SILCA-Newton-Krylov: Robust and Efficient Time-Domain VLSI Circuit Simulation by Krylov-Subspace Iterative Methods with Quasi-Newton Preconditioners

Zhao Li and C.-J. Richard Shi
{lz2000,shi}@ee.washington.edu

Dept of EE, University of Washington
Seattle WA, 98195-2500
SILCA-Newton-Krylov: Robust and Efficient Time-Domain VLSI Circuit Simulation by Krylov-Subspace Iterative Methods with Quasi-Newton Preconditioners

Zhao Li and C.-J. Richard Shi
{lz2000,shi}@ee.washington.edu

Dept of EE, University of Washington at Seattle
Seattle WA, 98195-2500

University of Washington, Dept. of EE, UWEETR-2005-001
January 2005

Abstract: In this paper, we present SILCA-Newton-Krylov, a new method for accurate, efficient and robust time-domain VLSI circuit simulation. Similar to SPICE, SILCA-Newton-Krylov uses time-difference and Newton-Raphson for solving nonlinear differential equations from circuit simulation. But different from SPICE, SILCA-Newton-Krylov explores a preconditioned flexible generalized minimal residual (FGMRES) method, instead of traditional LU factorization, to solve the system of linear equations in the inner iteration loop. Our key contribution is to introduce an effective and robust quasi-Newton preconditioning scheme to ensure both the robustness and efficiency of iterative methods. Quasi-Newton methods have been explored previously to reduce the number of LU factorization for circuit simulation, however, their converge rate and stability often deteriorate. In this paper, quasi-Newton methods are applied to construct the preconditioners instead of directly to construct and solve circuit equations. Specifically, a systematic method for adaptive time-step size control and a systematic method to generate piecewise weakly nonlinear (PWNL) definition of nonlinear devices are proposed to construct quasi-Newton preconditioners so that the total number of LU factorization for preconditioning is minimized during the entire time-domain simulation. With the PWNL definition, the preconditioner is kept constant if all nonlinear devices reside in their present operating PWNL regions. When nonlinear devices switch their operating PWNL regions, the low-rank update technique is applied to update the preconditioner efficiently rather than to perform new LU factorization. To reduce further the preconditioning cost, we test incomplete LU preconditioners derived from the factorized full L and U matrices, as well as incomplete LU preconditioners followed by an FGMRES preconditioner. All these techniques combined lead to a new time-domain circuit simulation method, named SILCA-Newton-Krylov, which has been implemented into SPICE3. Experimental results on a collection of analog and digital circuits have shown that SILCA-Newton-Krylov is as robust and accurate as SPICE. SILCA-Newton-Krylov is especially attractive for simulating circuits with a massive amount of parasitic RLC elements for post-layout verification. For example, for a nonlinear circuit with power/ground networks with tens of thousand elements, SILCA-Newton-Krylov is shown to yield the SPICE-like accuracy and over 20X overall CPU time speedup over SPICE3, and furthermore the speedup increases with the size of a circuit.

Keywords: Time-Domain Circuit Simulation, Parasitic-Coupled VLSI Circuits, Piecewise Weakly Nonlinear Device Models, Krylov-Subspace Methods, Quasi-Newton Methods, Preconditioners
1. Introduction

In modern deep-submicron very-large-scale integrated (VLSI) circuits, parasitic effects are no longer ignorable with the higher operation frequency, lower supply voltage and smaller device feature size. As the result, a netlist extracted from a layout contains not only the original designed circuit with transistors and passive devices, but also a massive amount of linear parasitic circuit elements arising from modeling of power/ground planes, substrates, interconnects and vias, etc. It is desirable, especially for high-performance analog and mixed-signal circuit design, to perform full-chip time-domain simulation of such massively parasitic coupled systems [22] before tape out using SPICE-like circuit simulators [20]. This is, however, an extremely time-consuming task, since SPICE-like circuit simulators use LU factorization based direct methods for solving nonlinear differential equations, and for massively parasitic coupled systems, the complexity of LU factorization is approaching its worst case $O(n^3)$, where $n$ is the size of a circuit, instead of its average case $O(n^{1.1-1.5})$ [20].

There are two basic ideas of reducing the cost of LU factorization for circuit simulation. The first idea, which has been explored extensively since the inception of SPICE, belongs to a class of methods known as quasi-Newton methods [6]. Several fixed leading coefficient integration methods [5][10][16] have been proposed for variable time step-size integration to keep the circuit matrix constant – thus to have a less number of LU factorization. Although shown to be effective on general nonlinear circuits and VLSI circuits containing strong parasitic coupling effects [5][16], fixed leading coefficient integration methods are typically less stable than that of their comparable standard implicit integration methods. In the extreme case, a fixed time step-size can be used for the entire transient simulation, so that LU factorization is only performed once for time-domain simulation of linear circuits. However, it is well known that the assumption of a fixed time step-size is not adequate for accurate and efficient simulation of circuits with widely distributed time constants, known as stiff circuits [19]. Virtually all commercial SPICE simulators explore techniques to reduce the number of LU factorization during nonlinear iteration such as Jacobian matrix bypass [20][29]. Very recently, a variant of successive chord method [19] – successive variable chord method [16] has been proposed as an alternate to the Newton-Raphson method [19] to keep the linearized conductance of nonlinear devices constant for a relatively larger voltage/ current range. Along with the successive variable chord method, a heuristic piecewise weakly nonlinear (PWNL) definition of nonlinear devices is introduced in [16] so that the circuit matrix is kept constant if all nonlinear devices in a circuit operate within their present operating PWNL regions during nonlinear iteration. However, it is well known that the convergence rate of quasi-Newton methods can degrade from quadratic as in the case of Newton-Raphson to linear. Techniques have been implemented to update the inverse of a circuit matrix during nonlinear iteration such as applying the Broyden’s method [6] as in [32]. However, with the Broyden’s method, the update has to be performed at each nonlinear iteration step until the convergence is reached. In
the context of successive variable chord method, low-rank update has been applied to update the LU factors when nonlinear devices change their PWNL operating regions [16].

The second idea is to apply Krylov-subspace (iterative) methods [24] for solving the system of linearized equations at each nonlinear iteration step. Despite its straightforwardness, there is no reported successful robust implementation of Krylov-subspace methods for time-domain simulation of general nonlinear circuits yet. Existing uses of Krylov-subspace methods for circuit simulation are limited to specific types of circuit applications, which can be classified into the following four groups. The first successful application is in the shooting-Newton method and harmonic balance analysis for RF circuit simulation [7][15][27][28], where a circuit matrix is generally block diagonally dominant and a preconditioner can be easily determined. The second application was for the analysis of large-scale linear circuits, such as substrate [22] and power/ground networks [3][13][14]. Although shown to be orders of magnitude faster than SPICE, those methods are mainly tailored for purely linear circuits and have difficulty incorporating nonlinear circuits. [4][17] have applied either hierarchical analysis [31] or Krylov-subspace methods for power/ground networks, and incorporated nonlinear circuits based on the Gauss-Seidel relaxation [29]. However, the number of nonlinear iterations might become very large with the Gauss-Seidel relaxation if nonlinear circuits are strongly coupled with power/ground networks. The third application is to use Krylov-subspace model order reduction techniques [11][21] for reducing a large-scale linear network to a smaller one. With this, the simulation time can be shortened. Model order reduction has been implemented for interconnect lines [21], substrates [11], and power/ground networks [26]. However, model order reduction will unfortunately destroy the sparse structure of a circuit matrix. The reduced circuit matrix is generally much denser than the original one. In practice, model order reduction is only effective when the number of ports is much smaller than the number of nodes in a circuit (i.e., interconnect lines). Very recently, multi-port model order reduction has also been explored [8]. Unfortunately, no model order reduction technique is available for general nonlinear circuits. Finally, Krylov-subspace methods have been utilized successfully for the parallel and distributed simulation of ultra-large integrated circuits [1].

In this paper, we explore the use of a particular type of Krylov-subspace methods, called preconditioned flexible generalized minimal residue (FGMRES) method, for time-domain simulation of general nonlinear circuits. Our implementation is as robust as LU factorization based direct methods, but can be orders of magnitude faster than SPICE for the simulation of VLSI circuits with a large amount of linear parasitic elements. The preconditioner we use for the FGMRES method comes directly from the already factorized L and U matrices in a previous time point or a previous nonlinear iteration. The preconditioner is kept constant when time step-sizes vary within a predefined range. Therefore, LU factorization is only performed to update the preconditioner if time step-sizes vary violently, i.e., near breakpoints. Since standard implicit integration methods as used in SPICE have been implemented for constructing circuit matrix equations \( Ax=b \), the stability property with the proposed preconditioned FGMRES method is better than that with fixed leading coefficient integration methods. The key contribution of this paper is to apply a quasi-Newton method to
construct as few preconditioners (therefore LU factorization) as possible for the entire time-domain simulation. Following the similar idea as in [16], we propose a generalized way to partition the entire operating region of nonlinear devices into piecewise weakly nonlinear (PWNL) regions. With this, the preconditioner is kept constant if all nonlinear devices reside in their present operating PWNL regions. When nonlinear devices switch their operating PWNL regions during nonlinear iteration, the low-rank update technique is applied to update the preconditioner efficiently rather than performing new LU factorization. We further explore two types of preconditioners for the best efficiency – a) An incomplete LU preconditioner derived from the factorized full L and U matrices; b) A hybrid preconditioner composed of the incomplete LU preconditioner followed by an FGMRES preconditioner. The entire method can be easily implemented into any SPICE simulator, and can be viewed as a Newton-Krylov [12] implementation of the SPICE-compatible iterative linear centric analysis (SILCA) methodology, thus called SILCA-Newton-Krylov, where a previous quasi-Newton implementation of SPICE-compatible iterative linear centric analysis (SILCA) [16] will be referred to as SILCA-quasi-Newton.

This paper is organized as follows. In Section 2, we provide an overview of quasi-Newton methods and Krylov-subspace methods. The overview was conducted in a unified framework to show that preconditioned Krylov-subspace methods are generally superior to quasi-Newton methods in terms of stability and convergence. Section 3 introduces FGMRES based time-domain circuit simulation with quasi-Newton preconditioning using adaptive step-size control. Section 4 presents the systematic way to generate the PWNL definition of MOSFETs and the low-rank update implementation for preconditioner computation. Section 5 describes the complete SILCA-Newton-Krylov time-domain simulation flow. Experimental results on general nonlinear circuits and power/ground network examples are reported in Section 6. Section 7 concludes the paper.

2. Iterative methods for time-domain circuit simulation

LU factorization has been widely used in classical circuit analysis tools due to its robustness. For example, in SPICE, circuit matrix equations $Ax=b$ are solved by first applying LU factorization to the circuit matrix $A$ and then performing forward/backward substitution [19]. Such methods are called direct methods. Although direct methods are efficient for small-scale to medium-scale circuit simulation, the cost of LU factorization is becoming the dominant per-iteration cost for large-scale circuit simulation incorporating parasitic effects [16][22]. To tackle this problem and to continue exploiting the robustness of LU factorization, a key idea is to reuse the previous LU factorization to solve circuit matrix equations $Ax=b$ for as many time points and/or nonlinear iteration steps as possible. This leads to two categories of iterative methods – quasi-Newton methods and Krylov-subspace methods.
2.1 Quasi-Newton methods

Suppose that we have a LU factorized matrix \( M \), which is considered to be close enough to the circuit matrix \( A \), circuit matrix equations \( Ax = b \) can be solved by Eq. (1) derived from the first-order Taylor expansion with the matrix \( M \) as the approximate Jacobian matrix (the first-order derivative matrix).

\[
x^{(k)} = x^{(k-1)} + M^{-1}(b - Ax^{(k-1)})
\]  

(1)

It should be noted that Eq. (1) will reduce to the Newton-Raphson method if \( M = A \). For nonlinear circuits, Eq. (1) is further written as follows,

\[
x^{(k)} = x^{(k-1)} + M^{-1}(b^{(k-1)} - A^{(k-1)}x^{(k-1)}) = x^{(k-1)} + M^{-1}(-f^{(k-1)})
\]  

(2)

where \( f \) is the vector contributed by function values of linear and nonlinear devices, circuit constitute equations, and numerical integration of charge/flux storage devices. If \( M \) is chosen as a constant matrix during nonlinear iteration, Eq. (2) reduces to the successive chord method [19]. To achieve a similar convergence rate to the Newton-Raphson method, various quasi-Newton methods [6], such as Broyden-Fletcher-Goldfarb-Shanno (BFGS) and Davidson-Fletcher-Powell (DFP), have been proposed to update the matrix \( M \) (or its inverse \( M^{-1} \)) during nonlinear iteration. Recently, the successive variable chord method [16] is proposed to use a constant \( M \) matrix when nonlinear devices reside in their present operating piece-wise nonlinear (PWNL) regions and update the L and U matrices of \( M \) only when nonlinear devices change their operating PWNL regions. In this paper, we adopt the terminology of referring to any iteration method in the form of Eq. (2) as quasi-Newton methods [6]. Therefore, the successive chord method and the successive variable chord method are both quasi-Newton methods. The convergence rate of quasi-Newton methods can degrade from quadratic as that of the Newton-Raphson method to linear. It has been reported in [16] that the number of nonlinear iterations with the successive variable chord method generally increases to two to four times of that with the Newton-Raphson method.

When quasi-Newton methods are applied to charge/flux storage devices in the time domain, it gives rise to iterative integration formulae [16]. The stability and convergence properties of iterative integration formulae have been characterized in [16]. Furthermore, it is shown in [16] that the stability property of iterative integration formulae is generally worse than that of their comparable standard implicit integration formulae unless the number of iterations is large enough.

To alleviate these problems, the line search technique as in Eq. (3) can be applied so that \( x^{(k)} \) is always the optimal value along the search direction of \( M^{-1}(b^{(k-1)} - A^{(k-1)}x^{(k-1)}) \) starting from \( x^{(k-1)} \).

\[
x^{(k)} = x^{(k-1)} + z^{(k-1)}M^{-1}(b^{(k-1)} - A^{(k-1)}x^{(k-1)}) = x^{(k-1)} + z^{(k-1)}M^{-1}(-f^{(k-1)})
\]  

(3)

Different choices of \( z^{(k-1)} \) will lead to different types of quasi-Newton methods. For example, when \( M^{-1}A^{(k-1)} \) is symmetric positive definite, the steepest descent method [24] will be derived if \( z^{(k-1)} \) is chosen as follows,

\[
z^{(k-1)} = \frac{(r, r)}{(M^{-1}A^{(k-1)}r, r)}, \quad r = M^{-1}(b^{(k-1)} - A^{(k-1)}x^{(k-1)}) = M^{-1}(-f^{(k-1)})
\]  

(4)
Similarly, when \( M^T A^{(k-1)} \) is positive definite, the minimal residual iteration method [24] will be obtained by choosing appropriate \( \lambda^{(k)} \).

Quasi-Newton methods with line search based on Eq. (3) have better stability and convergence properties than quasi-Newton methods without line search based on Eq. (2). For example, consider the iterative trapezoid formula [16]

\[
x_n^{(k)} = \frac{2}{h} x_n^{(k)} - \frac{2}{h} x_n^{(k-1)} + 2 \frac{x_n^{(k-1)} - x_{n-1}}{h} - x_{n-1}
\]

(5)

where \( h_n \) is the time step-size from the time point \( t_{n-1} \) to \( t_n \) and \( h \) is the basis time step-size. When it is applied to a one-dimensional test system \( r_x = -x \), we have the following quasi-Newton iteration formula,

\[
x_n^{(k)} = x_n^{(k-1)} + \left( -\frac{2r}{h} - 1 \right)^{-1} \left( -\frac{2r}{h} x_{n-1} + x_{n-1} - \left( -\frac{2r}{h} - 1 \right) x_n^{(k-1)} \right)
\]

(6)

where \( \left( -\frac{2r}{h} x_{n-1} + x_{n-1} \right) \), \( \left( -\frac{2r}{h} - 1 \right) \), and \( \left( -\frac{2r}{h} - 1 \right) \) represent \( b \), \( A \) and \( M \) in Eq. (1), respectively.

Substituting \( \lambda \) in Eq. (3) with \( \left( -\frac{2r}{h} - 1 \right)^{-1} \left( -\frac{2r}{h} - 1 \right) \) derived from Eq. (4), we obtain the following iteration formula:

\[
x_n^{(k)} = x_n^{(k-1)} + \left( -\frac{2r}{h} - 1 \right)^{-1} \left( -\frac{2r}{h} x_{n-1} + x_{n-1} - \left( -\frac{2r}{h} - 1 \right) x_n^{(k-1)} \right) = \left( -\frac{2r}{h} - 1 \right)^{-1} \left( -\frac{2r}{h} x_{n-1} + x_{n-1} \right)
\]

(7)

Eq. (7) represents exactly the Newton-Raphson method. Therefore, for one-dimensional systems, quasi-Newton methods with line search have the same convergence rate as that of converge the same as the Newton-Raphson method. The iterative trapezoid formulae with line search is as stable as the standard trapezoid formula. The reason is that, for one-dimensional systems, the search direction projected to the one-dimensional solution space is the same for quasi-Newton methods and the Newton-Raphson method. With a properly chosen \( \lambda \), quasi-Newton methods with line search is equivalent to the Newton-Raphson method.

Unfortunately, the above observation is no longer true for multi-dimensional systems, since the search direction \( r = b - Ax^{(k-1)} \) projected to the multi-dimensional solution space with quasi-Newton methods is generally different from that with the Newton-Raphson method. Consequently, the stability and convergence properties of quasi-Newton methods with line search are still worse than those of the Newton-Raphson method, when applied to multi-dimensional systems. Noting the improved stability and convergence properties of quasi-Newton methods with line search for one-dimensional systems, a natural idea is to apply multiple search directions for multi-dimensional systems during one iteration instead of only one single search direction as in quasi-Newton methods; i.e., to approximate the search direction of the Newton-Raphson method with the subspace of multiple search directions. This leads to Krylov-subspace methods, to be discussed in the next subsection.
2.2 Krylov-subspace methods

Given an initial guess \( x^{(0)} \) to the circuit matrix equation \( Ax = b \), Krylov-subspace methods seek an approximate solution \( x^{(m)} \) from the subspace of \( \mathbf{K}_m(A, x^{(0)}) \) by imposing the Petrov-Galerkin condition [24]

\[
b - Ax^{(m)} \perp \mathbf{L}_m(A, x^{(0)})
\]

where \( \mathbf{K}_m(A, x^{(0)}) = \text{span}\{r^{(0)}, Ar^{(0)}, A^2r^{(0)}, \ldots, A^{m-1}r^{(0)}\} \), \( r^{(0)} = b - Ax^{(0)} \), and \( \mathbf{L}_m(A, x^{(0)}) \) is a subspace of dimension \( m \). The Arnoldi’s procedure [24] is generally applied to build an orthogonal basis of the Krylov subspace \( \mathbf{K}_m \). Choosing different subspace \( \mathbf{L}_m \), will lead to different types of Krylov-subspace methods. For example, \( \mathbf{L}_m = \mathbf{K}_m \) gives rise to the Arnoldi method, and \( \mathbf{L}_m = A\mathbf{K}_m = \text{span}\{Ar^{(0)}, A^2r^{(0)}, \ldots, A^m r^{(0)}\} \) leads to the generalized minimal residual (GMRES) method.

It is well known that a preconditioner [24] (or a preconditioning matrix) \( M \) is the key to the fast convergence of Krylov-subspace methods. The purpose of a preconditioner is to make the preconditioned matrix \( M^{-1}A \) as close to the identity matrix as possible. With left-preconditioned Krylov-subspace methods, circuit matrix equations to be solved become \( M^{-1}Ax = M^{-1}b \) and the Krylov subspace \( \mathbf{K}_m \) is defined as follows,

\[
\mathbf{K}_m(M^{-1}A, x^{(0)}) = \text{span}\{r^{(0)}, M^{-1}Ar^{(0)}, (M^{-1}A)^2r^{(0)}, \ldots, (M^{-1}A)^{m-1}r^{(0)}\}
\]

where \( r^{(0)} = M^{-1}(b - Ax^{(0)}) \). It is not surprising that the effect of the preconditioner \( M \) on preconditioned Krylov-subspace methods is similar to that of the approximate Jacobian matrix \( M \) on quasi-Newton methods. It is observed that preconditioned Krylov-subspace methods will reduce to quasi-Newton methods with line search if \( m \) is equal to 1. For example, it can be verified that when \( m \) is equal to 1, the Arnoldi method will reduce to the steepest descent method and the GMRES method to the minimal residual iteration method [24].

Due to the multi-dimensional orthogonal Krylov subspace \( \mathbf{K}_m \) used during one iteration, it is expected that the stability and convergence properties of preconditioned Krylov-subspace methods be better than those of quasi-Newton methods and close to those of the Newton-Raphson method. This has been confirmed by experimental results on general nonlinear circuits in Section 6.1. It should be noted that preconditioned Krylov-subspace methods require extra cost for each iteration including the preconditioning procedure, matrix-vector products, the Arnoldi procedure, etc., which can be significant when \( m \) is large. Therefore, a good preconditioner is the key to exploit full benefits of Krylov-subspace methods. However, due to the orthogonal Krylov subspace \( \mathbf{K}_m \) used during each iteration, the requirement for the preconditioner \( M \) of Krylov-subspace methods is much lower than that for the approximate Jacobian matrix \( M \) of quasi-Newton methods. For example, incomplete LU factorization [24] can be applied to circuit matrix \( A \) as an effective preconditioner.

In summary, although the per-iteration cost of preconditioned Krylov-subspace methods is higher than that of quasi-Newton methods, the stability and convergence properties of preconditioned Krylov-subspace
methods are better and close to those of the Newton-Raphson method. Therefore, preconditioned Krylov-subspace methods with effective preconditioners can be robust for general nonlinear circuit simulation.

3. Time-Domain Simulation with a Preconditioned Flexible GMRES Method

With an overview of the flexible GMRES method in Section 3.1, we introduce how to construct effective preconditioners based on quasi-Newton methods in Section 3.2. Section 3.3 presents adaptive time-step size control for preconditioner computation. Section 3.4 describes two implementations of incomplete LU preconditioners.

3.1 The flexible GMRES iterative method

We choose to use the flexible GMRES (FGMRES) method [23], an extension of the original right-preconditioned GMRES method [24], to solve the system of linear(ized) circuit equations \((AM^{-1})Mx = b\). The basic flow of the FGMRES method is illustrated in Algorithm I. It should be noted that Algorithm I presented here is only for the explanation purpose, which has to be improved in practice for the best efficiency.

The FGMRES method is chosen since flexible preconditioners can be used during the GMRES solving process. It is known that the use of flexible preconditioners can lead to better efficiency and robustness. For example, Krylov-subspace methods have been used as preconditioners and shown to be effective for harmonic balance analysis [15][28]. Furthermore, preconditioners can be changed during the FGMRES solving process to explore the advantages of different preconditioners for the best performance.

Algorithm I. The basic flow of the FGMRES method.

1. Choose an initial guess \(x_0\) and a maximum iteration number \(m\) for restarting the Arnoldi process.

   Initialize \((m+1) \times m\) matrix \(H_m\) to zero.

2. Arnoldi process

   (a) Compute \(r_0 = b - Ax_0\) and \(v_1 = r_0 / \|r_0\|\);

   (b) For \(j = 1, \ldots, m\)

      Compute \(z_j = M^{-1}v_j, \quad w = Az_j;\)

      For \(i = 1, \ldots, j\)

      \(h_{i,j} = \langle w, v_j \rangle, \quad w = w - h_{i,j}v_j;\)

      Compute \(h_{j+1,j} = \|w\|\) and \(v_{j+1} = w / \|w\|;\)

   (c) Define \(Z_m = [z_1, \ldots, z_m].\)
3.2 Quasi-Newton preconditioners

Now we consider how to select robust and effective preconditioners for FGMRES-based time-domain circuit simulation. In the context of general circuit simulation, $A$ is a general invertible matrix and does not have the diagonal dominance property. Clearly if we select $M$ to be $A^{-1}$, then $M^TA$ will be the identify matrix. Therefore we would like to select $M$ to be as close to $A^{-1}$ as possible so that $M^TA$ is as close to the identify matrix as possible. With circuit matrix $A$ arising from the inner iteration loop, $M$ can be selected as $A^{-1}$ computed from a previous nonlinear iteration point or a previous time point. Then the problem of interest has the following two objectives:

1. to use as few $A^{-1}$ as possible during the entire time domain circuit simulation, and
2. to use as few matrix and vector products as possible ($M^TA$ close to the identify matrix).

These are exactly the objectives of quasi-Newton methods for circuit simulation.

Quasi-Newton methods involve two steps: how to select time-step sizes and how to approximate the Jacobian matrices. We describe adaptive step-size control in Section 3.3 and how to approximate and calculate the Jacobian matrices in Section 4.

3.3 Adaptive step-size control for preconditioner computation

Suppose that $h$ is the base time-step size, and $h_n$ is the current time-step size. To develop a guideline for adaptive step-size control for preconditioner computation, let us write the system of linear(ized) circuit equations as:

$$ Gx + Cx = b $$

where $G$ and $C$ represent the conductance and susceptance (capacitance) matrices, and $b$ is the vector of input sources and residuals of nonlinear devices. Replace time derivatives by the standard trapezoid formula, we have

$$ \left( G + \frac{2C}{\alpha h} \right) x^{(k)} + \frac{2C}{\alpha h} x_{n-1} + Cx_{n-1} = b $$

where $h_n = \alpha h$. To solve the above equation with preconditioned Krylov-subspace methods, the preconditioner we use is $\left( G + \frac{2C}{h} \right)$, which should be as close to $\left( G + \frac{2C}{\alpha h} \right)$ as possible. Therefore, we introduce a parameter $\eta < 1$ so that the preconditioner should satisfy the following inequality

$$ \left| \left( G + \frac{2C}{h} \right) \left( G + \frac{2C}{\alpha h} \right)^{-1} - I \right| = \left| G + \frac{2C}{h} \left( 1 - \frac{1}{\alpha} \right) \frac{2C}{h} \right| < \eta < 1 $$
where \( \| \cdot \| \) represents the spectral radius of the iteration matrix. The above inequality can be re-written as

\[
\frac{|1 - 1/\alpha|}{1 - z} < \eta < 1
\]

where \( z = -h/(2\tau) \) and \( \tau \) is an eigenvalue of the matrix \( G^{-1}C \). Let us refer to the region defined by the above inequality as the effective preconditioner region. Then we can draw the effective preconditioner region for both \( \alpha = 0.625 \) and \( \alpha = 2.5 \) as in Figure 1.

![Figure 1](image)

**Figure 1.** The effective preconditioner region for \( \alpha = 0.625 \) and \( \alpha = 2.5 \).

From the inequality above, the region \( |z - 1| \leq |1 - 1/\alpha| \) is not effective. To ensure the effectiveness of preconditioners for any decaying or stable oscillating system (\( \text{Re}(z) \leq 0 \)); i.e., to have the effective region to includes all of the left half of the complex \( z \)-plane, we must choose \( \alpha > 0.5 \). In this case, \( |1 - 1/\alpha| < 1 \), the effective preconditioner region for a system with increasing components or unstable oscillation (\( \text{Re}(z) > 0 \)) lies in the region of \( |z - 1| \leq |1 - 1/\alpha| < 1 \). In practice, to have most effective preconditioners, \( 0.625 < \alpha < 2.5 \) is used in our implementation. We note that the effective preconditioner region is exactly the same as the convergence region for the iterative integration formulae in SILCA-quasi-Newton [16].

### 3.4 Incomplete LU Factors as Preconditioners

Three types of preconditioners have been tested in our experiments:

1) A LU preconditioner composed of the factorized full L and U matrices.

2) An incomplete LU preconditioner composed of matrices approximated from the factorized full L and U matrices – a matrix element \( l(i,j) \) is removed if \( |l(i,j)| < c \times \max(|l(*,j)|) \) in L or \( |u(i,j)| < c \times \max(|u(i,*)|) \) in U. \( c \) is a coefficient for the incomplete LU factorization, 0.001 is mainly used in our experiments.

Since the incomplete LU preconditioner we use is derived from the already factorized L and U matrices, it is more robust and effective than general incomplete LU preconditioners in the cost of more memory resources.
3) A hybrid preconditioner composed of the incomplete LU preconditioner and a following FGMRES preconditioner. The purpose is to reduce the number of total FGMRES iterations by using accurate preconditioners. Further, it should be noted that the FGMRES preconditioner can be added to or dropped from the hybrid preconditioner for efficiency depending on the following two criteria:

a. If the residual of $||b-Ax||$ is already small enough after the incomplete LU preconditioner, it is not necessary to further use the FGMRES preconditioner.

b. For the first several top-level FGMRES iterations, it is recommended to apply the hybrid preconditioner for accuracy, since it is the first several vectors that dominate the Krylov subspace. However, since the FGMRES preconditioner also uses the incomplete LU preconditioner for preconditioning, we have shown in Section 6.2 that the hybrid preconditioner is not always superior to the incomplete LU preconditioner alone for time-domain simulation.

4. Quasi-Newton preconditioner computation by piecewise weakly nonlinearity partitioning

In this section, we present a systematic technique for quasi-Newton preconditioner computation. The central idea is to partition any nonlinear device function into a collection of regions, where in each region the nonlinear function is a weakly nonlinear function, i.e., its derivatives can be approximated by a constant (called chord). As an example, Figure 2 shows an example of the PWNL definition of a nonlinear function, where three PWNL regions are defined with three different chords (fixed first-order derivatives), one for each region.

![Figure 2. The PWNL definition of a nonlinear function.](image-url)
Now consider how to generate PWNL regions automatically for arbitrary nonlinear functions. Suppose that nonlinear iteration is performed within a PWNL region of a nonlinear function \( f(x) \) to solve \( f(x) = 0 \), the nonlinear iteration equation can be expressed by,

\[
x_{i+1} = x_i - \frac{f(x_i)}{g}
\]

(10)

where \( g \) is the chord for this PWNL region. Let the exact solution be \( x^* = x_i + \epsilon_i = x_{i+1} + \epsilon_{i+1} \). Subtracting \( x^* \) from the both sides of Eq. (10) gives

\[
\epsilon_{i+1} = \epsilon_i + \frac{f(x_i)}{g}
\]

(11)

After applying the Taylor expansion on \( f(x) \) at \( x_i \), we obtain the following error estimation,

\[
\epsilon_{i+1} \approx \epsilon_i (1 - \frac{f'(x_i)}{g}) - \epsilon_i^2 \frac{f''(x_i)}{2g}
\]

(12)

From Eq. (12), if \( g \) is always equal to \( f'(x_i) \), as in the Newton-Raphson method, the convergence rate is quadratic. The smaller the \( |1 - f'(x_i)/g| \) is, the closer to the quadratic convergence rate Eq. (12) is. On the other hand, the larger the \( |1 - f'(x_i)/g| \) is, the larger the range of a PWNL region could be. Therefore, there exists a tradeoff between the convergence rate and the range of a PWNL region. In practice, we define the following condition with a parameter \( 0 < \delta < 1 \),

\[
\left| 1 - \frac{f'(x_i)}{g} \right| < \delta
\]

(13)

If the maximum and minimum first-order derivatives for the studied PWNL region are \( f'_{\text{max}} \) and \( f'_{\text{min}} \) and both of them are positive (this assumption is generally true in practice except specific nonlinear devices, such as tunnel diodes), it can be derived from Eq. (13) that the chord for this PWNL region should be chosen as follows,

\[
\frac{f'_{\text{max}}}{1 + \delta} < g < \frac{f'_{\text{min}}}{1 - \delta}
\]

(14)

It should be noted that the above analysis is done in the context that nonlinear iterations are performed within a PWNL region. In case that nonlinear iterations run across two or more PWNL regions, such as the example shown in Fig. 3 where the exact solution resides at the boundary of two PWNL regions, the condition of \( x_2 < x_0 \) should be satisfied to achieve the convergence. Hence \( g_1 \) and \( g_2 \) in Fig. 3 should satisfy the following inequality:

\[
(g_1 - a) + (g_2 - a) > \frac{f(x_i) - f(x_0)}{x_1 - x_0} - a
\]

(15)
In our implementation, the chord is chosen to be the maximum first-order derivative in each PWNL region to satisfy both Eqs. (14) and (15). The maximum first-order derivative could be computed with the knowledge of nonlinear device model behaviors, such as monotonicity. Note that PWNL regions for a nonlinear function are equivalent to piecewise constant (PWC) regions for first-order derivatives of the same nonlinear function.

Now that the chord is chosen to be the maximum first-order derivative in each PWNL region, PWNL regions for MOSFETs can be generated automatically with the following rules:

(1) The maximum voltages of $V_{ds}$ and $V_{gs}$ are predefined. In our experiments, we use $V_{dd}$ as the maximum voltage for both of them. Given model parameters, the maximum $g_{ds}$ and $g_m$ for all operating regions can then be calculated. It should be noted that the actual voltages of $V_{ds}$ and $V_{gs}$ could be larger than $V_{dd}$. In such cases, PWC region values of $g_{ds}$ and $g_m$ are extended by the extrapolation based on rule (2).

(2) With a predefined $0 < \delta < 1$, PWC region values for $g_{ds}$ and $g_m$ can be calculated as follows,

$$g_{n} = g_{\max}$$
$$g_{i+1} = (1 - \delta) g_i, \quad i = n, n-1, ..., 2$$

It is observed that $g_i$ satisfies Eqs. (14) and (15). In case that extrapolation is required, extrapolated PWC region values for $g_{ds}$ and $g_m$ can be calculated as follows,

$$g_{n} = g_{\max}$$
$$g_{i+1} = \frac{g_i}{(1 - \delta)}, \quad i = n, n + 1, ...$$

(3) A lower bound of $g_{ds}$ and $g_m$ is predefined, so that rule (2) will stop whenever $g_{ds}$ and $g_m$ are less than the predefined lower bound. This is necessary to avoid a PWC region for $g_{ds}$ and $g_m$ to be too narrow.

(4) A voltage step-size is chosen so that at least one PWC region exists for each of the calculated PWC region values of $g_{ds}$ and $g_m$ in the $V_{ds} - (V_{gs} - V_{th})$ plane. Otherwise, either a smaller voltage step-size should be chosen or the lower bound of $g_{ds}$ and $g_m$ should be adjusted in rule (3). A uniform voltage
step-size has been used in our implementation for the simplicity. Once the voltage step-size is finalized, $g_{ds}$ and $g_m$ at each grid point of the $V_{ds}$-$V_{gs}$ plane can be evaluated, so that each patch of the $V_{ds}$-$V_{gs}$ plane will be allocated to a PWC region of $g_{ds}$ and $g_m$.

As an example, using the above rules, the PWC regions for $g_{ds}$ and $g_m$ of the MOSFET level 1 model are shown in Figs. 4 and 5, respectively, where $\delta$ is set to $1/3$. It can be seen that there are a total of six PWC region values for $g_{ds}$ and $g_m$ (including the cutoff region #0). It should be noted that effects due to $V_{be}$ have been incorporated into $V_{th}$. For the MOSFET level 1 model, $g_{subh}$ has a simple relationship with $g_m$ [25]:

$$g_{subh} = g_m \times \frac{dV_{th}}{dV_{ab}} = g_m \times \frac{\gamma}{2\sqrt{\Phi + V_{ab}}}$$

(16)

For the simplicity, we use the maximum $dV_{th} / dV_{ab} = \gamma / (2\sqrt{\Phi})$. Therefore, the PWC regions for $g_{subh}$ are the same as those for $g_m$. The proposed method can be easily implemented using model compilers such as MCAST for automatic generation of piece-wise weakly nonlinear regions for any nonlinear device with any device model [29].

Figure 3. PWC regions for $g_{ds}$ in the $V_{ds}$-$V_{gs}$ plane.

Figure 5. PWC regions for $g_m$ in the $V_{ds}$-$V_{gs}$ plane.
Different \( \delta \) values will lead to different numbers of PWNL regions generated for nonlinear devices. Although a smaller \( \delta \) could reduce the number of FGMRES iterations during transient simulation, it would unfortunately increase the number of low-rank updates due to more PWNL regions generated for nonlinear devices. In our experiments, to achieve a tradeoff between the number of low-rank updates and the number of FGMRES iterations, \( \delta \) is chosen between 1/4 (8 PWNL regions) and 1/3 (6 PWNL regions) so that the number of PWNL regions for MOSFETs is below 10.

If only a few nonlinear devices change their operating PWNL regions during nonlinear iteration, the low-rank update technique [9] can be used to efficiently update the previously factorized L and U matrices rather than LU factorization. Therefore, in practice, the ratio of the number of nonlinear devices switching their operating PWNL regions vs. the total number of nonlinear and linear devices could be used as a guide for choosing either the low-rank update technique or LU factorization. To utilize the low-rank update technique, it is required that the new circuit matrix \( A_{\text{new}} \) be derived from the old circuit matrix \( A_{\text{old}} \) as follows,

\[
A_{\text{new}} = A_{\text{old}} + cr^T
\]

where \( A_{\text{new}} \) and \( A_{\text{old}} \) are both \( n \times n \) matrices, \( c \) and \( r \) are both \( n \times m \) matrices, and \( m < n \).

When a MOSFET switches its operating PWNL region within either the normal mode or the reverse mode, the contribution of this MOSFET to the circuit matrix is changed as follows,

\[
\begin{bmatrix}
\Delta g_{dr} & \Delta g_{mn} & -\Delta g_{dr} & -\Delta g_{mn} & -\Delta g_{mbs} & \Delta g_{mbs} \\
-\Delta g_{dr} & -\Delta g_{mn} & \Delta g_{dr} + \Delta g_{mn} + \Delta g_{mbs} & -\Delta g_{mbs}
\end{bmatrix}
\]

which can be represented by the following rank-one update \((m=1)\),

\[
\begin{bmatrix}
\sqrt{a} & \Delta g_{dr} / \sqrt{a} & -\Delta g_{dr} / \sqrt{a} & \Delta g_{mbs} / \sqrt{a}
\end{bmatrix}
\]

\[
a = \max(|\Delta g_{dr}|, |\Delta g_{mn}|, |\Delta g_{mbs}|)
\]

When a MOSFET switches its operating PWNL region from the normal mode to the reverse mode, the contribution of this MOSFET to the circuit matrix is changed as follows,

\[
\begin{bmatrix}
\Delta g_{dr} + g_m + g_m & \Delta g_{mn} - 2g_m & -\Delta g_{dr} - \Delta g_{mn} - \Delta g_{mbs} + g_m & \Delta g_{mbs} - 2g_m \\
-\Delta g_{dr} - g_m - g_m & -\Delta g_{mn} + 2g_m & \Delta g_{dr} + \Delta g_{mn} + \Delta g_{mbs} - g_m - g_m & -\Delta g_{mbs} + 2g_m
\end{bmatrix}
\]

which also can be represented by a similar rank-one update format. If multiple MOSFETs switch their operating PWNL regions during nonlinear iteration, a series of rank-one updates will be performed.

We note that to preserve the SPICE-like convergence property for nonlinear circuits during transient simulation, the PWNL definitions of nonlinear devices are used for computing the preconditioner for the FGMRES method, and the original nonlinear device models are used for constructing circuit matrix equations \( Ax=b \). This is especially advantageous when incorporating nonlinear capacitors into the proposed preconditioned FGMRES method, since simplified linear capacitors could be used for the preconditioner while original complicated nonlinear capacitors are kept for building circuit matrix equations. However, the
PWNLD definitions of nonlinear devices could still be used directly as device models to reduce the cost of device model evaluation, as implemented in [18], since PWC values of first-order derivatives have been pre-calculated and stored in the PWNL definitions of nonlinear devices; this would significantly save device evaluation and setup time.

5. FGMRES-based transient simulation flow

The flow of the proposed SILCA-Newton-Krylov method for time-domain simulation is shown in Algorithm II. It can be seen that LU factorization is only performed when time step-sizes vary out of the predefined $h_n/h$ range (0.625 < $h_n/h$ < 2.5 has been set to make comparison with SILCA-quasi-Newton [16]). In other cases, L and U matrices are either kept unchanged or updated by the low-rank update technique when nonlinear devices change their operating PWNL regions. During the whole process, L and U matrices are used for preconditioning the FGMRES method.

Algorithm II. Transient simulation flow.

```
DC operating point analysis
Choose an initial step size $h_0$, the basis step size $h = h_0, t = 0$
WHILE ($t < T_{\text{final}}$){
    OUTER LOOP: do{
        $\alpha = h_n/h$, iter_no = 0
        INNER LOOP: do{
            IF (0.625 < $\alpha$ < 2.5){
                IF (PWNL region is changed)
                    Apply low-rank update on L/U matrices
            } ELSE{
                IF (iter_no == 0)
                    Apply LU factorization
                ELSE{
                    IF (PWNL region is changed)
                        Apply low-rank update on L/U matrices
                }
            }
        }
    }
    Apply the preconditioned FGMRES method
```
\[
\text{iter_no} = \text{iter_no} + 1
\]

\}\) while (not converged)

Choose a new \(h_n\) based on LTE requirement

\}\) while (LTE greater than predefined error limit)

\[t = t + h_n\]

6. Experimental results

6.1 General nonlinear circuits

To verify the robustness of the proposed SILCA-Newton-Krylov method for the simulation of general nonlinear circuits, several digital, analog and RF circuits have been tested. The simulation results are summarized in Table I. During the test, the full LU preconditioner has been used for the FGMRES method. Parameter \(\delta\) is set to 1/3 to generate PWNL regions for MOSFETs.

<table>
<thead>
<tr>
<th>Test Circuits</th>
<th>SPICE3</th>
<th></th>
<th>FGMRES w LU Preconditioner</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Total</td>
<td>#Accepted</td>
<td>#Tran</td>
<td>#Total</td>
</tr>
<tr>
<td></td>
<td>points</td>
<td>points</td>
<td>iteration</td>
<td>points</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#Tran</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#Tran LUD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#Rank-one</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>update</td>
</tr>
<tr>
<td>1 Inverter</td>
<td>142</td>
<td>127</td>
<td>340</td>
<td>141</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>127</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>338</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>63</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>2 20-stage inverter</td>
<td>356</td>
<td>260</td>
<td>1159</td>
<td>369</td>
</tr>
<tr>
<td>chain</td>
<td></td>
<td></td>
<td></td>
<td>266</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1185</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>643</td>
</tr>
<tr>
<td>3 Nand2</td>
<td>132</td>
<td>123</td>
<td>305</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>123</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>305</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>51</td>
</tr>
<tr>
<td>4 One-shot trigger</td>
<td>431</td>
<td>371</td>
<td>1360</td>
<td>431</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>371</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1348</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>169</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>639</td>
</tr>
<tr>
<td>5 Comparator</td>
<td>140</td>
<td>126</td>
<td>410</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>126</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>47</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>163</td>
</tr>
<tr>
<td>6 Opamp follower</td>
<td>220</td>
<td>148</td>
<td>814</td>
<td>221</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>149</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>819</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>7 Ring oscillator</td>
<td>243</td>
<td>173</td>
<td>1020</td>
<td>256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>176</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1060</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>746</td>
</tr>
<tr>
<td>8 VCO</td>
<td>1281</td>
<td>887</td>
<td>4529</td>
<td>1280</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>887</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4524</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2251</td>
</tr>
<tr>
<td>9 Power amplifier</td>
<td>873</td>
<td>587</td>
<td>3451</td>
<td>840</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>581</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3331</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1761</td>
</tr>
</tbody>
</table>

It can be seen from Table I that, with the preconditioned FGMRES method, the number of LU factorization (#Tran LU) during transient simulation is reduced dramatically compared to that with SPICE3 (#Tran LU = #Tran iteration). Further, the number of total simulated time points (#Total points), the number of accepted time points (#Accepted points), and the number of transient iterations (#Tran iteration) with the preconditioned FGMRES method are kept almost the same as those with SPICE3, and are generally less than those with quasi-Newton base SILCA-quasi-Newton [16]. Figure 6 shows the histogram of the number of MOSFETs that switch their operating PWNL regions at each time point in the 20-stage inverter chain. Clearly the number of switching MOSFETs is much less than the total number of MOSFETs (40) in the inverter chain.
Figure 6. The number of MOSFETs switching PWNL regions in the inverter chain.

Figure 7. The average number of FGMRES iterations for general nonlinear circuits.

Figure 7 shows the average number of FGMRES iterations in each FGMRES solving process for test circuits (the dimension $m$ of the Krylov subspace $\mathbf{K}_m$). The average number of FGMRES iterations is below 5 for most of test circuits. For the power amplifier example, the average number of FGMRES iterations is about 7.6, which is higher than that for other test circuits. The reason is that most MOSFETs in the power amplifier operate in the PWNL region #1 during transient simulation, which can be seen by comparing Figs. 4, 5 and 8. The PWNL region #1 is not modeled as accurately as other PWNL regions. Therefore, the efficiency of the incomplete LU preconditioner becomes worse.
6.2 Power/ground network examples

To test the efficiency of the proposed SILCA-Newton-Krylov method for the simulation of parasitic-coupled VLSI circuits, a power/ground network example shown in Fig. 9, which is similar to that used in [4][16], is simulated. The power and ground supply networks are modeled as two RCL mesh layers (parasitic coupling capacitors are not shown). In our example, between these two layers is a 20-stage inverter chain representing nonlinear circuits, different inverters of which are connected to different power/ground nodes. Furthermore, RCL loads are added for each inverter to model interconnect lines between adjacent stages. The size of two RCL meshes is changed to vary the number of elements. Parameter $\delta$ is set to $1/3$ to generate PWNL regions for MOSFETs.
Table II. Simulation results for the power/ground network example ($\varepsilon = 1e-10$, LU preconditioner).

<table>
<thead>
<tr>
<th>#Elems</th>
<th>SPICE3</th>
<th>Preconditioned FGMRES</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Tran Iter</td>
<td>Tran LU (sec)</td>
<td>Tran LU Iter</td>
</tr>
<tr>
<td>4002</td>
<td>4023</td>
<td>371.20</td>
<td>403.99</td>
</tr>
<tr>
<td>3480</td>
<td>4006</td>
<td>4.549e4</td>
<td>4.760e4</td>
</tr>
<tr>
<td>6160</td>
<td>4377</td>
<td>1.797e5</td>
<td>1.848e5</td>
</tr>
</tbody>
</table>

Table II summarizes the simulation results for the power/ground network example using SPICE3 and the FGMRES method with the full LU preconditioner. The error tolerance $\varepsilon$ is set to $1e-10$ for the preconditioned FGMRES method. The speedup over SPICE3 is over 10X for the largest example we test. It can be expected that more speedup could be achieved for larger power/ground networks. The average number of FGMRES iterations in each FGMRES solving process ($\#$FGMRES Iter / $\#$Tran Iter) is about 5. It is worthy noting that the number of LU factorization ($\#$Tran LU) is reduced greatly with the preconditioned FGMRES method compared to SPICE3 ($\#$Tran LU = $\#$Tran Iter). It is observed that the speedup over SPICE3 with the preconditioned FGMRES method is less than that with SILCA-quasi-Newton [16]. This is due to the fact that the per-iteration cost with the preconditioned FGMRES method is higher than that with SILCA-quasi-Newton. However, as shown in Table II, the number of transient iterations with the preconditioned FGMRES method has been kept almost the same as that with SPICE3, which shows that the SPICE-like convergence property has been preserved. Further, it will be shown next that more speedup over SPICE3 can be achieved by applying efficient preconditioners.

Table III. Simulation results for the power/ground network example ($\varepsilon = 1e-10$, ILU preconditioner).

<table>
<thead>
<tr>
<th>#Elems</th>
<th>#Tran Iter</th>
<th>#Tran LU</th>
<th>#FGMRES Iter</th>
<th>Tran LU (sec)</th>
<th>FGMRES ES (sec)</th>
<th>Tot Tran (sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>4002</td>
<td>4241</td>
<td>52</td>
<td>24208</td>
<td>4.83</td>
<td>67.20</td>
<td>84.01</td>
<td>4.81</td>
</tr>
<tr>
<td>3480</td>
<td>4199</td>
<td>53</td>
<td>28311</td>
<td>688.40</td>
<td>3081.03</td>
<td>4066.19</td>
<td>11.71</td>
</tr>
<tr>
<td>6160</td>
<td>4254</td>
<td>56</td>
<td>28901</td>
<td>2323.74</td>
<td>5995.01</td>
<td>8938.69</td>
<td>20.68</td>
</tr>
</tbody>
</table>
The simulation results of the FGMRES method with the incomplete LU (ILU) preconditioner are shown in Table III. It is seen that the FGMRES method with the ILU preconditioner achieves the best speedup over SPICE3 for the largest power/ground network – 20.68X, which is about 2X speedup over the FGMRES method with the LU preconditioner (The run time comparison is shown in Fig. 10.). The reason is that the number of matrix elements in the ILU preconditioner is much less than those in the LU preconditioner. For the power/ground network example with 61602 elements, the histogram of the number of L and U matrix elements during transient simulation are shown in Fig. 11. The number of matrix elements in the full L and U matrices is 3116915 for this example. The number of nonzero matrix elements in the ILU preconditioner is reduced to about 1/4 to 1/15 of that of the full LU preconditioner. Consequently, the cost of the preconditioning procedure with the ILU preconditioner has been reduced greatly. The histogram of the number of FGMRES iterations per FGMRES solving process is shown in Fig. 12. The average number of FGMRES iterations is about 6 to 7 with the ILU preconditioner.

**Figure 10.** The run time vs. the number of elements in the power/ground network.

**Figure 11.** The histogram of the number of L and U matrix elements.
Figure 12. The histogram of the number of FGMRES iterations with the ILU preconditioner.

Figure 13 shows the variation of the average number of FGMRES iterations in each FGMRES solving process with the coefficient $c$ for the ILU preconditioner. It can be seen that the larger the coefficient $c$ is, the more FGMRES iterations will be taken. The reason is that more matrix elements in the L and U matrices will be removed with a larger coefficient $c$, which causes the incomplete LU preconditioner less effective. However, the cost of evaluating the ILU preconditioner is reduced with an increased coefficient $c$. Therefore, it is expected that there will be an optimum coefficient $c$ to minimize the FGMRES cost. This has been confirmed by Fig. 14, which shows the variation of the average FGMRES cost in each FGMRES solving process with the coefficient $c$ for the ILU preconditioner. The average FGMRES cost in each FGMRES solving process is minimized near $c=0.001$, which has been used in our experiments.

Figure 13. The variation of the average number of FGMRES iterations with the ILU coefficient.
Figure 14. The variation of the average FGMRES cost with the ILU coefficient.

Table IV. Simulation results for the power/ground network example ($\varepsilon = 1e-10$, ILU+FGMRES hybrid preconditioner).

<table>
<thead>
<tr>
<th>#Elems</th>
<th>#Tran Iter</th>
<th>#Tran LU</th>
<th>#Top-level FGMRES Iter</th>
<th>#Total FGMRES Iter</th>
<th>Tran LU (sec)</th>
<th>FGMRES (sec)</th>
<th>Tot Tran (sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>4002</td>
<td>4277</td>
<td>48</td>
<td>12661</td>
<td>27034</td>
<td>4.35</td>
<td>77.01</td>
<td>93.96</td>
<td>4.30</td>
</tr>
<tr>
<td>34802</td>
<td>4214</td>
<td>51</td>
<td>13164</td>
<td>33267</td>
<td>662.59</td>
<td>3598.73</td>
<td>4560.36</td>
<td>10.44</td>
</tr>
<tr>
<td>61602</td>
<td>4116</td>
<td>57</td>
<td>12688</td>
<td>32252</td>
<td>2324.96</td>
<td>6056.12</td>
<td>8936.80</td>
<td>20.68</td>
</tr>
</tbody>
</table>

The simulation results of the FGMRES method with the hybrid preconditioner are shown in Table IV. A larger error tolerance $\varepsilon = 1e-1$ has been set to the FGMRES preconditioner. Although the number of top-level FGMRES iterations has been reduced compared to that in Table III, the number of total FGMRES iterations has been increased unfortunately. Therefore, the speedup over SPICE3 with the hybrid preconditioner is not as good as that with the ILU preconditioner alone. The main reason is that the ILU preconditioner is already effective enough for transient simulation in terms of the number of FGMRES iterations per FGMRES solving process, as shown in Fig. 12. The overhead cost of further using the FGMRES preconditioner is even more than the induced saving. However, the hybrid preconditioner does reduce the number of FGMRES iterations per FGMRES solving process, as shown in Fig. 15. Therefore, it can be expected that the hybrid preconditioner will gain when the convergence of the FGMRES method is slow. For example, in harmonic
balance analysis for RF circuits, it has been observed that Krylov-subspace methods with hybrid preconditioners [15][28] converge better than those with block diagonal preconditioners alone [7].

Figure 15. The histogram of the number of FGMRES iterations with the hybrid preconditioner.

Figure 16. The output waveform of the power/ground network example.

The FGMRES method has been tested to produce the same accurate results as SPICE3. As an example, Figure 16 shows the output waveforms of the inverter chain between the power and ground networks simulated using SPICE3 and using the FGMRES. It can be seen that the low-level voltage of the output is larger than the ideal ground voltage due to the IR drop and $L \cdot dI/dt$ effects.

7. Conclusion

In this paper, we have studied the relationship between Krylov-subspace methods and quasi-Newton methods. It is concluded that preconditioned Krylov-subspace methods can be robust for circuit simulation in terms of the stability and convergence properties. A preconditioned FGMRES method with quasi-Newton preconditioners has been presented and implemented to speed up time-domain simulation of parasitic-coupled VLSI circuits. Systematic methods of adaptive time-step size control and piece-wise weakly
nonlinearity (PWNL) partitioning of any nonlinear device have been proposed to minimize the cost of LU factorization during the entire variable time step-size transient simulation. With PWNL, the preconditioner cost can be reduced with the low-rank update technique. Two types of preconditioners have been studied to further reduce the preconditioning cost – an incomplete LU preconditioner derived from the factorized full L and U matrices and a hybrid preconditioner composed of the incomplete LU preconditioner followed by a FGMRES preconditioner. Experimental results have shown that these techniques dramatically reduce the number of required LU factorization during transient simulation. Orders of magnitude speedup has been achieved on power/ground network examples with SPICE-like accuracy.

References


