions using a minor modification of the above finite-difference method. The Trapezoidal method is also interesting to study in this setting because of a curious property we will make clear by example.

Again consider the differential equation for the *RC* circuit from (12), repeated here reorganized and with g replacing $1/r$

$$c\frac{d}{dt} v(t) = -gv(t) - u(t). \tag{33}$$

The mth timestep of the Trapezoidal rule applied to computing the solution to (33) is

$$\frac{c}{h_m}(\hat{v}(t_m) - \hat{v}(t_{m-1})) = -\frac{1}{2}(g\hat{v}(t_m) + u(t_m) + g\hat{v}(t_{m-1}) + u(t_{m-1})) \tag{34}$$

and when used to compute a periodic steady-state solution yields the matrix equation

$$\begin{bmatrix} \frac{c}{h_1} + 0.5g & 0 & \dots & 0 & -\frac{c}{h_1} + 0.5g \\ -\frac{c}{h_2} + 0.5g & \frac{c}{h_2} + 0.5g & 0 & \dots & 0 \\ & \ddots & & & \\ 0 & \dots & 0 & -\frac{c}{h_M} + 0.5g & \frac{c}{h_M} + 0.5g \end{bmatrix} \times \begin{bmatrix} v(t_1) \\ v(t_2) \\ \vdots \\ v(t_M) \end{bmatrix} = \begin{bmatrix} 0.5(u(t_1) + u(t_M)) \\ 0.5(u(t_2) + u(t_1)) \\ \vdots \\ 0.5(+u(t_M) + u(t_{M-1})) \end{bmatrix}. \tag{35}$$

Now suppose the capacitance approaches zero, then the matrix in (35) takes on a curious property. If the number of

timesteps M, is odd, then a reasonable solution is computed. However, if the number of timesteps is even, then the matrix in (35) is singular, and the eigenvector associated with the zero eigenvalue is of the form

$$\begin{bmatrix} 1.0 \\ -1.0 \\ 1.0 \\ -1.0 \\ \vdots \\ 1.0 \\ -1.0 \end{bmatrix}. \tag{36}$$

The seasoned circuit simulation user or developer may recognize this as the periodic steady-state representation of an artifact known as the trapezoidal rule *ringing* problem [33]. Nevertheless, the fact that the ringing appears and disappears simply by incrementing the number of timesteps makes this issue the numerical equivalent of a pretty good card trick, no doubt useful for impressing one's guests at parties.

4) Jacobian for Newton's Method: Before deriving the Jacobian needed to apply Newton's method to (28), it is useful (or perhaps just very appealing to the authors) to note the simplicity with which the Kronecker product can be used to express the linear algebriac system which must be solved to compute the steady-state solution associated with the finite-difference method applied to the linear differential equation system in (4). For this case, (28) simplifies to

$$(D_{fd} \otimes C + I_M \otimes G)\hat{\boldsymbol{v}} = \boldsymbol{u} \tag{37}$$

where D_{fd} is the differentiation matrix associated with the selected finite-difference scheme.

The Jacobian for $F(\cdot)$ in (28) has structural similarities to (37) but will require the M derivative matrices $C_m = dq(\hat{v}(t_m))/dv$ and $G_m = di(\hat{v}(t_m))/dv$ for $m \in \{1, \ldots, M\}$. By first defining the $MN \times MN$ block diagonal matrices

$$\boldsymbol{C} = \begin{bmatrix} C_1 & 0 & 0 & \ldots & 0 \\ 0 & C_2 & 0 & \ldots & 0 \\ & & \ddots & & \\ & & & \ddots & \\ 0 & 0 & \ldots & 0 & C_M \end{bmatrix} \tag{38}$$

and

$$\boldsymbol{G} = \begin{bmatrix} G_1 & 0 & 0 & \ldots & 0 \\ 0 & G_2 & 0 & \ldots & 0 \\ & & \ddots & & \\ & & & \ddots & \\ 0 & 0 & \ldots & 0 & G_M \end{bmatrix} \tag{39}$$

it is possible to give a fairly compact form to represent the $MN \times MN$ Jacobian of $F(\cdot)$ in (28)

$$J_F(\boldsymbol{v}) = (D_{fd} \otimes I_N)\boldsymbol{C} + \boldsymbol{G}. \tag{40}$$

C. Basis Collocation Methods

An alternative to the finite-difference method for solving (9) is to represent the solution approximately as a weighted sum of basis functions that satisfy the periodicity constraint and then generate a system of equations for computing the weights. In circuit simulation, the most commonly used techniques for generating equations for the basis function weights are the so-called spectral collocation methods [44]. The name betrays a history of using sines and cosines as basis functions, though other bases, such as polynomials and wavelets, have found recent use [41], [45]. In spectral collocation methods, the solution is represented by the weighted sum of basis functions that exactly satisfy the differential equation, but only at a set of collocation timepoints.

The equations for spectral collocation are most easily derived if the set of basis functions have certain properties. To demonstrate those properties, consider a basis set being used to approximate a periodic function $x(t)$, as in

$$x(t) \approx \sum_{k=1}^{K} X[k]\phi_k(t) \tag{41}$$

where $\phi_k(t)$ and $X[k]$, $k \in 1, \ldots, K$ are the periodic basis functions and the basis function weights, respectively. We will assume that each of the $\phi_k(t)$'s in (41) are differentiable, and in addition we will assume the basis set must have an interpolation property. That is, it must be possible to determine uniquely the K basis function weights given a set of K sample values of $x(t)$, though the sample timepoints may depend on the basis. This

interpolation condition implies that there exists a set of K timepoints, $t_1, \ldots t_K$, such that the $K \times K$ matrix

$$\Gamma^{-1} \equiv \begin{bmatrix} \phi_1(t_1) & \ldots & \phi_K(t_1) \\ \vdots & & \vdots \\ \phi_1(t_K) & \ldots & \phi_K(t_K) \end{bmatrix} \tag{42}$$

is nonsingular and therefore the basis function coefficients can be uniquely determined from the sample points using

$$\begin{bmatrix} X_1 \\ \vdots \\ X_K \end{bmatrix} = \Gamma \begin{bmatrix} x(t_1) \\ \vdots \\ x(t_K) \end{bmatrix}. \tag{43}$$

To use basis functions to solve (9), consider expanding $q(v(t))$ in (9) as

$$q(v(t)) \approx \sum_{k=1}^{K} Q[k]\phi_k(t) \tag{44}$$

where $Q[k]$ is the N-length vector of weights for the kth basis function.

Substituting (44) in (3) yields

$$\frac{d}{dt}\left(\sum_{k=1}^{K} Q[k]\phi_k(t)\right) + i(v(t)) + u(t) \approx 0. \tag{45}$$

Moving the time derivative inside the finite sum simplifies (45) and we have

$$\sum_{k=1}^{K} Q[k]\dot{\phi}_k(t) + i(v(t)) + u(t) \approx 0 \tag{46}$$

where note that the *dot* above the basis function is used to denote the basis function time derivative.

In order to generate a system of equations for the weights, (46) is precisely enforced at M collocation points $\{t_1, \ldots, t_M\}$,

$$\sum_{k=1}^{K} Q[k]\dot{\phi}_k(t_m) + i(v(t_m)) + u(t_m) = 0 \tag{47}$$

for $m \in \{1, \ldots, M\}$. It is also possible to generate equations for the basis function weights by enforcing (46) to be orthogonal to each of the basis functions. Such methods are referred to as Galerkin methods [7], [44], [46] and have played an important role in the development of periodic steady-state methods for circuits though they are not the focus in this paper.

If the number of collocation points and the number of basis functions are equal, $M = K$, and the basis set satisfies the interpolation condition mentioned above with an $M \times M$ interpolation matrix Γ, then (47) can be recast using the Kronecker notation and paralleling (28) as

$$F(\hat{v}) = \dot{\Gamma}^{-1}\Gamma \otimes I_N \boldsymbol{q}(\hat{\boldsymbol{v}}) + \boldsymbol{i}(\hat{\boldsymbol{v}}) + \boldsymbol{u} = 0 \tag{48}$$

where I_N is the $N \times N$ identity matrix and

$$\dot{\Gamma}^{-1} \equiv \begin{bmatrix} \dot{\phi}_1(t_1) & \ldots & \dot{\phi}_K(t_1) \\ \vdots & & \vdots \\ \dot{\phi}_1(t_K) & \ldots & \dot{\phi}_K(t_K) \end{bmatrix}. \tag{49}$$

By analogy to (28), the product $\dot{\Gamma}^{-1}\Gamma$ in (48) can be denoted as a basis function associated differentiation matrix D_{bda}

$$D_{\text{bda}} = \dot{\Gamma}^{-1}\Gamma \tag{50}$$

and (48) becomes identical in form to (28)

$$F(\hat{v}) = D_{\text{bda}} \otimes I_N \boldsymbol{q}(\hat{\boldsymbol{v}}) + \boldsymbol{i}(\hat{\boldsymbol{v}}) + \boldsymbol{u} = 0. \tag{51}$$

Therefore, regardless of the choice of the set of basis functions, using the collocation technique to compute the basis function weights implies the resulting method is precisely analgous to a finite-difference method with a particular choice of discretization matrix. For the backward-difference methods described above, the $M \times M$ matrix D_{fd} had only order M nonzeros, but as we will see in the Fourier example that follows that for spectral collocation methods the D_{bda} matrix is typically dense.

Since basis collocation methods generate nonlinear systems of equations that are structurally identical to those generated by the finite-difference methods, when Newton's is used to solve (51), the formula for the required $MN \times MN$ Jacobian of $F(\cdot)$ in (51) follows from (40) and is given by

$$J_F((\boldsymbol{v})) = (D_{\text{bda}} \otimes I_N)\boldsymbol{C} + \boldsymbol{G} \tag{52}$$

where $\boldsymbol{C}$ and $\boldsymbol{G}$ are as defined in (38) and (39).

1) Fourier Basis Example: If a sinusoid is the input to a system of differential equations generated by a linear time-invariant circuit, then the associated periodic steady-state solution will be a scaled and phase-shifted sinusoid of the same frequency. For a mildly nonlinear circuit with a sinusoidal input, the solution is often accurately represented by a sinusoid and a few of its harmonics. This observation suggests that a truncated Fourier series will be an efficient basis set for solving periodic steady-state problems of mildly nonlinear circuits.

To begin, any square integrable T-periodic waveform $x(t)$ can be represented as a Fourier series

$$x(t) = \sum_{k=-\infty}^{k=\infty} X[k] e^{j2\pi f_k t} \tag{53}$$

where $f_k = kt/T$ and

$$X[k] = \frac{1}{T} \int_{-T/2}^{T/2} x(t) e^{-j2\pi f_k t} dt. \tag{54}$$

If $x(t)$ is both periodic and is sufficiently smooth, (e.g., infinitely continuously differentiable), then $X[k] \to 0$ exponentially fast with increasing k. This implies $x(t)$ can be accurately represented with a truncated Fourier series, that is $\hat{x}(t) \approx x(t)$ where $\hat{x}(t)$ is given by the truncated Fourier series

$$\hat{x}(t) = \sum_{k=-K}^{k=K} \hat{X}[k] e^{j2\pi f_k t} \tag{55}$$

where the number of harmonics K is typically fewer than one hundred. Note that the time derivative of $\hat{x}(t)$ is given by

$$\frac{d}{dt}\hat{x}(t) = \sum_{k=-K}^{k=K} \hat{X}[k] j2\pi f_k e^{j2\pi f_k t}. \tag{56}$$

If a truncated Fourier series is used as the basis set when approximately solving (9), the method is referred to as harmonic balance [6] or a Fourier spectral method [17].

If the $M = 2K + 1$ collocation timepoints are uniformly distributed throughout a period from $-(T/2)$ to $T/2$, as in $t_m = ((m - (K + 1/2))/M)(1/T)$, then the associated interpolation matrix Γ_F is just the discrete Fourier transform and Γ_F^{-1} is the inverse discrete Fourier transform, each of which can be applied in order $M \log M$ operations using the fast Fourier transform and its inverse. In addition, $\dot{\Gamma}_F^{-1}$, representing the time derivative of the series representation, is given by

$$\dot{\Gamma}_F^{-1} = \Gamma_F^{-1} \Omega \tag{57}$$

where Ω is the diagonal matrix given by

$$\Omega \equiv \begin{bmatrix} j2\pi f_K & & & \\ & j2\pi f_{K-1} & & \\ & & \ddots & \\ & & & j2\pi f_{-K} \end{bmatrix}. \tag{58}$$

The Fourier basis collocation method generates a system of equations of the form (51), where

$$D_F = \Gamma^{-1} \Omega \Gamma \tag{59}$$

is the differentiation matrix. The weights for this spectral differentiation matrix for the case of $T = 17$, $M = 17$, and at timepoint $t_9 = 9$ are plotted in Fig. 6. Note that the weights at t_8 and t_{10} are approximately -1 and 1, respectively, so spectral differentiation is somewhat similar to a central-difference scheme in which

$$\frac{d}{dt}x(t_9) \approx \frac{x(t_{10}) - x(t_8)}{t_{10} - t_8}. \tag{60}$$

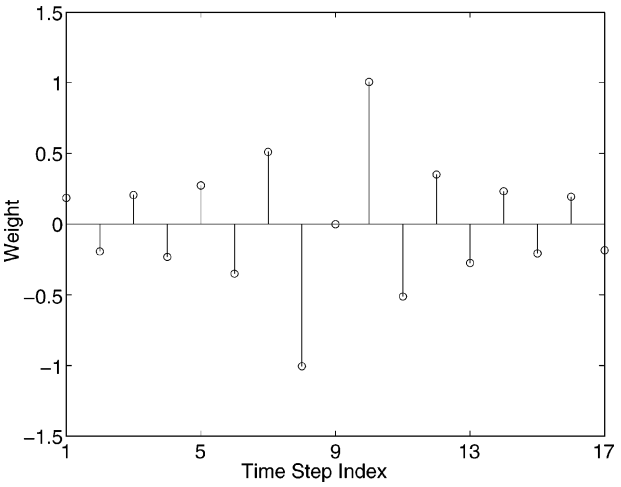

Fig. 6. ***Harmonic balance discretization weights for t_9 where $T = 17$ and $M = 17$.***

The connection between spectral differentiation and standard differencing schemes can be exploited when developing fast methods for computing solutions to (28) and (51), a point we will return to subsequently.

The error analysis of spectral-collocation methods can be found in [17] and [19]. Also, many implementations of harmonic balance in circuit simulators use spectral Galerkin methods rather than collocation schemes [7], and if a small number of harmonics are used, Galerkin spectral methods can have superior accuracy and often lead to nonlinear systems of equations that are more easily solved with Newton's method [47].

D. Shooting Methods

The numerical procedure for solving the state transition function based periodic steady-state formulation in (11) is most easily derived for a specific discretization scheme and then generalized. Once again, consider applying the simple backward-Euler algorithm to (3). Given any $\hat{v}(t_0)$, the nonlinear equation

$$\frac{q(\hat{v}(t_1)) - q(\hat{v}(t_0))}{h_1} + i(\hat{v}(t_1)) + u(t_1) = 0 \tag{61}$$

can be solved, presumably using a multidimensional Newton method, for $\hat{v}(t_1)$. Then, $\hat{v}(t_1)$ can be used to solve

$$\frac{q(\hat{v}(t_2)) - q(\hat{v}(t_1))}{h_2} + i(\hat{v}(t_2)) + u(t_2) = 0 \tag{62}$$

for $\hat{v}(t_2)$. This procedure can be continued, effectively integrating the differential equation one timestep at a time until $\hat{v}(t_m)$ has been computed. And since the nonlinear equations are solved at each timestep, $\hat{v}(t_m)$ is an implicitly defined algebraic function of $\hat{v}(t_0)$. This implicitly defined function is a numerical approximation to the state-transition function $\Phi(\cdot)$ described in the previous section. That is,

$$\hat{v}(t_m) = \hat{\Phi}(\hat{v}(t_0), t_m) \approx \Phi(\hat{v}(t_0), t_m). \tag{63}$$

The discretized version of the state-transition function-based periodic steady-state formulation is then

$$F(\hat{v}(t_m)) \equiv \hat{v}(t_M) - \hat{\Phi}(\hat{v}(t_M), t_M) = 0. \tag{64}$$

Using (64) to compute a steady-state solution is often referred to as a shooting method, in which one guesses a periodic steady state and then *shoots* forward one period with the hope of arriving close to the guessed initial state. Then, the difference between the initial and final states is used to correct the initial state, and the method *shoots* forward another period. As is commonly noted in the numerical analysis literature, this shooting procedure will be disasteriously ineffective if the first guess at a periodic steady state excites rapidly growing unstable behavior in the nonlinear system [18], but this is rarely an issue for circuit simulation. Circuits with such unstable "modes" are unlikely to be useful in practice, and most circuit designers using periodic steady-state analysis have already verified that their designs are quite stable.

The state correction needed for the shooting method can be performed with Newton's method applied to (64), in which case the correction equation becomes

$$\left[I_N - J_{\hat{\Phi}}\left(\hat{v}^k(t_M), T\right)\right]\left[\hat{v}^{k+1}(t_M) - \hat{v}^k(t_M)\right] = -F_{sh}\left(\hat{v}^k(t_M)\right) \tag{65}$$

where k is the Newton iteration index, I_N is $N \times N$ identity matrix, and

$$J_{\hat{\Phi}}(v, T) \equiv \frac{d}{dv}\hat{\Phi}(v, T) \tag{66}$$

is referred to as discretized sensitivity matrix.

To complete the description of the shooting-Newton method, it is necessary to present the procedure for computing $\hat{\Phi}(v, T)$ and $J_{\hat{\Phi}}(v, T)$. As mentioned above, computing the approximate state transition function is equivalent to solving the backward-Euler equations as in (24) one timestep at a time. Solving the backward-Euler equations is usually accomplished using an *inner* Newton iteration, as in

$$\left[\frac{C_{mkl}}{h_m} + G_{mkl}\right]\left(\hat{v}^{k,(l+1)}(t_m) - \hat{v}^{k,l}(t_m)\right) = -\frac{1}{h_m} \times \left(q\left(\hat{v}^{k,l}(t_m)\right) - q\left(\hat{v}^{k,l}(t_m)\right)\right) - i\left(\hat{v}^{k,l}(t_m)\right) - u(t_m) \tag{67}$$

where m is the timestep index, k is the shooting-Newton iteration index, l is the inner Newton iteration index, $C_{mkl} = (dq(\hat{v}^{k,l}(t_m))/dv$ and $G_{mkl} = (di(\hat{v}^{k,l}(t_m))/dv$. Sometimes, there are just too many indices.

To see how to compute $J_{\hat{\Phi}}(v, T)$ as a by-product of the Newton method in (67), let $l = *$ denote the inner Newton iteration index which achieves sufficent convergence and let $\hat{v}^{k,*}(t_m)$ denote the associated inner Newton converged solution. Using this notation

$$\frac{q(\hat{v}^{k,*}(t_m)) - q(\hat{v}^{k,*}(t_{m-1}))}{h_m} + i\left(\hat{v}^{k,*}(t_m)\right) + u(t_m) = \epsilon^m \tag{68}$$

where the left-hand side of (68) is almost zero, so that ϵ^m is bounded by the inner Newton convergence tolerance.

Implicitly differentiating (68) with respect to v, and assuming that ϵ^m is independent of v, results in

$$\left[\frac{C_{mk*}}{h_m} + G_{mk*}\right] \frac{d\hat{v}^{k,*}(t_m)}{dv} = \frac{C_{(m-1)k*}}{h_m} \frac{d\hat{v}^{k,*}(t_{m-1})}{dv} \tag{69}$$

where it is usually the case that the matrices C_{mk*}/h_m and G_{mk*} are available as a by-product of the inner Newton iteration.

Recursively applying (69), with the initial value $\hat{v}^{k,*}(t_0) = v$, yields a product form for the Jacobian

$$J_{\hat{\Phi}}(v, t_M) = \prod_{m=1}^{M} \left[\frac{C_{mk*}}{h_m} + G_{mk*}\right]^{-1} \frac{C_{(m-1)k*}}{h_m} \tag{70}$$

where the notation $\prod_{m=1}^{M}$ indicates the M-term product rather than a sum [15].

1) Linear Example: If a fixed timestep backward-Euler discretization scheme is applied to (4), then

$$\frac{C}{h}(\hat{v}(t_m) - \hat{v}(t_{m-1})) + Gv(t_m) + u(t_m) = 0 \tag{71}$$

for $m \in \{1, \ldots, M\}$, where periodicity implies that $\hat{v}(t_M) = \hat{v}(t_0)$. Solving for $\hat{v}(t_M)$ yields a linear system of equations

$$[I_N - J_\Phi]\hat{v}(t_M) = b \tag{72}$$

where J_Φ is the derivative of the state transition function, or the sensitivity matrix, and is given by

$$J_\Phi = \left(\left(\frac{C}{h} + G\right)^{-1} \frac{C}{h}\right)^M \tag{73}$$

and the right-hand side N-length vector b is given by

$$b = \sum_{m=1}^{M} \left(\left(\frac{C}{h} + G\right)^{-1} \frac{C}{h}\right)^m \left(\frac{C}{h} + G\right)^{-1} u(t_{M-m}). \tag{74}$$

It is interesting to compare (72) and (74) to (16), note how the fixed-timestep backward-Euler algorithm is approximating the matrix exponential in (73) and the convolution integral in (74).

2) Comparing to Finite-Difference Methods: If Newton's method is used for both the shooting method, as in (65), and for the finite difference method, in which case the Jacobian is (40), there *appears* to be an advantage for the shooting-Newton method. The shooting-Newton method is being used to solve a system of N nonlinear equations, whereas the finite-difference-Newton method is being used to solve an NM system of nonlinear equations. This advantage is not as significant as it seems, primarily because computing the sensitivity matrix according to (70) is more expensive than computing the finite-difference Jacobian. In this section, we examine the backward-Euler discretized equations to show that solving a system the shooting method Jacobian, as in (65), is nearly equivalent to solving a *preconditioned* system involving the finite-difference method Jacobian, in (40).

To start, let L be the $NM \times NM$ the block lower bidiagonal matrix given by

$$L \equiv \begin{bmatrix} \frac{C_1}{h_1} + G_1 & & & \\ -\frac{C_1}{h_2} & \frac{C_2}{h_2} + G_2 & & \\ & \ddots & \ddots & \\ & & -\frac{C_{M-1}}{h_M} & \frac{C_M}{h_M} + G_M \end{bmatrix} \tag{75}$$

and define B as the $NM \times NM$ matrix with a single nonzero block

$$B \equiv \begin{bmatrix} 0 & \ldots & 0 & \frac{C_M}{h_1} \\ & 0 & & \\ & \ddots & \ddots & \\ & & 0 & 0 \end{bmatrix} \tag{76}$$

where C_m and G_m are the $N \times N$ matrices which denote $dq(\hat{v}(t_j))/dv$ and $di(\hat{v}(t_j))/dv$, respectively.

The matrix L defined in (75) is block lower bidiagonal, where the diagonal blocks have the same structure as the single timestep Jacobian in (67). It then follows that the cost of applying L^{-1} is no more than computing one Newton iteration at each of M timesteps. One simply factors the diagonal blocks of L and backsolves. Formally, the result can be written as

$$(I_{NM} - L^{-1}B)(\tilde{\boldsymbol{v}}^{k+1} - \tilde{\boldsymbol{v}}^k) = -L^{-1}\boldsymbol{F}(\tilde{\boldsymbol{v}}^k) \tag{77}$$

though L^{-1} would never be explicitly computed. Here, I_{NM} is the $NM \times NM$ identity matrix.

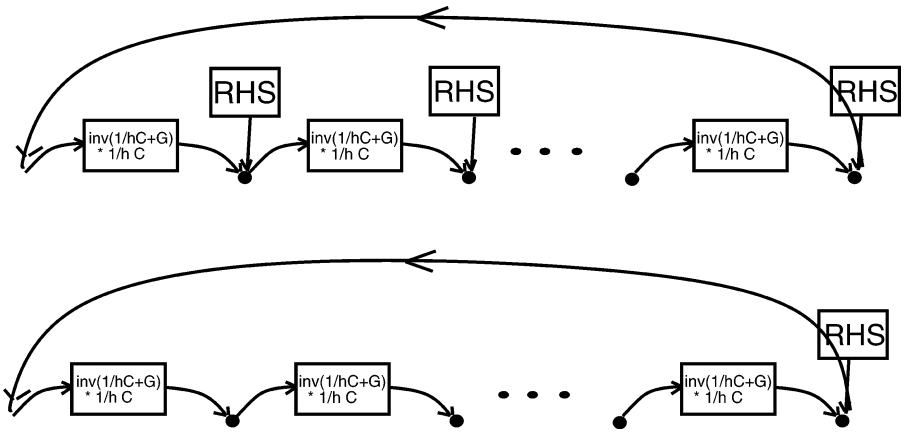

Fig. 7. ***Diagram of solving backward-Euler discretized finite-difference-Newton iteration equation (top picture) and solving shooting-Newton iteration equation (bottom picture). Note that for shooting method the RHS contribution at intermediate timesteps is zero, due to the inner newton iteration.***

Examining (77) reveals an important feature, that $L^{-1}B$ is an $NM \times NM$ matrix whose only nonzero entries are *in the last N columns*. Specifically,

$$(I_{NM} - L^{-1}B) = \begin{bmatrix} I_N & 0 & 0 & \dots & 0 & -P_1 \\ 0 & I_N & 0 & \dots & 0 & -P_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & -P_{M-2} \\ \vdots & \ddots & \ddots & \ddots & I_N & -P_{M-1} \\ 0 & \dots & \dots & \dots & 0 & I_N - P_M \end{bmatrix} \tag{78}$$

where the $N \times N$ matrix P_M is the $N(M-1)+1$ through NM rows of the last N columns of $L^{-1}B$. This bordered-block diagonal form implies that $\tilde{\boldsymbol{v}}^{k+1} - \tilde{\boldsymbol{v}}^{k}$ in (77) can be computed in three steps. The first step is to compute P_M, the second step is to use the computed P_M to determine the last N entries in $\tilde{\boldsymbol{v}}^{k+1} - \tilde{\boldsymbol{v}}^{k}$. The last step in solving (77) is to compute the rest of $\tilde{\boldsymbol{v}}^{k+1} - \tilde{\boldsymbol{v}}^{k}$ by backsolving with L.

The close relation between solving (77) and (65) can now be easily established. If L and B are formed using C_{mk*} and G_{mk*} as defined in (69), then by explicitly computing $L^{-1}B$ it can be shown that the shooting method Jacobian $J_\Phi(\hat{v}^k(t_0), t_M)$ is equal to P_M. The importance of this observation is that solving the shooting-Newton update equation is nearly computationally equivalent to solving a preconditioned finite-difference-Newton update (77). The comparison can be made precise if $q(v)$ and $i(v)$ are linear, so that the C and G matrices are independent of v, then the $\hat{v}^{k+1}(t_M) - \hat{v}^{k}(t_M)$ produced by the kth iteration of Newton's method applied to the finite-difference formulation will be identical to the $\hat{v}^{k+1}(t_M) - \hat{v}^{k}(t_M)$ produced by solving (65).

An alternative interpretation of the connection between shooting-Newton and finite-difference-Newton methods is to view the shooting-Newton method as a two-level Newton method [10] for solving (24). In the shooting-Newton method, if $\hat{v}(t_m)$ is computed by using an inner Newton method to solve the nonlinear equation F_m at each timestep, starting with $\hat{v}(t_o) = \hat{v}(t_M)$, or equivalently if $\hat{v}(t_m)$ is computed by evaluating $\hat{\Phi}(\hat{v}(t_M), t_m)$, then $F_i(\hat{v})$ in (24) will be nearly zero for $i \in \{1, \dots, M-1\}$, regardless of the choice of $\hat{v}(t_M)$. Of course, $\hat{\Phi}(\hat{v}(t_M), t_M)$ will not necessarily be equal to $\hat{v}(t_M)$; therefore, $F_M(\hat{v})$ will not be zero, unless $\hat{v}(t_M) = \hat{v}(t_0)$ is the right initial condition to produce the periodic steady state. In the shooting-Newton method, an *outer* Newton method is used to correct $\hat{v}(t_M)$. The difference between the two methods can then be characterized by differences in inputs to the linearized systems, as diagrammed in Fig. 7.

3) More General Shooting Methods: Many finite-difference methods can be converted to shooting methods, but only if the underlying time discretization scheme treats the periodicity constraint by "wrappping around" a single final timepoint. For discretization schemes which satisfy this constraint, the $M \times M$ periodic differentiation matrix, D, is lower triangular except for a single entry in the upper right-hand corner. As an example, the differentiation matrix D_{be} in (27) is lower bidiagonal except for a single entry in the upper right-hand corner. To be more precise, if

$$D_{i,j} = 0, \quad j > i, i \neq 1, j \neq M \tag{79}$$

consider separating D into its lower triangular part, D^L, and a single entry $D_{1,M}$. The shooting method can then

be compactly described as solving the nonlinear algebriac system

$$F(v_T) = v_T - \left(\vec{e}_M^T \otimes I_N\right)\hat{\boldsymbol{v}} = 0 \tag{80}$$

for the N-length vector v_T, where $\vec{e}_M$ an M-length vector of zeros except for unity in the Mth entry as in

$$\vec{e}_M = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{81}$$

and $\hat{\boldsymbol{v}}$ is an MN-length vector which is an implicitly defined function of v_T. Specifically, $\hat{\boldsymbol{v}}$ is defined as the solution to the system of MN nonlinear equations

$$D^L \otimes I_N \boldsymbol{q}(\hat{\boldsymbol{v}}) + \boldsymbol{i}(\hat{\boldsymbol{v}}) + \boldsymbol{u} + (D_{1,M}\vec{e}_1) \otimes q(v_T) = 0 \tag{82}$$

where $\vec{e}_1$ is the M-length vector of zeros except for unity in the first entry. Perhaps the last Kronecker term generating an NM-length vector in (82) suggests that the authors have become carried away and overused the Kronecker product, as

$$(D_{1,M}\vec{e}_1) \otimes q(v_T) = \begin{bmatrix} D_{1,M}q(v_T) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{83}$$

is more easily understood without the use of the Kronecker product, but the form will lend some insight, as will be clear when examining the equation for the Jacobian.

If Newton's method is used to solve (80), then note an inner Newton's method will be required to solve (82). The Jacobian associated with (80) can be determined using implicit differentiation and is given by

$$J_{\text{shoot}}(v_T) = I_N + \left(\vec{e}_M^T \otimes I_N\right)\left((D^L \otimes I_N)\boldsymbol{C} + \boldsymbol{G}\right)^{-1} \times \left(D_{1,M}\vec{e}_1 \otimes \frac{\partial q(v_T)}{\partial v}\right) \tag{84}$$

where $\boldsymbol{C}$ and $\boldsymbol{G}$ are as defined in (38) and (39), and $((D^L \otimes I_N)\boldsymbol{C} + \boldsymbol{G}$ is a block lower triangular matrix whose inverse is inexpensive to apply.

IV. FAST METHODS

As described in the previous section, the finite-difference basis-collocation and shooting methods all generate systems of nonlinear equations that are typically solved with the multidimensional Newton's method. The standard approach for solving the linear systems that generate the Newton method updates, as in (21), is to use sparse matrix techniques to form and factor an explicit representation of the Newton method Jacobian [6], [12]. When this standard approach is applied to computing the updates for Newton's method applied to any of the periodic steady-state methods, the cost of forming and factoring the explicit Jacobian grows too fast with problem size and the method becomes infeasible for large circuits.

It is important to take note that we explicitly mention the cost of forming the explicit Jacobian. The reason is that for all the steady-state methods described in the previous section, the Jacobians were given in an implicit form, as a combination of sums, products, and possibly Kronecker products. For the backward-Euler based shooting method, the $N \times N$ Jacobian J_{shoot}, is

$$J_{\text{shoot}} = I_N - \prod_{m=1}^{M}\left(\left[\frac{C_m}{h_m} + G_m\right]^{-1}\frac{C_m}{h_m}\right) \tag{85}$$

and is the difference between the identity matrix and a product of M $N \times N$ matrices, each of which must be computed as the product of a scaled capacitance matrix and the inverse of a weighted sum of a capacitance and a conductance matrix. The $NM \times NM$ finite-difference or basis-collocation Jacobians, denoted generally as J_{fdbc}, are also implicitly defined. In particular

$$J_{fdbc} = (D \otimes I_N)\boldsymbol{C} + \boldsymbol{G} \tag{86}$$

is constructed from an $M \times M$ differentiation matrix D, a set of M $N \times N$ capacitance and conductance matrices which make up the diagonal blocks of $\boldsymbol{C}$ and $\boldsymbol{G}$, and an $N \times N$ identity matrix.

Computing the explicit representation of the Jacobian from either (85) or (86) is expensive, and therefore any fast method for these problems must avoid explicit Jacobian construction. Some specialized methods with this property have been developed for shooting methods based on sampling results of multiperiod transient integration [25], and specialized methods have been developed for Fourier basis collocation methods which use Jacobian approximations [36]. Most of these methods have been abandoned in favor of solving the Newton iteration equation using a Krylov subspace method, such as the generalized minimum residual method (GMRES) and the quasi-minimum residual method (QMR) [9], [27].

When combined with good *preconditioners*, to be discussed subsequently, Krylov subspace methods reliably solve linear systems of equations but do not require an explicit system matrix. Instead, Krylov subspace methods only require matrix-vector products, and such products can be computed efficiently using the implicit forms of the Jacobians given above.

Claiming that Krylov subspace methods do not require explicit system matrices is somewhat misleading, as these methods converge quite slowly without a good *preconditioner*, and preconditioners often require that at least some of the system matrix be explicitly represented. In the following section, we describe one of the Krylov subspace methods, GMRES, and the idea of preconditioning. In the subsections that follow we address the issue of preconditioning, first demonstrating why Krylov subspace methods converge rapidly for shooting methods even without preconditioning, then describing lower triangular and averaging based preconditioning for basis-collocation and finite-difference methods, drawing connections to shooting methods when informative.

A. Krylov Subspace Methods

Krylov subspace methods are the most commonly used iterative technique for solving Newton iteration equations for periodic steady-state solvers. The two main reasons for the popularity of this class of iterative methods are that only matrix-vector products are required, avoiding explicitly forming the system matrix, and that convergence is rapid when preconditioned effectively.

As an example of a Krylov subspace method, consider the generalized minimum residual algorithm, GMRES [9]. A simplified version of GMRES applied to solving a generic problem is given as follows.

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 $k = k + 1$.
}

Krylov subspace methods converge rapidly when applied to matrices which are not too nonnormal and whose eigenvalues are contained in a small number of tight clusters [44]. Therefore, Krylov subspace methods converge rapidly when applied to matrices which are small perturbations from the identity matrix, but can, as an example, converge remarkably slowly when applied to a diagonal matrix whose diagonal entries are widely dispersed. In many cases, convergence can be accelerated by replacing the original problem with a *preconditioned* problem

$$PAX = Pb \tag{87}$$

where A and b are the original system's matrix and right-hand side, and P is the preconditioner. Obviously, the preconditioner that best accelerates the Krylov subspace method is $P = A^{-1}$, but were such a preconditioner available, no iterative method would be needed.

B. Fast Shooting Methods

Applying GMRES to solving the backward-Euler discretized shooting Newton iteration equation is straightforward, as multiplying by the shooting method Jacobian in (85) can be accomplished using the simple M-step algorithm as follows.

Computing $p^k = J_{\text{shoot}} p^{k-1}$
Initialize $p_{\text{temp}} = p^{k-1}$
For $k = 1$ to M {
  Solve $((C_m/h_m) + G_m)p^k = (C_m/h_m)p_{\text{temp}}$
  Set $p_{\text{temp}} = p^k$
}
Finalize $p^k = p^{k-1} - p^k$

The $N \times N$ C_m and G_m matrices are typically quite sparse, so each of the M matrix solutions required in the above algorithm require roughly order N operations, where $order(\cdot)$ is used informally here to imply proportional growth. Given the order N cost of the matrix solution in this case, computing the entire matrix-vector product requires order MN operations.

Note that the above algorithm can also be used N times to compute an explicit representation of J_{shoot}, generating the explicit matrix one column at a time. To compute the ith column, set $p^{k-1} = \vec{e}_i$, where $\vec{e}_i$ can be thought of as the ith unit vector or the ith column of the $N \times N$ identity matrix. Using this method, the cost of computing an explicit representation of the shooting method Jacobian requires order MN^2 operations, which roughly equals the cost of performing N GMRES iterations.

When applied to solving the shooting-Newton iteration equation, GMRES and other Krylov subspace methods converge surprisingly rapidly *without preconditioning* and typically require many fewer than N iterations to achieve sufficient accuracy. This observation, combined with the above analysis, implies that using a matrix-implicit GMRES algorithm to compute the shooting-Newton update will be far faster than computing the explicit shooting-Newton Jacobian.

In order to develop some insight as to why GMRES converges so rapidly when applied to matrices like the shooting method Jacobian in (85), we consider the

linear problem (4) and assume the C matrix is invertible so that $A = C^{-1}G$ is defined as in (15). For this linear case and a steady-state period T, the shooting method Jacobian is approximately

$$J_{\text{shoot}} \approx I - e^{-AT}. \tag{88}$$

Since eigenvalues are continuous functions of matrix elements, any eigenproperties of $I - e^{-AT}$ will roughly hold for J_{shoot}, provided the discretization is accurate. In particular, using the properties of the matrix exponential implies

$$\text{eig}(J_{\text{shoot}}) \approx \text{eig}(I - e^{-AT}) = 1 - e^{-\lambda_i T} \quad i \in 1, \ldots n \tag{89}$$

where λ_i is the ith eigenvalue of A, or in circuit terms, the inverse of the ith time constant. Therefore, if all but a few of the circuit time constants are smaller than the steady-state period T, then $e^{-\lambda_i T} \ll 1$ and the eigenvalues of J_{shoot} will be tightly clustered near one. That there might be a few "outliers" has a limited impact on Krylov subspace method convergence.

As a demonstration example, consider applying GMRES to solving the shooting method matrix associated a 500 node *RC* line circuit as in Fig. 3, with one farad capacitors and 1-Ω resistors. The time constants for the circuit vary from tenths of a second to nearly 30 000 s and are plotted in Fig. 8. The error versus iteration for the GMRES algorithm is plotted in Fig. 9 for solving the periodic steady-state equation for the *RC* line with three different periods. The fastest converging case is when the period is 10 000 s, as only a few time constants are larger than the period. The convergence of GMRES is slower when the period is 1000, as more of the time constants are larger than the period. And as one might have predicted, the GMRES algorithm requires dozens of iterations when the period is 100, as now there are then dozens of time constants much longer than the period.

1) Results: In this section, we present experimental results for the performance of three methods for solving the shooting-Newton update equations: direct factorization or Gaussian elimination, explicit GMRES, and matrix-implicit GMRES.

Table 1 contains a comparison of the performance of the three equation solvers in an implementation of a shooting-Newton method in a prototype circuit simulator. The test examples 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, explicit GMRES, and the matrix-implicit form. All the results were obtained on a HP712/80

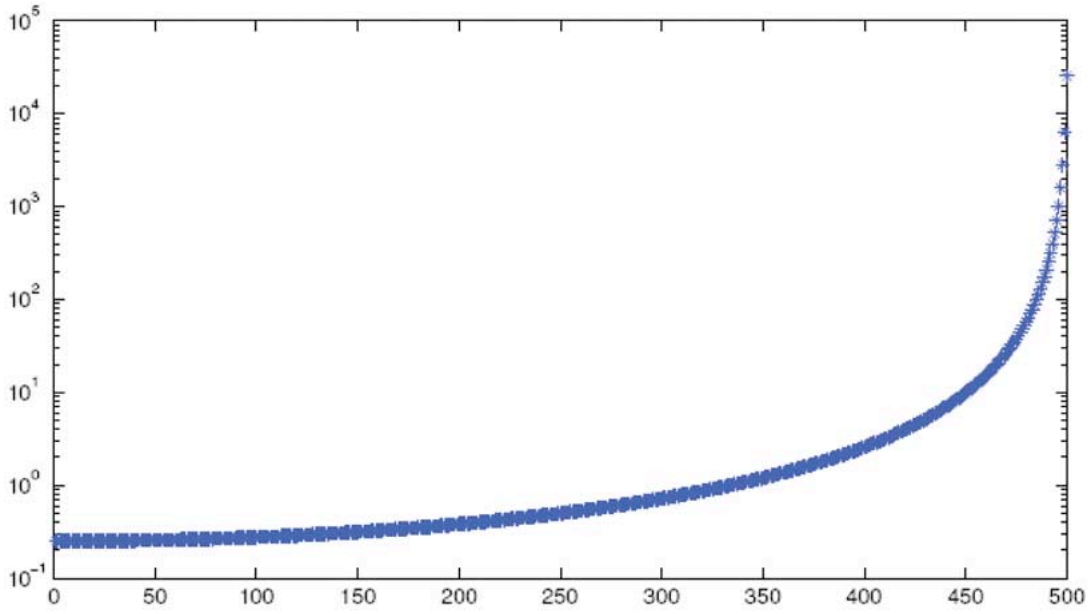

Fig. 8. *Time constants for 500 node RC line.*

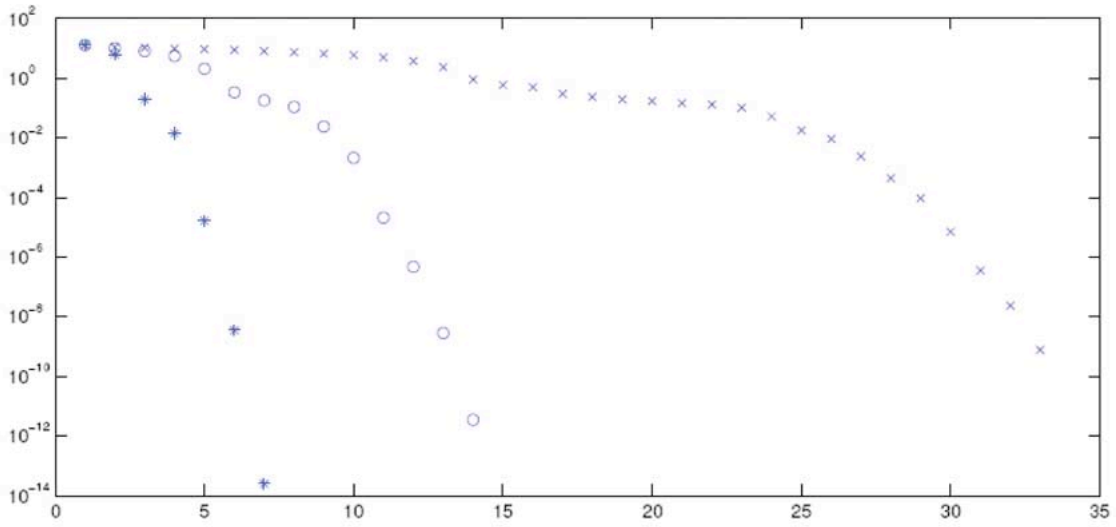

Fig. 9. *Error versus iteration for GMRES, × for 100-s period, o for 1000-s period, and ∗ for a 10 000-s period.*

workstation. The seventh column demonstrates the effectiveness of the matrix-implicit approach, listing the speedup obtained with respect to the Gaussian-elimination method. Note that the speed-up over the explicit GMRES algorithm would be similar for the size examples examined.

2) Lower Triangular Preconditioners: The fast shooting methods can be extended to any finite-difference discretization method that admits a shooting method, but the simpliest approach to describing the extension is to first consider a lower triangular preconditioner for a finite-difference or basis-collocation method. Given a differentiation matrix D, consider separating D into a lower triangular matrix D^L and strictly upper triangular matrix D^U where

$$D^L_{i,j} = 0, \quad j > i \tag{90}$$
$$D^L_{i,j} = D_{i,j}, \quad j \leq i \tag{91}$$

Table 1 Comparison of Different Shooting Method Schemes

circuit	eqns	it	GE	GMRES	MI	GE/MI
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

and

$$D^U_{i,j} = 0, \quad j \leq i \tag{92}$$
$$D^U_{i,j} = D_{i,j}, \quad j > i. \tag{93}$$

Rewriting the finite-difference or basis-collocation Jacobian as

$$J_{fdbc} = \left((D^L + D^U) \otimes I_N\right)\boldsymbol{C} + \boldsymbol{G} \tag{94}$$

suggests a preconditioner after noting that

$$(D^L \otimes I_N)\boldsymbol{C} + \boldsymbol{G} \tag{95}$$

is a block lower triangular matrix whose inverse is easily applied. Using the inverse of (95) as a preconditioner yields

$$J_{\text{precond}} = I_{NM} + \left((D^L \otimes I_N)\boldsymbol{C} + \boldsymbol{G}\right)^{-1}\left((D^U \otimes I_N)\boldsymbol{C} + \boldsymbol{G}\right). \tag{96}$$

As noted for backward-Euler in (78), if D^U has its only nonzero at $D_{1,M}$ (only one point wraps around), then the

preconditioned system will have a bordered block diagonal form

$$J_{\text{precond}} = \begin{bmatrix} I_N & 0 & 0 & \dots & 0 & -P_1 \\ 0 & I_N & 0 & \dots & 0 & -P_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & -P_{M-2} \\ \vdots & \ddots & \ddots & \ddots & I_N & -P_{M-1} \\ 0 & \dots & \dots & \dots & 0 & I_N - P_M \end{bmatrix} \tag{97}$$

where the $N \times N$ matrix $-P_M$ is the $N(M-1)+1$ through NM rows of the last N columns of

$$\left((D^L \otimes I_N)\boldsymbol{C} + \boldsymbol{G}\right)^{-1}\left((D^U \otimes I_N)\boldsymbol{C} + \boldsymbol{G}\right). \tag{98}$$

In particular, $I_N - P_M$ will be precisely the general shooting method matrix given in (84).

In the case when the differencing scheme admits a shooting method, the preconditioned system will usually have its eigenvalues tightly clustered about one. To see this, consider the structure of (97). Its upper block diagonal form implies that all of its eigenvalues are either one, due to the diagonal identity matrix blocks, or the eigenvalues of the shooting method matrix, which have already been shown to cluster near one.

3) Averaging Preconditioners: Before describing preconditioners for the general finite-difference or basis-collocation Jacobians in (40) or (52), we consider finite-difference or basis-collocation applied to the linear problem in (4). The periodic steady-state solution can be computed by solving the $MN \times MN$ linear system

$$(D \otimes C + I_M \otimes G)\hat{\boldsymbol{v}} = \boldsymbol{u} \tag{99}$$

where D is the differentiation matrix for the selected method. Explicitly forming and factoring $(D \otimes C + I_M \otimes G)$ can be quite computationally expensive, even though C and G are typically extremely sparse. The problem is that the differentation matrix D can be dense and the matrix will fill in substantially during sparse factorization.

A much faster algorithm for solving (99) which avoids explicitly forming $(D \otimes C + I_M \otimes G)$ can be derived by making the modest assumption that D is diagonalizable as in

$$D = S\lambda S^{-1} \tag{100}$$

where λ is the $M \times M$ diagonal matrix of eigenvalues and S is the $M \times M$ matrix of eigenvectors [41]. Using the eigendecompostion of D in (99) leads to

$$\left((S\lambda S^{-1}) \otimes C + (SI_M S^{-1}) \otimes G\right)\hat{\boldsymbol{v}} = \boldsymbol{u}. \tag{101}$$

Using the Kronecker product property [29]

$$(AB) \otimes (CD) = (A \otimes C)(B \otimes D) \tag{102}$$

and then factoring common expressions in (101) yields

$$\left((S \otimes I_N)(\lambda \otimes C + I_M \otimes G)(S^{-1} \otimes I_N)\right)\hat{\boldsymbol{v}} = \boldsymbol{u}. \tag{103}$$

Solving (103) for $\hat{\boldsymbol{v}}$

$$\hat{\boldsymbol{v}} = \left((S^{-1} \otimes I_N)^{-1}(\lambda \otimes C + I_M \otimes G)^{-1}(S \otimes I_N)^{-1}\right)\boldsymbol{u}. \tag{104}$$

The Kronecker product property that if A and B are invertible square matrices

$$(A \otimes B)^{-1} = (A^{-1} \otimes B^{-1}) \tag{105}$$

can be used to simplify (104)

$$\hat{\boldsymbol{v}} = \left((S \otimes I_N)(\lambda \otimes C + I_M \otimes G)^{-1}(S^{-1} \otimes I_N)\right)\boldsymbol{u} \tag{106}$$

where $(\lambda \otimes C + I_M \otimes G)^{-1}$ is a block diagonal matrix given by

$$\begin{bmatrix} (\lambda_1 C + G)^{-1} & 0 & 0 & \dots & 0 \\ 0 & (\lambda_2 C + G)^{-1} & 0 & \dots & 0 \\ & & \ddots & & \\ & & & \ddots & \\ 0 & 0 & \dots & 0 & (\lambda_M C + G)^{-1} \end{bmatrix}. \tag{107}$$

To compute $\hat{\boldsymbol{v}}$ using (106) requires a sequence of three matrix-vector multiplications. Multiplying by $(S^{-1} \otimes I_N)$ and $(S \otimes I_N)$ each require NM^2 operations. As (107) makes clear, multiplying by $(\lambda \otimes C + I_M \otimes G)^{-1}$ is M times the cost of applying $(\lambda_1 C + G)^{-1}$, or roughly order MN

operations as C and G are typically very sparse. The resulting solution algorithm therefore requires

$$\text{order}(M^3) + 2NM^2 + \text{order}(MN) \tag{108}$$

operations. The first term in (108) is the cost of eigendecomposing D, the second term is the cost of multiplying by $(S^{-1} \otimes I_N)$ and $(S \otimes I_N)$, and third term is associated with the cost of factoring and solving with the sparse matrices $(\lambda_m C + G)$. The constant factors associated with the third term are large enough that it dominates unless the number of timepoints, M, is quite large.

It should be noted that if D is associated with a periodized fixed timestep multistep method, or with a basis-collocation method using Fourier series, then D will be circulant. For circulant matrices, S and S^{-1} will be equivalent to the discrete Fourier transform matrix and its inverse [28]. In these special cases, multiplication by S and S^{-1} can be performed in order $M\text{log}M$ operations using the forward and inverse fast Fourier transform, reducing the computational cost of (106) to

$$\text{order}(M\text{log}M) + \text{order}(NM\text{log}M) + \text{order}(MN) \tag{109}$$

which is a substantial improvement only when the number of discretization timepoints or basis functions is quite large.

4) Extension to the Nonlinear Case: For finite-difference and basis collocation methods applied to nonlinear problems, the Jacobian has the form

$$J_{fdbc}(\boldsymbol{v}) = (D \otimes I_N)\boldsymbol{C} + \boldsymbol{G} \tag{110}$$

where $\boldsymbol{C}$ and $\boldsymbol{G}$ are as defined in (38) and (39). In order to precondition J_{fdbc}, consider computing averages of the individual C_m and G_m matrices in the M diagonal blocks in $\boldsymbol{C}$ and $\boldsymbol{G}$. Using these C_{avg} and G_{avg} matrices in (110) results in a decomposition

$$\begin{aligned} J_{fdbc}(\boldsymbol{v}) = {} & (D \otimes C_{\text{avg}} + I_M \otimes G_{\text{avg}}) \\ & + ((D \otimes I_N)\Delta\boldsymbol{C} + \Delta\boldsymbol{G}) \end{aligned} \tag{111}$$

where $\Delta\boldsymbol{C} = \boldsymbol{C} - (I_M \otimes C_{\text{avg}})$ and $\Delta\boldsymbol{G} = \boldsymbol{G} - (I_M \otimes G_{\text{avg}})$. To see how the terms involving C_{avg} and $\boldsymbol{C}$ in (111) were derived from (110), consider a few intermediate steps. First, note that by the definition of $\Delta\boldsymbol{C}$

$$(D \otimes I_N)\boldsymbol{C} = (D \otimes I_N)\big((I_M \otimes C_{\text{avg}}) + \Delta\boldsymbol{C}\big) \tag{112}$$

which can be reorganized as

$$(D \otimes I_N)\boldsymbol{C} = (D \otimes I_N)(I_M \otimes C_{\text{avg}}) + (D \otimes I_N)\Delta\boldsymbol{C}. \tag{113}$$

The first term on the right-hand side of (113) can be simplified using the reverse of the Kronecker property in (102). That is

$$(D \otimes I_N)(I_M \otimes C_{\text{avg}}) = (DI_M \otimes I_N C_{\text{avg}}) = (D \otimes C_{\text{avg}}) \tag{114}$$

where the last equality follows trivially from the fact that I_M and I_N are identity matrices. The result needed to derive (111) from (110) is then

$$(D \otimes I_N)\boldsymbol{C} = (D \otimes C_{\text{avg}}) + (D \otimes I_N)\Delta\boldsymbol{C}. \tag{115}$$

Though (111) is somewhat cumbersome to derive, its form suggests preconditioning using $(D \otimes C_{\text{avg}} + I_M \otimes G_{\text{avg}})^{-1}$, which, as shown above, is reasonbly inexpensive to apply. The preconditioned Jacobian is then

$$(D \otimes C + I_M \otimes G)^{-1} J_{fdbc}(\boldsymbol{v}) = I + \Delta_{DCG} \tag{116}$$

where

$$\Delta_{DCG} = (D \otimes C + I_M \otimes G)^{-1}((D \otimes I_N)\Delta\boldsymbol{C} + \Delta\boldsymbol{G}). \tag{117}$$

If the circuit is only mildly nonlinear, then Δ_{DCG} will be small, and the preconditioned Jacobian in (117) will close to the identity matrix and have tightly clustered eigenvalues. As mentioned above, for such a case a Krylov-subspace method will converge rapidly.

5) Example Results: In this section, we present some limited experimental results to both demonstrate the reduction in computation time that can be achieved using matrix-implicit iterative methods and to show the effectiveness of the averaging preconditioner.

In Table 2, we compare the megaflops required for different methods to solve the linear system associated with a Fourier basis-collocation scheme. We compare Gaussian elimination (GE), preconditioned explicit GMRES (GMRES), and matrix implicit GMRES (MI). Megaflops rather than CPU time are computed because our implementations are not uniformly optimized. The example used to generate the table is a distortion analysis of a 31-node CMOS operational transconductance amplifier. Note that for a 32 harmonic simulation, with 2232 unknowns, the matrix-implicit GMRES is more than ten

Table 2 Megaflops Required by Different Techniques Single-Frequency Distortation Analysis of a Nonlinear 31-Node Operational Transconductance Amplifier

num harms	GE	GMRES	MI
4	2.62	13.00	9.43
8	20.60	28.87	15.89
16	159.65	110.52	32.19
32	1280.50	561.79	81.93

Table 3 Results From Single-Frequency Distortation Analysis of a Nonlinear 31-Node Operational Transconductance Amplifier. Note the Increase in Number of Iterations With Number of Harmonics Without Preconditioning

num harms	GMRES Iters	Precond Iters
10	139	41
14	155	48
18	159	49
20	173	47
32	200	41

times faster than Gaussian elimination and five times faster than explicit GMRES.

In Table 3, we examine the effectiveness of the averaging preconditioner, again for the example of distortion analysis of an CMOS amplifier. As the table shows, the number of GMRES iterations required to solve the linear system is reduced by a factor of four when preconditioning is used.

C. Remarks

The derivations and analyses lead to three observations about preconditioning.

1) Shooting-Newton methods do not need preconditioners.
2) Averaging preconditioners are effective for nearly linear problems.
3) Lower-triangular preconditioners are effective when D is almost lower triangular.

Unfortunately, the above three observations do not cover an important case. If Fourier basis collocation methods are used on highly nonlinear problems, then D is dense and not nearly lower triangular, and the averaging preconditioners are not always effective. Fourier basis collocation methods are very commonly used, so a number of strategies have been employed which make use of frequency domain interpretations to enhance the averaging preconditioners [5], [6], [11], [20], [21]. Another alternative that has met with limited recent success is to use the lower triangular preconditioner for a nearly lower triangular differentiation matrix as a preconditioner for the Jacobian associated with a dense differention matrix. The idea is that if both matrices accurately represent differentiation, one can precondition the other, an idea that first appeared in the partial differential equation literature [19], [22], [47].

V. CONCLUSION

Although this paper is focused on periodic steady-state analysis, most RF communication circuit problems require multitone, noise, and autonomous oscillator analysis. For example, mixers used for down conversion generate sum and difference frequencies that can be separated by several orders of magnitude. For this reason, the newest work in this area is applying matrix-implicit iterative techniques to accelerating multitone and noise problems using multitone harmonic balance [11], linear time-varying noise analysis [38], [39], frequency-time techniques [6], multitime PDE methods [24], and even wavelets [45]. It is hoped that novice researchers will find our attempt at a comprehensive treatment of the basics of periodic steady-state methods a helpful introduction to this rapidly evolving field. ■

Acknowledgment

The authors would like to thank A. El-Fadel, K. Gullapalli, R. Melville, P. Feldmann, C. S. Ong, D. Ramaswamy, J. Roychowdhury, M. Steer, D. Sharrit, and Y. Thodesen for many valuable discussions over the years. They would also like to thank J. Phillips and M. Tsuk for patiently convincing them of the value of the Kronecker product abstraction and would particularly like to thank Dr. Tsuk for his thorough proofreading of this manuscript.

REFERENCES

[1] L. Nagel and R. Rohrer, "Computer analysis of nonlinear circuits, excluding radiation (CANCER)," *IEEE J. Solid State Circuits*, Aug. 1971.

[2] L. Zadeh and C. Desoer, *Linear System Theory*. New York: McGraw-Hill, 1963.

[3] L. Chua, C. Desoer, and E. Kuh, *Linear and Nonlinear Circuits*. New York: McGraw-Hill, 1987.

[4] A. Ushida, L. Chua, and T. Sugawara, "A substitution algorithm for solving nonlinear circuits with multifrequency components," *Int. J. Circuit Theory Applic.*, vol. 15, 1987.

[5] P. Heikkilä, "Object-Oriented approach to numerical circuit analysis," Ph.D. dissertation, Helsinki Univ. Technology, Helsinki, Jan. 1992.

[6] K. Kundert, J. White, and A. Sangiovanni-Vincentelli, *Steady-State Methods for Simulating Analog and Microwave Circuits*. Boston, MA: Kluwer, 1990.

[7] R. Gilmore and M. Steer, "Nonlinear circuit analysis using the method of harmonic balance—A review of the art. Part I—Introductory concepts," *Int. J. Microwave Millimeter Wave Computer Aided Engineering*, vol. 1, no. 1, 1991.

[8] M. Okumura, H. Tanimoto, T. Itakura, and T. Sugawara, "Numerical noise analysis for nonlinear circuits with a periodic large signal excitation including cyclostationary noise sources," *IEEE Trans. Circuits Systems—I Fundamental Theory Applic.*, vol. 40, no. 9, pp. 581–590, Sep. 1993.

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

[10] N. Rabbat, A. Sangiovanni-Vincentelli, and H. Hsieh, "A multilevel Newton algorithm with macromodeling and latency for the analysis of large-scale non-linear circuits in the time domain," *IEEE Trans. Circuits Syst.*, vol. 9, no. 1979, pp. 733–741.

[11] R. Melville, P. Feldmann, and J. Roychowdhury, "Efficient multi-tone distortion analysis of analog integrated circuits," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1995.

[12] L. T. Watson, *Appl. Math. Comput.*, vol. 5, no. 1979, pp. 297–311.

[13] R. Telichevesky, K. S. Kundert, and J. K. White, "Efficient steady-state analysis based on matrix-free Krylov-subspace methods," in *Proc. Design Automation Conf.*, Santa Clara, CA, Jun. 1995.

[14] H. Keller, *Numerical Solution of Two Point Boundary-Value Problems*. Philadelphia, PA: SIAM, 1976.

[15] T. J. Aprille and T. N. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," *Proc. IEEE*, vol. 60, no. 1, pp. 108–114, Jan. 1972.

[16] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*. New York: Springer-Verlag, 1989.

[17] D. Gottlieb and S. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*. Philadelphia, PA: SIAM, 1977.

[18] J. Stoer and R. Burlisch, *Introduction to Numerical Analysis*. New York: Springer.

[19] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Mechanics*. New York: Springer-Verlag, 1987.

[20] B. Troyanovsky, Z. Yu, L. So, and R. Dutton, "Relaxation-based harmonic balance technique for semiconductor device simulation," in *Proc. Int. Conf. Computer-Aided Design*, Santa Clara, CA, Nov. 1995.

[21] M. M. Gourary, S. G. Rusakov, S. L. Ulyanov, M. M. Zharov, K. Gullapalli, and B. J. Mulvaney, "The enhancing of efficiency of the harmonic balance analysis by adaptation of preconditioner to circuit nonlinearity," in *Asia and South Pacific Design Automation Conf. (ASP-DAC'00)*, p. 537.

[22] F. Veerse, "Efficient iterative time preconditioners for harmonic balance RF circuit simulation," *in Proc. Int. Conf. Computer-Aided Design (ICCAD '03)*, p. 251.

[23] H. G. Brachtendorf, G. Welsch, R. Laur, and A. Bunse-Gerstner, "Numerical steady state analysis of electronic circuits driven by multi-tone signals," *Electrical Eng.*, vol. 79, pp. 103–112, 1996.

[24] J. Roychowdhury, "Analyzing circuits with widely separated time scales using numerical PDE methods," *IEEE Trans. Circuits Systems I: Fundamental Theory Applic.*, vol. 48, no. 5, pp. 578–594, 2001.

[25] S. Skelboe, "Computation of the periodic steady-state response of nonlinear networks by extrapolation methods," *IEEE Trans. Cts. Syst.*, vol. CAS-27, pp. 161–175, 1980.

[26] V. Rizzoli and A. Neri, "State of the art and present trends in nonlinear microwave CAD techniques," *IEEE Trans. Microwave Theory Tech.*, vol. 36, no. 2, pp. 343–365, Feb. 1988.

[27] R. W. Freund, G. H. Golub, and N. M. Nachtigal, "Iterative solution of linear systems," *Acta Numerica*, pp. 57–100, 1991.

[28] C. V. Loan, *Computational Frameworks for the Fast Fourier Transform*. Philadelphia, PA: SIAM, 1992.

[29] L. L. Whitcomb, *Notes on Kronecker Products*. [Online]. Available: spray.me.jhu.edu/llw/courses/me530647/kron_1.pdf

[30] P. Feldmann, R. C. Melville, and D. Long, "Efficient frequency domain analysis of large nonlinear analog circuits," in *Proc. IEEE CICC*, May 1996.

[31] K. Kundert, J. White, and A. Sangiovanni-Vincentelli, "A mixed frequency-time approach for distortion analysis of switching filter circuits," *IEEE J. Solid-State Cts.*, vol. 24, no. 2, pp. 443–451, Apr. 1989.

[32] C. W. Gear, *Numerical Initial Value Problem in Ordinary Differential Equations*. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[33] K. S. Kundert, *The Designer's Guide to SPICE and Spectre*. Boston, MA: Kluwer, 1995.

[34] D. Feng, J. Phillips, K. Nabors, K. Kundert, and J. White, "Efficient computation of quasi-periodic circuit operating conditions via a mixed frequency/time approach," in *Proc. 36th Design Automation Conf.*, Jun. 1999.

[35] J. Chen, D. Feng, J. Phillips, and K. Kundert, "Simulation and modeling of intermodulation distortion in communication circuits," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1999.

[36] K. Kundert, "Introduction to RF simulation and its application," *J. Solid-State Circuits*, vol. 34, no. 9, Sep. 1999.

[37] Y. Thodeson and K. Kundert, "Parametric harmonic balance," in *Proc. IEEE MTT-S Int. Microwave Symp. Dig.*, Jun. 1996.

[38] R. Telichevesky, K. S. Kundert, and J. K. White, "Efficient AC and noise analysis of two-tone RF circuits," in *Proc. 33rd Design Automation Conf.*, Jun. 1996.

[39] J. Roychowdhury, D. Long, and P. Feldmann, "Cyclostationary noise analysis of large RF circuits with multi-tone excitations," *IEEE J. Solid-State Circuits*, Mar. 1998.

[40] R. Telichevesky, K. Kundert, I. El-Fadel, and J. White, "Fast simulation algorithms for RF circuits," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1996.

[41] J. Phillips and B. Yang, "A multi-interval Chebyshev collocation method for efficient high-accuracy RF circuit simulation," in *Proc. 37th ACM/IEEE Design Automation Conf. (DAC 2000)*, Los Angeles, CA, Jun. 2000, pp. 178–183.

[42] K. Mayaram, D. C. Lee, S. Moinian, D. Rich, and J. Roychowdhury, "Computer-aided circuit analysis tools for RFIC simulation: Algorithms, features, and limitations," *IEEE Trans. CAS II*, pp. 274–286, Apr. 2000.

[43] L. N. Trefethen, *Spectral Methods in Matlab*. Philadelphia, PA: SIAM, Jun. 2000.

[44] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*. Philadelphia, PA: SIAM, 1997.

[45] N. Soveiko and M. Nakhla, "Wavelet harmonic balance," *IEEE Microwave Wireless Components Lett.*, Jul. 2003.

[46] O. J. Nastov and J. White, "Time-mapped harmonic balance," in *Proc. IEEE Design Automation Conf.*, pp. 641–646.

[47] O. J. Nastov, "Spectral methods for circuit analysis," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, 1999.

ABOUT THE AUTHORS

Ognen Nastov, photograph and biography not available at the time of publication.

Rircardo Telichevesky (Member, IEEE) received the B.S. degree in electrical engineering from the Universidade Federal do Rio Grande do Sul, Brazil, in 1985, the M.S. degree in electrical engineering from the Technion, Israel, in 1988, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1994.

He specialized in computer architectures and numerical simulation. He is the Principal Researcher at Kineret Design Automation, Santa Clara, CA, working in the simulation of analog and mixed signal systems. He played a key role in the development of Cadence's SpectreRF and Spectre simulator. He also worked in the modeling and real-time simulation of physics-based mechanical systems for virtual environments at Xulu Entertainment, Inc. His current research interests include numerical algorithms and efficient data structures for analog, RF, mixed signal, and system level simulation, and its interoperability with other tools in the design flow.

Ken Kundert received the B.S. degree in 1979, the M.S. degree in 1983, and the Ph.D. degree in electrical engineering from the University of California, Berkeley, in 1989.

He is renowned for creating two of the most innovative, influential, and highest grossing circuit simulators ever produced: Cadence's Spectre and Agilent's harmonic balance simulator. He has a deep understanding of analog/mixed-signal and RF simulation technology and is well versed in its application. He Co-founded Designer's Guide Consulting, Inc., in 2005. From 1989 to 2005, he worked at Cadence Design Systems as a Fellow. He created Spectre and was the Principal Architect of the Spectre circuit simulation family. As such, he has led the development of Spectre, SpectreHDL, and SpectreRF. He also played a key role in the development of Cadence's AMS Designer and made substantial contributions to both the Verilog-AMS and VHDL-AMS languages. Before that , he was a Circuit Designer at Tektronix and Hewlett-Packard and contributed to the design of the HP 8510 microwave network analyzer. He has written three books on circuit simulation: *The Designer's Guide to Verilog-AMS* in 2004, *The Designer's Guide to SPICE and Spectre* in 1995, and *Steady-State Methods for Simulating Analog and Microwave Circuits* in 1990. He is the author of 11 patents.

Jacob White (Member, IEEE) received the B.S. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, and the S.M. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley.

He worked at the IBM T.J. Watson Research Center, from 1985 to 1987, and was the Analog Devices Career Development Assistant Professor at the Massachusetts Institute of Technology from 1987 to 1989. He is currently at the Massachusetts Institute of Technology where he is the C. H. Green Professor in Electrical Engineering and Computer Science. His research interests include numerical algorithms for problems in the design and robust optimization of circuits, interconnect, nanophotonics, carbon nanotubes, micromachining for biological applications, biomolecules, and network models of biological systems.

Dr. White was a 1988 Presidential Young Investigator and was an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN from 1992 until 1996. He was Chair of the International Conference on Computer-Aided Design in 1999.