Fast Simulation Algorithms for RF Circuits

R. Telichevesky, K. Kundert
Cadence Design Systems
San Jose, California

I. Elfadel, J. White
Massachusetts Institute of Technology
Cambridge, Massachusetts

Abstract

RF integrated circuit designers make extensive use of simulation tools which perform nonlinear periodic steady-state analysis and its extensions. However, the computational costs of these simulation tools have restricted users from examining the detailed behavior of complete RF subsystems. Recent algorithmic developments, based on matrix-implicit iterative methods, is rapidly changing this situation and providing new faster tools which can easily analyze circuits with hundreds of devices. In this paper we present these new methods by describing how they can be used to accelerate finite-difference, shooting-Newton, and harmonic-balance based algorithms for periodic steady-state analysis.

1 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 narrow-band amplifiers. Although methods like multitone harmonic balance, linear time-varying, and mixed frequency-time techniques [6, 3, 1, 4] can perform these analyses, the traditional implementations of these techniques grow so rapidly with increasing circuit size that they have been too computationally expensive to use for more complex circuits.

*This work was supported by a grants from Motorola and Cadence Design Systems

Recent algorithmic developments, based on preconditioned matrix-implicit Krylov-subspace iterative methods, is rapidly changing the situation and providing new tools which can easily analyze circuits with hundreds of devices. Preconditioned iterative techniques have been applied to accelerating periodic steady-state analysis using both harmonic balance methods [2, 5] and time-domain shooting methods [9]. Additional results for more general analyses are also under investigation.

In this tutorial paper, we try to present these accelerated methods for periodic steady-state analysis in a cohesive fashion, and perhaps make the more specialized papers that have or will shortly appear a little more approachable. We start in the next section by presenting the time domain finite-difference-Newton and shooting-Newton methods, give a brief idea of their computational costs using standard factorization, and give the faster matrix-implicit algorithm. In section three we present, using as much common notation as possible, the same aspects of the harmonic-balance approach. Where possible, we give some computational results so as to give the reader a feel for the cost reductions. Also, we have tried to pay particular attention to providing approachable references in the mathematical literature. Finally, conclusions and acknowledgements are given in Section 4.

2 Time Domain Methods

Finding the periodic steady-state solution of a circuit means 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, the steady-state solution must be a particular solution to the circuit equations, as in

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

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), respectively. In addition to satisfying (1), the periodic steady solution must also satisfy the two-point constraint

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

A sampling discretization is usually introduced to numerically solve the combination of (1) and (2), by which

we mean that $v(t)$ is represented over the period T by a sequence of M numerically computed discrete points,

$$\begin{bmatrix} v(t_1) \\ v(t_2) \\ \vdots \\ v(t_M) \end{bmatrix} \approx \begin{bmatrix} \hat{v}(t_1) \\ \hat{v}(t_2) \\ \vdots \\ \hat{v}(t_M) \end{bmatrix} \equiv \hat{\boldsymbol{v}}, \tag{3}$$

where $t_M = T$, the hat is used to denote numerical approximation, and $\hat{\boldsymbol{v}} \in \Re^{MN}$ is introduced for notational convenience.

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

$$F_1(\hat{\boldsymbol{v}}, \hat{v}_(t_0)) \equiv \frac{q(\hat{v}(t_1)) - q(\hat{v}(t_0))}{h_1} + i(v(t_1)) + u(t_1) = 0 \tag{4}$$

and for $j \in \{2, ..., M\}$

$$F_j(\hat{\boldsymbol{v}}) \equiv \frac{q(\hat{v}(t_j)) - q(\hat{v}(t_{j-1}))}{h_j} + i(v(t_j)) + u(t_j) = 0, \tag{5}$$

where $h_j = t_j - t_{j-1}$ and $\hat{v}(t_0)$ is the unknown differential equation initial condition which generates the periodic steady-state. Combining (2) with (5) yields a system of nonlinear equations for $\hat{\boldsymbol{v}}$ and $\hat{v}(t_0)$.

Two variants of Newton's method are commonly used to solve the combination of (2), (4) and (5): the finite-difference-Newton and the shooting-Newton methods [10, 11]. The finite-difference method is a direct Newton method but the shooting-Newton method uses an additional inner Newton iteration. The use of an inner iteration implies that the shooting-Newton method is a multi-level Newton method, and therefore has typically better global convergence properties for stable problems. Despite this convergence property difference, computing the Newton iterates in these two methods leads to surprisingly similar systems of linear equations, and both these linear systems are easily solved using iterative methods like preconditioned GMRES [7]. Below we will present the two methods, and then describe how to apply the iterative matrix-implicit solution techniques.

2.1 Finite-Difference-Newton Method

The finite-difference-Newton method is precisely Newton's method applied directly to (5), where (2) is included by the identification $\hat{v}(t_0) = \hat{v}(t_M)$. Before giving the finite-difference-Newton iteration equation, we establish

some notation. Let L be the block lower bidiagonal matrix given by

$$L \equiv \begin{bmatrix} \frac{C_1}{h_1} + G_1 & & & \\ -\frac{C_1}{h_2} & \frac{C_2}{h_2} + G_2 & & \\ & \ddots & \ddots & \\ & & -\frac{C_{M-1}}{h_M} & \frac{C_M}{h_M} + G_M \end{bmatrix}, \tag{6}$$

and define B as

$$B \equiv \begin{bmatrix} 0 & \dots & 0 & -\frac{C_M}{h_1} \\ & 0 & & \\ & \ddots & \ddots & \\ & & 0 & 0 \end{bmatrix}, \tag{7}$$

where $C_j, G_j \in \Re^{N \times N}$ denote $dq(\hat{v}(t_j))/dv$ and $di(\hat{v}(t_j))/dv$, respectively. For most practical circuits, the connectivity is very sparse. This implies that for typical problems, there are fewer than ten entries per row in C_i and G_i, regardless of the circuit size.

Using the above notation, the finite-difference-Newton iteration equation is given by

$$(L + B)\left(\hat{\boldsymbol{v}}^{k+1} - \hat{\boldsymbol{v}}^{k}\right) = -\boldsymbol{F}_{fd}(\hat{\boldsymbol{v}}^{k}), \tag{8}$$

where k is the Newton iteration index and

$$-\boldsymbol{F}_{fd}(\hat{\boldsymbol{v}}^{k}) \equiv \begin{bmatrix} -F_1(\hat{\boldsymbol{v}}^{k}, \hat{v}^{k}(t_M)) \\ -F_2(\hat{\boldsymbol{v}}^{k}) \\ \vdots \\ -F_M(\hat{\boldsymbol{v}}^{k}) \end{bmatrix}. \tag{9}$$

2.2 Shooting-Newton Method

Another approach to solving (5) is to exploit the fact that if $\hat{v}(t_0)$ is given, then the nonlinear equation

$$\frac{q(\hat{v}(t_1)) - q(\hat{v}(t_0))}{h_1} + i(\hat{v}(t_1)) + u(t_1) = 0 \tag{10}$$

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

$$\frac{q(\hat{v}(t_2)) - q(\hat{v}(t_1))}{h_2} + i(\hat{v}(t_2)) + u(t_2) = 0 \tag{11}$$

for $\hat{v}(t_2)$. This procedure can be continued, effectively integrating the differential equation one timestep at a time in the standard fashion, using the initial condition $\hat{v}(t_0)$. And since the nonlinear equations are solved at each timestep, $\hat{v}(t_j)$ is an implicitly defined function of $\hat{v}(t_0)$. This function, computed by the numerical integration, is referred to as the state-transition function of the discretized system and is denoted as

$$\phi(\hat{v}(t_0), t_0, t_j). \tag{12}$$

Note that if $\hat{v}(t_j)$ is computed by solving the nonlinear equation F_j in (4) or (5) at each timestep, or equivalently if $\hat{v}(t_j)$ is computed by evaluating $\phi(\hat{v}(t_0), t_0, t_j)$, then (4) and (5) will always be satisfied, regardless of the choice of $\hat{v}(t_0)$. If $\hat{v}(t_0)$ equals $\hat{v}(t_M)$, then (2) is also satisfied and a periodic steady-state solution has been computed. This statement is mathematically equivalent to the condition

$$F_{sh}(\hat{v}(t_0)) \equiv \phi(\hat{v}(t_0), t_0, t_M) - \hat{v}(t_0) = 0. \tag{13}$$

The shooting-Newton method then computes $\hat{v}(t_0)$ for the periodic steady-state by applying Newton's method to solving (13), resulting in the iteration equation

$$[J_\phi(\hat{v}^k(t_0), 0, T) - I][\hat{v}^{k+1}(t_0) - \hat{v}^k(t_0)] = -F_{sh}(\hat{v}^k(t_0)), \tag{14}$$

where k is the Newton iteration index, I is the identity matrix, and

$$J_\phi(v, 0, T) \equiv \frac{d}{dv}\phi(v, 0, T) \tag{15}$$

is referred to as the sensitivity matrix.

To complete the description of the shooting-Newton method, and to see why we refer to it as a multilevel-Newton method, it is necessary to present the procedure for computing $\phi(v, 0, T)$ and $J_\phi(v, 0, T)$. As mentioned above, computing the state transition function is equivalent to solving the backward-Euler equations in (4) and (5) one timestep at a time. Solving the backward-Euler equations is usually accomplished using an inner Newton iteration, as in

$$\begin{aligned} \left[\frac{C_{jkl}}{h_j} + G_{jkl}\right](\hat{v}^{k,(l+1)}(t_j) - \hat{v}^{k,l}(t_j)) = \\ -\frac{1}{h_j}\left(q(\hat{v}^{k,l}(t_j)) - q(\hat{v}^{k,l}(t_j))\right) - i(\hat{v}^{k,l}(t_j)) - u(t_j) \end{aligned} \tag{16}$$

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

To see how to compute $J_\phi(v, 0, T)$ as a by-product of the Newton iteration in (16), let $l = *$ denote the inner Newton iteration which achieves sufficent convergence, and let $\hat{v}^{k,*}(t_j)$ denote the associated converged solution. Using this notation,

$$\frac{q(\hat{v}^{k,*}(t_j)) - q(\hat{v}^{k,*}(t_{j-1}))}{h_j} + i(\hat{v}^{k,*}(t_j)) + u(t_j) = 0 \tag{17}$$

to within the Newton iteration convergence tolerence. Implicitly differentiating (17) with respect to $\hat{v}^k(t_0)$ results in

$$\left[\frac{C_{jk*}}{h_j} + G_{jk*}\right] \frac{d\hat{v}^{k,(*+1)}(t_j)}{d\hat{v}^k(t_0)} = \frac{C_{(j-1)k*}}{h_j} \frac{d\hat{v}^{k,*}(t_{j-1})}{d\hat{v}^k(t_0)} \tag{18}$$

By recursively applying the above equation, one can derive that

$$J_\phi(\hat{v}^k(t_0), t_0, t_M) = \prod_{j=1}^{M} \left[\frac{C_{jk*}}{h_j} + G_{jk*}\right]^{-1} \frac{C_{(j-1)k*}}{h_j}, \tag{19}$$

where the notation $\prod_{j=1}^{M}$ indicates a product rather than a sum.

2.3 Direct Matrix Solution

In comparing the shooting-Newton iteration, (14), to the finite-difference-Newton iteration, (8), 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 (19) is more expensive than computing the finite-difference Jacobian. In this section we will show that using direct factorization to solve either (14) or (8) leads to nearly equivalent computations.

To start, consider that L defined in (6) is block lower bidiagonal, where the diagonal blocks have the same structure as the single timestep Jacobian in (16). 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

$$(I_{NM} + L^{-1}B) \left(\tilde{\boldsymbol{v}}^{k+1} - \tilde{\boldsymbol{v}}^{k}\right) = -L^{-1}\boldsymbol{F}(\tilde{\boldsymbol{v}}^{k}), \tag{20}$$

though L^{-1} would never be explicitly computed. Here, we have denoted the identity matrix in $\Re^{NM \times NM}$ by I_{NM}.

Examining (20) reveals an important feature, that $L^{-1}B \in \Re^{NM \times NM}$ has nonzero entries *only in the last N columns*. Specifically,

$$(I_{NM} + L^{-1}B) = \begin{bmatrix} I_N & \dots & 0 & P_1 \\ & I_N & & \\ & \ddots & \ddots & \\ & & 0 & I_N + P_M \end{bmatrix} \tag{21}$$

where $P_i \in \Re^{N \times N}$ is the $((i-1)*N)+1$ through $i*N$ rows of the last N columns of $L^{-1}B$. This bordered-block diagonal form implies that $\tilde{v}^{k+1} - \tilde{v}^k$ in (20) can be computed in three steps. The first step is to compute P_M. This can be accomplished by forming the N products Be_i, where the $e_i's$ are the first N unit vectors in $\Re^{NM}$, and then backsolving N times with L. The second step is to use the computed P_M to determine the last N entries in $\tilde{v}^{k+1} - \tilde{v}^k$. This second task can be performed by directly factoring $I + P_M$, and using it to solve solving the last N equations in (20). The last step in solving (20) is to compute the rest of $\tilde{v}^{k+1} - \tilde{v}^k$ by backsolving again with L.

The close relation between solving (20) and (14) can now be easily established. If L and B are formed using C_{jk*} and G_{jk*} as defined in (18), then by explicitly computing $L^{-1}B$ it can be shown that $J_\phi(\hat{v}^k(t_0), t_0, t_M) = P_M$. The importance of this observation is that solving the shooting-Newton update equation is nearly computationally equivalent to solving the finite-difference-Newton update equation. Specifically, (14) can be solved by following the first two steps of the above procedure, except the factorization of $P_M - I_N$ replaces the factorization of $I_N + P_M$.

The strong connection between the computational steps in solving the shooting-Newton and finite-difference-Newton update equations implies their computational costs are very similar. For both methods, the dominant costs are the more than order MN^2 operations required to form P_M and the N^3 operations required to factor the dense $I_N + P_M$ or $P_M - I_N$ matrix. The reason forming P_M is so expensive is that it requires N backsolves with L, and each backsolve with L requires *at least* order NM operations. We say *at least* because the cost of backsolving with L depends critically on the amount of fill-in produced by factoring the $N \times N$ sparse diagonal

blocks $\frac{C_j}{h_j} + G_j$. Since these blocks are generated by the linearization of a circuit, their structure is problem dependent. For example, if the circuit happens to be a tree structure, then $\frac{C_j}{h_j} + G_j$ will have order N elements, the factorization will produce no fill, and backsolving with a factored L will require only order NM operations. However, if the circuit is a three-dimensional mesh, then factoring $\frac{C_j}{h_j} + G_j$ will produce substantial fill-in, and backsolving with a factored L will cost order $N^{4/3}M$ operations.

2.4 Matrix-Implicit Iterative Methods

As described above, applying direct factorization to solving (14) or (20) results in an algorithm whose computational complexity grows faster than order $MN^2 + N^3$. This rapid growth of computation time with problem size severely limits the size of circuit which can be analyzed, and in this section we examine how to reduce matrix solution cost using matrix-implicit iterative methods. To begin, consider solving the linear system in (8) using an iterative method like the Krylov-subspace based GMRES algorithm [7]. A simplified version of GMRES is given below.

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 $k = k + 1$.

}

When the GMRES algorithm is applied to solving (8), if the number of iterations required to achieve convergence is bounded by a constant independent of problem size, then the resulting solution algorithm is much faster than direct factorization. The cost is reduced from order $MN^2 + N^3$ to order NM operations. To see this, consider that forming $(L + B)p^k$ costs order NM operations because of the circuit matrix sparsity, and we have assumed a constant number of iterations. Unfortunately, for typical circuit problems, $(L+B)$ in (8) is much too ill-conditioned for GMRES to converge rapidly. Instead, the iterative method should be preconditioned, or premultiplied, with L^{-1}, and this preconditioner insures rapid convergence [8].

Preconditioning using L^{-1} corresponds to applying the GMRES algorithm to solving (20) instead of (8). It is crucial to observe that $I_{NM} + L^{-1}B$ is not explicitly required in the GMRES algorithm, it is only necessary to be able to compute the matrix-vector products $(I_{NM} + L^{-1}B)p^k$. This observation implies that each preconditioned GMRES iteration can be performed in nearly order NM operations by multiplying p^k by B, then backsolving with a factored L, and finally adding p^k. Therefore, the cost per iteration is somewhat more than order NM operations. Also, since the L^{-1} preconditioning has insured rapid convergence independent of problem size, the entire solution algorithm is nearly order NM operations. Finally, an almost identical algorithm can be used to solve (14). Consider that computing the matrix-vector product $(J_\phi - I_N)\,p^k$, formed when GMRES is applied to solving (14), is equivalent to padding the N-length p^k with zeros to make it an NM-length vector, and then computing $(L^{-1}B - I_{NM})p^k$. If such an approach is used, $P_M - I_N$ is represented "quite" implicitly and the so derived shooting-Newton-GMRES method has the same nearly order NM operation cost as finite-difference methods [9]. Be reminded, however, that shooting-Newton methods still have superior global convergence properties for stable problems.

2.5 Results

In this section we experimentally examine the performance of three shooting-Newton schemes: direct factorization or Gaussian elimination, forming GMRES, and matrix-implicit GMRES.

Table 1 compares the performance of the various Newton-Raphson shooting method approaches implemented

experimentally into the Spectre circuit simulator. 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, explicit GMRES, and the matrix-implicit form. All the results were obtained on a HP712/80 workstation. The sixth column demonstrates the effectiveness of the matrix-implicit approach, listing the speedup obtained in respect to the Gaussian-elimination method. Note that the speed-up over the explicit GMRES algorithm would be similar for the size examples examined.

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

Table 1: Comparison of different shooting method schemes

3 Frequency-Domain Methods

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

$$x(t) = \sum_{k=-\infty}^{k=\infty} X[k] e^{j 2\pi f_k t} \tag{22}$$

where $f_k = \frac{kt}{T}$ and

$$X[k] = \frac{1}{T} \int_{-T/2}^{T/2} x(t) e^{-j 2\pi f_k t} dt. \tag{23}$$

If in addition to being periodic, $x(t)$ is sufficiently smooth, formally infinitely continuously differentiable, then the $X[k]$'s vanish 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,

$$\hat{x}(t) = \sum_{k=-K}^{k=K} \hat{X}[k] e^{j 2\pi f_k t} \tag{24}$$

where the number of harmonics, K, is typically fewer than 15. Note that the time derivative of $\hat{x}(t)$ is given by

$$\frac{d}{dt}\hat{x}(t) = \sum_{k=-K}^{k=K} \hat{X}[k] j 2\pi f_k e^{j 2\pi f_k t}. \tag{25}$$

Representing a periodic waveform as a truncated Fourier series can be used to approximately solve (1) and (2), in which case the resulting methods are referred to as harmonic balance methods [3], or Fourier spectral methods [13]. Of particularly wide use in circuit simulation are the spectral-collocation-on-charge variants of these methods [3]. To derive the collocation method requires a little notation. If $q(v)$ in (1) is globally invertible, then we can denote $\tilde{q}(t) \equiv q(v(t))$ and $\tilde{i}(\tilde{q}(t)) \equiv i(v(t))$. Using this notation, (1) can be rewritten as a differential equation in normal form in terms of q,

$$\frac{d}{dt}\tilde{q}(t) + \tilde{i}(\tilde{q}(t)) + u(t) = 0. \tag{26}$$

As an aside, standard integration methods are typically applied to the above form of (1) to insure charge conservation.

Introducing the truncated Fourier series approximation for $\tilde{q}(t)$,

$$\hat{\tilde{q}}(t) = \sum_{k=-K}^{k=K} \hat{Q}[k] e^{j 2\pi f_k t}, \tag{27}$$

into (26) results in

$$\sum_{k=-K}^{k=K} \hat{Q}[k] j 2\pi f_k e^{j 2\pi f_k t} + \tilde{i}\left(\sum_{k=-K}^{k=K} \hat{Q}[k] e^{j 2\pi f_k t} \right) + u(t) = 0. \tag{28}$$

Of course, it would be best if (28) holds for all t, but there are only $M = 2K + 1$ degrees of freedom, the $\hat{Q}[k]'s$. Note, each $\hat{Q}[k]$ is an $N-$length vector, so we are using "degrees of freedom" in a very loose sense. In order to derive M systems of equations from which to determine the $\hat{Q}[k]$'s, (28) is enforced only at selected collocation points $t_1, t_2, ..., t_M = T$, as in

$$\sum_{k=-K}^{k=K} \hat{Q}[k] j 2\pi f_k e^{j 2\pi f_k t_j} + \tilde{i}\left(\sum_{k=-K}^{k=K} \hat{Q}[k] e^{j 2\pi f_k t_j} \right) + u(t_j) = 0. \tag{29}$$

Convergence analysis of spectral-collocation methods can be found in [13, 14].

Note that (29) represents NM equations in NM unknowns, but its form is not so readily identified with the original functions $q(v)$ and $i(v)$. The relation is easily unraveled, however. Let $\Gamma^{-1} \in C^{NM \times NM}$ denote the discrete Fourier transform matrix which represents the relationship between the $\hat{Q}[k]'s$ and $\hat{\tilde{q}}(t_j)'s$, as in

$$\Gamma^{-1} \begin{bmatrix} \hat{Q}[K]) \\ \hat{Q}[(K-1)] \\ \vdots \\ \hat{Q}[-K] \end{bmatrix} = \begin{bmatrix} \hat{\tilde{q}}(t_1) \\ \hat{\tilde{q}}(t_2) \\ \vdots \\ \hat{\tilde{q}}(t_M) \end{bmatrix}. \tag{30}$$

Using the definitions of $\tilde{q}$ and $\tilde{i}$, (29) can be simplified as

$$\boldsymbol{F}_{time}(\hat{\boldsymbol{v}}) = \Gamma^{-1}\Omega\Gamma\boldsymbol{q}(\hat{\boldsymbol{v}}) + \boldsymbol{i}(\hat{\boldsymbol{v}}) + \boldsymbol{u} = 0 \tag{31}$$

where

$$\boldsymbol{q}(\hat{\boldsymbol{v}}) \equiv \begin{bmatrix} q(\hat{v}(t_1)) \\ q(\hat{v}(t_2)) \\ \vdots \\ q(\hat{v}(t_M)) \end{bmatrix}, \quad \boldsymbol{i}(\hat{\boldsymbol{v}}) \equiv \begin{bmatrix} i(\hat{v}(t_1)) \\ i(\hat{v}(t_2)) \\ \vdots \\ i(\hat{v}(t_M)) \end{bmatrix}, \tag{32}$$

$$\boldsymbol{u} \equiv \begin{bmatrix} u(t_1) \\ u(t_2) \\ \vdots \\ u(t_M)) \end{bmatrix}, \tag{33}$$

and Ω is the diagonal matrix given by

$$\Omega \equiv \begin{bmatrix} j2\pi f_K I_N & & & \\ & j2\pi f_{K-1} I_N & & \\ & & \ddots & \ddots \\ & & & j2\pi f_{-K} I_N \end{bmatrix}. \tag{34}$$

Using a change of variables, $\hat{\boldsymbol{V}} = \Gamma\hat{\boldsymbol{v}}$, (31) can be written in the form

$$\boldsymbol{F}_{freq}(\hat{\boldsymbol{V}}) = \Omega\Gamma\boldsymbol{q}(\Gamma^{-1}\hat{\boldsymbol{V}}) + \Gamma\boldsymbol{i}(\Gamma^{-1}\hat{\boldsymbol{V}}) + \Gamma\boldsymbol{u} = 0. \tag{35}$$

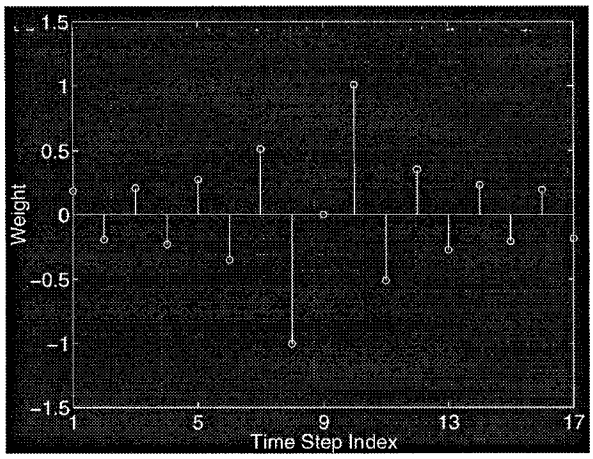

Figure 1: The harmonic balance discretization weights for t_9 where $T = 17$ and $M = 17$.

The form in (35) is often referred to as the harmonic balance form because the unknowns are the Fourier coefficients and the equations are in the frequency domain.

From the formulation in (31), it is clear that $\Gamma^{-1}\Omega\Gamma$ is an approximate differentiation operator. The weights for this spectral differentiation operator for the case of $T = 17$, $M = 17$, and at timepoint $t_9 = 9$ are plotted in Figure (1). 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

$$\frac{d}{dt}x(t_9) \approx \frac{x(t_{10}) - x(t_8)}{t_{10} - t_8} \tag{36}$$

The connection between spectral differentiation and standard differencing schemes can be exploited when developing preconditioners, a point we will return to subsequently.

3.1 Applying Newton's Method

Newton's method applied to (31) or (35) yield the iteration equations

$$\left(\Gamma^{-1}\Omega\Gamma\boldsymbol{C} + \boldsymbol{G}\right)\left(\hat{\boldsymbol{v}}^{k+1} - \hat{\boldsymbol{v}}^{k}\right) = -\boldsymbol{F}_{time}(\hat{\boldsymbol{v}}) \tag{37}$$

or

$$\left(\Omega\Gamma\boldsymbol{C}\Gamma^{-1} + \Gamma\boldsymbol{G}\Gamma^{-1}\right)\left(\hat{\boldsymbol{V}}^{k+1} - \hat{\boldsymbol{V}}^{k}\right) = -\boldsymbol{F}_{freq}(\hat{\boldsymbol{v}}) \tag{38}$$

where k is the Newton iteration index, $\boldsymbol{F}_{time}$ and $\boldsymbol{F}_{freq}$ are given in (31) and (35), and $C, \mathrm{G} \in \Re^{\mathrm{NM}\times\mathrm{NM}}$ are block diagonal matrices given by

$$\boldsymbol{C} \equiv \begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_M \end{bmatrix}, \tag{39}$$

$$\boldsymbol{G} \equiv \begin{bmatrix} G_1 & & & \\ & G_2 & & \\ & & \ddots & \\ & & & G_M \end{bmatrix}, \tag{40}$$

where C_j denotes $dq(v(t_j))/dv$ and G_j denotes $di(v(t_j))/dv$.

The Jacobian in (37) has a structure that is similar, but much denser, than the Jacobian associated with the backward-Euler based finite-difference-Newton Method. This is clear from the explicit representation of $(\Gamma^{-1}\Omega\Gamma\boldsymbol{C} + \boldsymbol{G})$,

$$\begin{bmatrix} G_1 & \alpha_1 C_1 & \alpha_2 C_1 & ,\ldots, & \alpha_{-1} C_1 \\ \alpha_{-1} C_2 & G_2 & \alpha_1 C_2 & ,\ldots, & \alpha_{-2} C_2 \\ & \ddots & \ddots & \ddots & \\ \alpha_1 C_M & \alpha_2 C_M & ,\ldots, & \alpha_{-1} C_M & G_M \end{bmatrix} \tag{41}$$

where the α_i's are the weights for the spectral differentiation. Note that the block diagonal terms depend only on the G's (incremental conductances). Also, for linear time-invariant problems, where $C_j = C$ and $G_j = G$ for all j, the matrix density is basically unchanged.

The explicit representation for $\Omega\Gamma\boldsymbol{C}\Gamma^{-1} + \Gamma\boldsymbol{G}\Gamma^{-1}$ is, in general, similarly dense. However, one of the advantages of the harmonic balance form of the equations is that if the problem is linear and time-invariant the matrix is

block diagonal and is given by

$$\begin{bmatrix} \frac{j2\pi K}{T}C+G & & & \\ & \frac{j2\pi(K-1)}{T}C+G & & \\ & & \ddots & \ddots \\ & & & \frac{j2\pi(-K)}{T}C+G \end{bmatrix} \tag{42}$$

where $C = C_j$ and $G = G_j$ for all j. In this case, of course, the matrix just represents phasor analysis at a collection of isolated frequencies.

3.2 Iterative Matrix Solution

As mentioned in Section 2.4, if GMRES is applied to solving a linear system of equations, then the matrix representing the system is not needed explicitly. It is only necessary to be able to compute matrix-vector products. This implies that if GMRES is applied to solving (37) or (38), then the iterations can be performed in order $NMlogM$ operations by employing the fast Fourier transform [12]. This is easily shown, consider computing

$$\left(\Gamma^{-1}\Omega\Gamma\boldsymbol{C}+\boldsymbol{G}\right)p^k \;=\; \Gamma^{-1}\Omega\Gamma\boldsymbol{C}p^k+\boldsymbol{G}p^k. \tag{43}$$

Forming $\boldsymbol{G}p^k$ and $y=\boldsymbol{C}p^k$ require only order NM operations, due to the sparsity of C_i and G_i. Forming $w=\Gamma y$ can be accomplished in $NMlogM$ operations using N fast Fourier Transforms. Multiplying w by Ω is order NM operations because Ω is diagonal. And finally, forming $\Gamma^{-1}\Omega w$ requires order $NMlogM$ operations using N inverse fast Fourier transforms. A similar argument demonstrates that

$$\left(\Omega\Gamma\boldsymbol{C}\Gamma^{-1}+\Gamma\boldsymbol{G}\Gamma^{-1}\right)p^k \tag{44}$$

can be computed in order $MNlogM$ operations.

GMRES applied directly to solving (37) or (38) converges too slowly to be practical. And, since the matrices in (37) or (38) are related by an orthonormal similarity transform, GMRES will converge identically slowly in either case. However, when one considers preconditioners, there is a reason for choosing between the formulations. For the harmonic-balance form, a block-diagonal, sometimes called a block Gauss-Jacobi, preconditioner can be extremely effective when the problem is nearly linear and time-invariant. This is because the off-diagonal blocks

of the matrix in (38) are produced by frequency translations which are minimal for the nearly linear and time-invariant case. Variations on this idea include adding in some of the off-diagonal blocks, using bands of blocks and incomplete factorization, or using the block lower triangular matrix [2, 5, 3, 15].

For rapidly time-varying or nonlinear problems, frequency coupling is more severe, and preconditioners based on discarding off-diagonal blocks of the matrix in (38) become ineffective. A different approach, which involves preconditioning (37), uses matrices associated with finite-difference methods as preconditioners, like the L matrix defined in (6) [14]. The key concept is that both matrices have an approximate differentiation operator, but, for example, the backward-Euler or a central-difference discretization use much sparser operators than spectral discretization. It is worth noting that backward-difference methods have a computational advantage over central-differencing schemes in that they are more lower triangular. Finally, the L matrix given in (6) is not a very effective preconditioner because the backward-Euler operator is not sufficiently accurate to properly precondition the very high order spectral differentiation operator. Matrices associated with higher order differencing schemes are much more effective.

4 Future Work and Acknowledgements

Although in this tutorial paper we have focussed on periodic steady-state analysis, most RF communication circuit problems require multitone analysis. For example, mixers used for down conversion generate sum and difference frequencies that can be seperated by several orders of magnitude. For this reason, the newest work in this area is applying matrix-implicit iterative techniques to accelerating multitone problems using multitone harmonic balance [5], linear time-varying noise analysis [8], and mixed frequency-time techniques [3]. The authors would like to thank Robert Melville, Peter Feldmann, Jaijeet Roychowdhury, Michael Steer, and David Sharrit for many valuable discussions over the years.

References

[1] A. Ushida, L. Chua, and T. Sugawara, "A Substitution Algorithm for solving nonlinear circuits with multi-frequency components. *International Journal on Circuit Theory and Applications*, vol. 15, 1987.

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

[3] K. Kundert, J. White and A. Sangiovanni-Vincentelli. *Steady-State Methods for Simulating Analog And Microwave Circuits.* Kluwer Academic Publishers, Boston 1990.

[4] 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. on Microwave and Millimeter Wave Computer Aided Engineering,* Vol. 1, No. 1, 1991.

[5] R. Melville, P. Feldmann, and J. Roychowdhury. "Efficient multi-tone distortion analysis of analog integrated circuits." *Proceedings of the 1995 IEEE Custom Integrated Circuits Conference,* May 1995.

[6] 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 Transactions On Circuits and Systems - I Fundamental Theory and Applications.*, vol. 40, no. 9, pp. 581–590, September 1993.

[7] 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.

[8] R. Telichevesky, K. Kundert, and J. White. "Matrix-Implicit Iterative Techniques For RF Circuit Analysis," *in preparation.*

[9] R. Telichevesky, Kenneth S. Kundert, Jacob K. White. "Efficient Steady-State Analysis based on Matrix-Free Krylov-Subspace Methods." *Proc. Design Automation Conference,* Santa Clara, California, June 1995.

[10] H. Keller. *Numerical Solution of Two Point Boundary-Value Problems,* SIAM, 1976.

[11] 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.

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

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

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

[15] B. Troyanovsky, Z. Yu, L. So. and R. Dutton, "Relaxation-Based Harmonic Balance Technique for Semiconductor Device Simulation," *Proc. International Conference on Computer-Aided Design*, Santa Clara, California, November 1995.