Power Grid Analysis Using a Flexible Conjugate Gradient Algorithm with Sparsification

Peter Feldmann
IBM T. J. Watson Research Center
1101 Kitchawan Rd.
Yorktown Heights, NY 10598
Email: feldmann@watson.ibm.com

Roland W. Freund
Department of Mathematics
University of California, Davis
One Shields Avenue
Davis, CA 95616
Email: freund@math.ucdavis.edu

Emrah Acar
IBM Austin Research Lab
1101 Kitchawan Rd.
Yorktown Heights, NY 10598
Email: emrah@us.ibm.com

***Abstract*— In this paper, we present a flexible conjugate gradient method that is tailored to the solution of the truly large-scale linear systems arising in VLSI power grid analysis. The algorithm allows changing preconditioners and sparsification of the search direction at each iteration. As a consequence, this variant of the conjugate gradient algorithm becomes compatible with implementations that avoid the modeling and representation of numerically irrelevant portions of the problem, and take natural advantage of the local and sparse nature of the solutions. The paper presents the flexible conjugate gradient algorithm in detail and explores strategies for preconditioning and sparsification. The algorithm is applied to a number of realistic power grid examples.**

I. Introduction

The design and verification of today's very large-scale integrated (VLSI) circuits involve some extremely challenging numerical problems. One of the truly large-scale problems in this area is power grid analysis. Power grids are modeled as networks with up to tens of millions nodes. Due to the use of Controlled Collapse Chip Connections, (C4), also called Solder Bumps in packaging, the internal chip power grid is connected to the board level power supply network not only on the periphery but at points distributed across the entire surface of the chip.

Steady-state analysis of power grids requires the solution of correspondingly large sparse symmetric positive definite linear systems. The coefficient matrices of these systems have the structure of weighted Laplacians on three-dimensional grids, but with 'boundary' conditions given on a subset of the interior grid points, (the points corresponding to the C4s).

The architecture of the power grid, distributed on several metal layers, each characterized by different wire cross-sections, connected by vias with varying electrical properties leads to widely varying Laplacian weights. Strongly-varying weights and the interior boundary conditions have the effect that solutions of these linear systems are often very localized.

These characteristics explain the relative success of random walk methods for power grid analysis [2], [3]. These methods avoid the explicit representation of the power grid analysis problem in terms of matrices and vectors and take natural advantage of solution locality. On the other hand, from a computational point of view, random walk algorithms are known to be inefficient and have poor and unreliable convergence properties especially in comparison with powerful preconditioned Krylov-subspace iterative methods and algebraic multigrid techniques[1]. As a consequence, we can expect important gains, from properly adapting the numerically more sophisticated techniques in ways that exploit the locality of the solution and avoid representing numerically irrelevant portions of the problem.

In this paper, we present a flexible conjugate gradient method that is tailored to the solution of the truly large-scale linear systems arising in VLSI power grid analysis. The algorithm allows changing preconditioners and sparsification of the search direction at each iteration. These are the key features to exploit the local nature of the solutions.

The remainder of the paper is organized as follows. In Section II, we briefly describe the problem of DC analysis of power grids and the resulting linear algebra problem. In Section III, we review some basic properties of the classical conjugate gradient (CG) method for solving symmetric positive definite systems of linear equations. In Section IV, we introduce a new flexible variant of CG, referred to as FCG in the sequel, that allows sparsification of the search direction at each iteration. In Section V, we present some theoretical properties of FCG. In Section VI, we discuss some practical details for the use of FCG. In Section VII, we report the results of numerical experiments with FCG. Finally, in Section VIII, we make some concluding remarks.

II. Power Grid DC Analysis

Figure 1 shows one representative node in a power grid. At such a node k, by Kirchhoff's current law, Kirchhoff's voltage law, and Ohm's law, we have the equation

$$\left(\sum_{i=1}^{\deg(k)} g_i \right) V_k - \sum_{i=1}^{\deg(k)} g_i V_i = -I_k \qquad (1)$$

for the unknown voltage V_k at node k and the unknown voltages V_i, $i = 1, 2, \ldots, \deg(k)$, at the nodes i adjacent to k. The equations (1) for all N nodes of the power grids can be written in compact form as a system of linear equations

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}. \qquad (2)$$

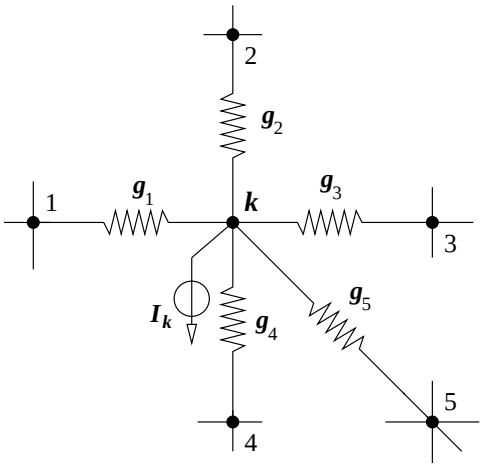

Fig. 1. Representative node in a power grid

Here $\mathbf{A}$ is a real symmetric positive definite $N \times N$ matrix, $\mathbf{b}$ is a real vector of length N, and the entries of the solution vector $\mathbf{x}$ are the unknown voltages at each node of the power grid. Recall that a real symmetric $N \times N$ matrix $\mathbf{A}$ is said to be *positive definite* if

$$\mathbf{y}^{\mathrm{T}} \mathbf{A}\, \mathbf{y} > 0 \quad \text{for all} \quad \mathbf{y} \in \mathbb{R}^N,\ \mathbf{y} \neq \mathbf{0}.$$

When the solution of the system is required as part of a dynamic analysis of the network, when reactive elements are also present, the structure of the problem is maintained in most practical cases.

Recall, however, the structure of the network as described in Section I. Due to the presence of the C4s, most of the current drawn by one device in any given point of the chip is going to be supplied mainly through the nearby C4s and the voltage drop caused by this current will be confined in the immediate neighborhood, and practically negligible outside it. As a consequence, the computation of the voltage drops should be done in ways that avoid constructing the entire matrix $\mathbf{A}$ and the full representation of the voltage solution $\mathbf{x}$, which may require a computational and storage effort orders of magnitude beyond what would be necessary given the structure of the problem.

Alternatively sometimes the problem requires knowing the voltage drop in only one or a few probing points. This problem too can be solved by exploiting the "local" structure of the problem. In this case we seek, e.g., one value selected by the very sparse vector $\mathbf{l}$

$$y = \mathbf{l}^{\mathrm{T}} \mathbf{x} = \mathbf{l}^{\mathrm{T}} (\mathbf{A}^{-1} \mathbf{b}),$$

which can also be solved as

$$y = (\mathbf{A}^{-\mathrm{T}} \mathbf{l})^{\mathrm{T}} \mathbf{b},$$

In other words the "locality" of the problem can be exploited whenever either the excitation, $\mathbf{b}$, or the probing of the solution, $\mathbf{l}$ exhibit "locality". These "local", efficiently solvable problems can also be used as building blocks in a more comprehensive analysis schemes.

In the sequel, we present a solver algorithm capable of operating with only the relevant portion of the circuit matrix $\mathbf{A}$, while actively maintaining the sparsity of the solution.

III. A Brief Review of CG

The classical conjugate gradient (CG) algorithm [4] is an iterative method for the solution of symmetric positive definite linear systems (2). Next, we briefly review some of the key properties of CG.

The method allows the choice of an arbitrary initial guess $\mathbf{x}_0 \in \mathbb{R}^N$. We denote by $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$ the corresponding initial residual, and by $\rho_0 = \|\mathbf{r}_0\|_2$ the Euclidean norm of $\mathbf{r}_0$. The method is usually combined with preconditioning. A preconditioner for the linear system (2) is a real symmetric positive definite $N \times N$ matrix $\mathbf{M}$ with the following two properties. First, linear systems $\mathbf{M}\,\mathbf{z} = \mathbf{v}$ with coefficient matrix $\mathbf{M}$ are 'easy' to solve, compared to the solution of the original linear system (2). Second, CG applied to the *preconditioned* linear system

$$\left(\mathbf{A}\,\mathbf{M}^{-1}\right) \mathbf{y} = \mathbf{b}, \quad \mathbf{x} = \mathbf{M}^{-1}\,\mathbf{y}, \tag{3}$$

should converge significantly faster than CG applied to the original system (2). Note that the linear systems (2) and (3) are equivalent.

Starting from the initial guess $\mathbf{x}_0$, CG generates a sequence of iterates

$$\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, \ldots$$

that converge to the solution $\overline{\mathbf{x}} := \mathbf{A}^{-1}\,\mathbf{b}$ of the linear system (2). Moreover, at iteration n, the n-th CG iterate $\mathbf{x}_n$ is an *optimal* approximation to $\overline{\mathbf{x}}$ in the following sense. At iteration n, any vector

$$\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_n \tag{4}$$

is a possible choice for the iterate $\mathbf{x}_n$. Here, $\mathcal{K}_n$ denotes the n-dimensional subspace of $\mathbb{R}^N$ spanned by the n *Krylov* vectors

$$\mathbf{M}^{-1}\,\mathbf{r}_0, \left(\mathbf{M}^{-1}\,\mathbf{A}\right) \mathbf{M}^{-1}\,\mathbf{r}_0, \ldots, \left(\mathbf{M}^{-1}\,\mathbf{A}\right)^{n-1} \mathbf{M}^{-1}\,\mathbf{r}_0.$$

We remark that $\mathcal{K}_n$ is called the n-th Krylov subspace induced by the matrix $\mathbf{M}^{-1}\,\mathbf{A}$ and the vector $\mathbf{M}^{-1}\,\mathbf{r}_0$. Among all possible vectors (4), the n-th CG iterate $\mathbf{x}_n$ is the one that minimizes the distance to the solution $\overline{\mathbf{x}}$ in the $\mathbf{A}$-norm

$$\|\mathbf{x} - \overline{\mathbf{x}}\|_{\mathbf{A}} := \left((\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}}\, \mathbf{A}\, (\mathbf{x} - \overline{\mathbf{x}})\right)^{1/2}.$$

More precisely, $\mathbf{x}_n$ is the unique vector of the form (4) that satisfies

$$\|\mathbf{x}_n - \overline{\mathbf{x}}\|_{\mathbf{A}} = \min_{\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_n} \|\mathbf{x} - \overline{\mathbf{x}}\|_{\mathbf{A}}. \tag{5}$$

There are a number of different, but mathematically equivalent implementations of CG. A compact formulation of the first n iterations of a version of CG based on three-term recursions is as follows:

$$\begin{aligned} \mathbf{A}\,\mathbf{Z}_n &= \mathbf{V}_n\,\mathbf{T}_n + t_{n+1,n}\,\mathbf{v}_{n+1}\,\mathbf{e}_n^{\mathrm{T}}, \\ \mathbf{Z}_n &= \mathbf{M}^{-1}\,\mathbf{V}_n. \end{aligned} \tag{6}$$

Here, $\mathbf{Z}_n$ is an $N \times n$ matrix whose columns form a basis of the n-th Krylov subspace $\mathcal{K}_n$, $\mathbf{T}_n$ is an $n \times n$ nonsingular tridiagonal matrix, $t_{n+1,n} \in \mathbb{R}$, and $\mathbf{e}_n$ denotes the n-th unit vector in $\mathbb{R}^n$. Moreover, the columns of the matrices $\mathbf{V}_n$ and $\mathbf{Z}_n$ are constructed such that the following orthogonality relations hold true:

$$\mathbf{Z}_n^{\mathrm{T}} \mathbf{V}_n = \mathbf{Z}_n^{\mathrm{T}} \mathbf{M} \mathbf{Z}_n = \begin{bmatrix} \delta_1 & 0 & \cdots & 0 \\ 0 & \delta_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \delta_n \end{bmatrix}. \tag{7}$$

Using the quantities in (6), the n-th CG iterate $\mathbf{x}_n$ can be characterized as follows:

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Z}_n \, \alpha_n, \quad \text{where} \quad \mathbf{T}_n \, \alpha_n = \rho_0 \, \mathbf{e}_1. \tag{8}$$

Here, $\mathbf{e}_1$ denotes the first unit vector in $\mathbb{R}^n$.

Note that, in view of the second relation in (6), the columns $\mathbf{z}_j$ of the matrix $\mathbf{Z}_n$ and the columns $\mathbf{v}_j$ of the matrix $\mathbf{V}_n$ are connected via a solve with the preconditioner $\mathbf{M}$, i.e.,

$$\mathbf{z}_j = \mathbf{M}^{-1} \, \mathbf{v}_j, \quad j = 1, 2, \ldots, n. \tag{9}$$

IV. A Flexible Variant of CG

In each iteration of standard CG, the preconditioner $\mathbf{M}$ is used to compute the vectors $\mathbf{z}_n$ via the solution of the linear system $\mathbf{M} \, \mathbf{z}_n = \mathbf{v}_n$. Saad [5], with his flexible GMRES (FGMRES) algorithm, was the first to devise a method that allows a changing preconditioner $\mathbf{M}_n$ at each n-th iteration. The basic idea is as follows. Instead of using (9) with a fixed matrix $\mathbf{M}$, at iteration n, the vectors $\mathbf{z}_n$ and $\mathbf{v}_n$ are connected via the relation

$$\mathbf{z}_n = \mathbf{M}_n^{-1} \, \mathbf{v}_n, \tag{10}$$

where $\mathbf{M}_n$ is a nonsingular matrix that is allowed to change in each iteration. Note that FGMRES is an iterative method for the solution of nonsymmetric linear systems. Here, we use an approach similar to FGMRES to develop a flexible variant of CG that allows not only changing preconditioning but also sparsification of the iteration vectors. In fact, sparsification is the key property of our algorithm for efficient power grid analysis. We remark that other flexible variants of the CG method have been proposed [6], [7]. We stress that these variants are different from the flexible CG (FCG) algorithm proposed here. In particular, the algorithms in [6], [7] are not set up to allow sparsification.

Instead of a relation of the form (10), at iteration n, we generate vectors $\mathbf{z}_n$ and $\mathbf{v}_n$ that are connected via

$$\mathbf{z}_n = \mathbf{Q}_n \, \mathbf{v}_n. \tag{11}$$

Here, $\mathbf{Q}_n$ is an $N \times N$ matrix of the form

$$\mathbf{Q}_n = \mathbf{P}_n^{(2)} \, \mathbf{M}_n^{-1} \, \mathbf{P}_n^{(1)}, \tag{12}$$

where $\mathbf{M}_n$, $\mathbf{P}_n^{(1)}$, $\mathbf{P}_n^{(2)}$ are $N \times N$ matrices and $\mathbf{M}_n$ is assumed to be nonsingular. In (12), the matrix $\mathbf{M}_n$ is the preconditioner for the linear system (2) to be solved. Note that $\mathbf{M}_n$ is allowed to change at each iteration. However, we mostly use a fixed preconditioner, i.e.,

$$\mathbf{M}_n = \mathbf{M} \quad \text{for all} \quad n.$$

The purpose of the matrices $\mathbf{P}_n^{(1)}$ and $\mathbf{P}_n^{(2)}$ is to sparsify the vector $\mathbf{z}_n$ by zeroing out small entries of $\mathbf{v}_n$ via the matrix $\mathbf{P}_n^{(1)}$ and small entries of $\mathbf{M}_n^{-1} \, \mathbf{P}_n^{(1)} \, \mathbf{v}_n$ via the matrix $\mathbf{P}_n^{(2)}$. We stress that $\mathbf{P}_n^{(1)}$ and $\mathbf{P}_n^{(2)}$ depend on the size of the entries of $\mathbf{v}_n$, and so in general, these matrices do change at each iteration. Furthermore, we remark that the matrix $\mathbf{Q}_n$ is singular in general, and that $\mathbf{Q}_n$ is not even required to be symmetric.

First, we state the underlying recurrence relations of our FCG algorithm in compact form, similar to the compact form (6) of standard CG. After the first n iterations, FCG has generated the columns of the matrices

$$\mathbf{V}_n = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix} \text{ and } \mathbf{Z}_n = \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \cdots & \mathbf{z}_n \end{bmatrix}.$$

These matrices are connected as follows:

$$\begin{aligned} \mathbf{A} \, \mathbf{Z}_n &= \mathbf{V}_n \, \mathbf{H}_n + h_{n+1,n} \, \mathbf{v}_{n+1} \, \mathbf{e}_n^{\mathrm{T}}, \\ \mathbf{Z}_n &= \begin{bmatrix} \mathbf{Q}_1 \, \mathbf{v}_1 & \mathbf{Q}_2 \, \mathbf{v}_2 & \cdots & \mathbf{Q}_n \, \mathbf{v}_n \end{bmatrix}. \end{aligned} \tag{13}$$

Here, $\mathbf{H}_n$ is an $n \times n$ upper Hessenberg matrix, $h_{n+1,n} \in \mathbb{R}$, and $\mathbf{e}_n$ denotes the n-th unit vector in $\mathbb{R}^n$. Moreover, the columns of the matrices $\mathbf{V}_n$ and $\mathbf{Z}_n$ are constructed such that the following semi-biorthogonality relations hold true:

$$\mathbf{Z}_n^{\mathrm{T}} \mathbf{V}_n = \begin{bmatrix} \delta_1 & 0 & \cdots & 0 \\ \star & \delta_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \star & \cdots & \star & \delta_n \end{bmatrix}. \tag{14}$$

Using the quantities in (13), the n-th FCG iterate $\mathbf{x}_n$ can be characterized as follows:

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Z}_n \, \alpha_n, \quad \text{where} \quad \mathbf{H}_n \, \alpha_n = \rho_0 \, \mathbf{e}_1. \tag{15}$$

Here, $\mathbf{e}_1$ denotes the first unit vector in $\mathbb{R}^n$.

Next, we present the basic steps of an actual implementation of FCG.

Algorithm 1 (Flexible Conjugate Gradient Method)

0) Choose an initial vector $\mathbf{x}_0 \in \mathbb{R}^N$.
   Set $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \, \mathbf{x}_0$, $\rho_0 = \|\mathbf{r}_0\|_2$, and $\mathbf{v}_1 = \mathbf{r}_0 / \rho_0$.
1) For $n = 1, 2, \ldots,$ do:
   - Select $\mathbf{Q}_n = \mathbf{P}_n^{(2)} \, \mathbf{M}_n^{-1} \, \mathbf{P}_n^{(1)}$.
   - Compute $\mathbf{z}_n = \mathbf{Q}_n \, \mathbf{v}_n$.
   - Set $\delta_n = \mathbf{z}_n^{\mathrm{T}} \mathbf{v}_n$.
   - Compute $\mathbf{v} = \mathbf{A} \, \mathbf{z}_n$.
   - For $j = 1, 2, \ldots, n,$ do:
     Set
     $$h_{j,n} = \frac{\mathbf{z}_j^{\mathrm{T}} \mathbf{v}}{\delta_n} \quad \text{and} \quad \mathbf{v} = \mathbf{v} - \mathbf{v}_j \, h_{j,n}. \tag{16}$$
   - Set $h_{n+1,n} = \|\mathbf{v}\|_2$ and $\mathbf{v}_{n+1} = \mathbf{v} / h_{n+1,n}$.
   - Set $\mathbf{Z}_n = \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \cdots & \mathbf{z}_n \end{bmatrix}$.

- Set $\mathbf{H}_n = \big[h_{j,k}\big]_{j,k=1,2,\ldots,n}$.

2) Compute the solution α_n of the $n \times n$ linear system

$$\mathbf{H}_n\, \alpha_n = \rho_0\, \mathbf{e}_1 \tag{17}$$

and set $\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Z}_n\, \alpha_n$.

V. Some Theoretical Properties of FCG

In this section, we present some properties of the FCG method.

Let

$$\mathcal{S}_n = \operatorname{span}\{\, \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n \,\}. \tag{18}$$

denote the subspace of $\mathbb{R}^N$ spanned by the columns of the matrix $\mathbf{Z}_n$. In view of (15), the n-th FCG iterate $\mathbf{x}_n$ is is of the form

$$\mathbf{x}_n \in \mathbf{x}_0 + \mathcal{S}_n. \tag{19}$$

Then, using (13) and the semi-biorthogonality relations (14), it is easy to verify that among all possible iterates $\mathbf{x} \in \mathbf{x}_0 + \mathcal{S}_n$, the n-th FCG iterate $\mathbf{x}_n$ is the one that minimizes the distance to the solution $\overline{\mathbf{x}}$ in the $\mathbf{A}$-norm. More precisely, $\mathbf{x}_n$ is the unique vector of the form (19) that satisfies

$$\|\mathbf{x}_n - \overline{\mathbf{x}}\|_{\mathbf{A}} = \min_{\mathbf{x} \in \mathbf{x}_0 + \mathcal{S}_n} \|\mathbf{x} - \overline{\mathbf{x}}\|_{\mathbf{A}}. \tag{20}$$

Note that (20) is an extension of the optimality property (5) of standard CG. However, we stress that, in general, the subspace $\mathcal{S}_n$ in (20) is no longer a Krylov subspace.

In Algorithm 1 there are two operations that potentially could result in a breakdown, namely division by $\delta_n = 0$ in (16) and the solution of the linear system (17) in the case $\mathbf{H}_n$ is singular. Next, we show that such breakdowns cannot occur provided some weak assumptions are satisfied.

To ensure that $\delta_n > 0$ for all n, we choose the 'right' and 'left' sparsification matrices $\mathbf{P}_n^{(1)}$ and $\mathbf{P}_n^{(2)}$ in (12) such that

$$\mathbf{P}_n^{(1)} = \mathbf{P}_n \quad \text{and} \quad \mathbf{P}_n^{(2)} = \mathbf{P}_n^{\mathrm{T}}.$$

Hence, the matrix $\mathbf{Q}_n$ in (12) is of the form

$$\mathbf{Q}_n = \mathbf{P}_n^{\mathrm{T}}\, \mathbf{M}_n^{-1}\, \mathbf{P}_n. \tag{21}$$

Moreover, we assume that the preconditioner $\mathbf{M}_n$ in (21) is indeed symmetric positive definite. Then, provided that the sparsification matrix $\mathbf{P}_n$ does not zero out all entries of the vector $\mathbf{v}_n$, i.e.,

$$\mathbf{y}_n := \mathbf{P}_n\, \mathbf{v}_n \neq \mathbf{0}, \tag{22}$$

we have

$$\delta_n = \mathbf{z}_n^{\mathrm{T}} \mathbf{v}_n = \mathbf{v}_n^{\mathrm{T}} \mathbf{P}_n^{\mathrm{T}}\, \mathbf{M}_n^{-1}\, \mathbf{P}_n\, \mathbf{v}_n = \mathbf{y}_n^{\mathrm{T}}\, \mathbf{M}_n^{-1}\, \mathbf{y}_n > 0. \tag{23}$$

Thus division by $\delta_n = 0$ cannot occur. Furthermore, in view of (14), it also follows that the matrix

$$\mathbf{Z}_n^{\mathrm{T}}\, \mathbf{V}_n \quad \text{is nonsingular.} \tag{24}$$

In order to guarantee that the matrix $\mathbf{H}_n$ in (17) is nonsingular, we only need the standard assumption (see [5]) that the matrix $\mathbf{Z}_n$ has full column rank n. Indeed, by multiplying the first relation (13) from the left by $\mathbf{Z}_n^{\mathrm{T}}$ and by using that $\mathbf{Z}_n^{\mathrm{T}} \mathbf{v}_{n+1} = \mathbf{0}$, it follows that

$$\mathbf{Z}_n^{\mathrm{T}} \mathbf{A}\, \mathbf{Z}_n = \big(\mathbf{Z}_n^{\mathrm{T}}\, \mathbf{V}_n\big)\, \mathbf{H}_n. \tag{25}$$

Since $\mathbf{A}$ is symmetric positive definite and $\mathbf{Z}_n$ has full column rank, the matrix on the left-hand side of (25) is nonsingular. Therefore, the matrix $\mathbf{H}_n$ cannot be singular.

VI. Some Practical Details of FCG

In this section, we present some practical details for the use of FCG.

A. Sparsification

Recall from (12) that the purpose of the matrices $\mathbf{P}_n^{(1)}$ and $\mathbf{P}_n^{(2)}$ is to sparsify the iteration vectors, i.e., to zero out the small entries of these vectors. For example, the simplest choice of the matrices $\mathbf{P}_n^{(1)}$ and $\mathbf{P}_n^{(2)}$ is diagonal matrices with zeros and ones on the diagonal. Of course, in this case, we do not explicitly set up the matrices $\mathbf{P}_n^{(1)}$ or $\mathbf{P}_n^{(2)}$. Instead, we obtain the sparsified vector

$$\mathbf{y}_n = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \mathbf{P}_n^{(1)}\, \mathbf{v}_n, \quad \text{where} \quad \mathbf{v}_n = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix},$$

by setting

$$y_j = \begin{cases} v_j & \text{if } |v_j| \geq \texttt{sptol1} \times \|\mathbf{v}_n\|_2, \\ 0 & \text{if } |v_j| < \texttt{sptol1} \times \|\mathbf{v}_n\|_2, \end{cases} \tag{26}$$

for all $j = 1, 2, \ldots, N$. Here, $0 \leq \texttt{sptol1} < 1$ is some tolerance for the 'right' sparsification due to $\mathbf{P}_n^{(1)}$. The 'left' sparsification due to $\mathbf{P}_n^{(2)}$ is done analogous to (26), but with a tolerance $0 \leq \texttt{sptol2} < 1$. We stress that the tolerances `sptol1` and `sptol2` for the right and left sparsification can be (and in practice are) chosen to be different. Also, remark that the choice $\texttt{sptol1} = 0$ or $\texttt{sptol2} = 0$ corresponds to no right or no left sparsification, respectively.

B. Preconditioning

Recall from (12) that the matrix $\mathbf{M}_n$ is the actual preconditioner for the linear system (2). In all our numerical tests, we used a fixed preconditioner $\mathbf{M} = \mathbf{M}_n$ for all n.

Next, we describe two simple preconditioning techniques. Let

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{L}^{\mathrm{T}} \tag{27}$$

be the additive splitting of the coefficient matrix $\mathbf{A}$ of (2) into its strictly lower triangular part $\mathbf{L}$, its diagonal part $\mathbf{D}$, and its strictly upper triangular part $\mathbf{L}^{\mathrm{T}}$.

For *diagonal preconditioning*, one chooses

$$\mathbf{M} = \mathbf{D}.$$

Since $\mathbf{A}$ is symmetric positive definite, all the diagonal entries of $\mathbf{A}$ are positive. Thus the diagonal preconditioner $\mathbf{M} = \mathbf{D}$ is guaranteed to be symmetric positive definite and, in particular,

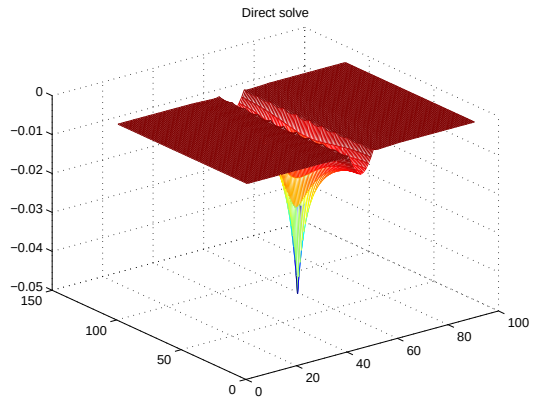

Fig. 2. Example 1, direct solver

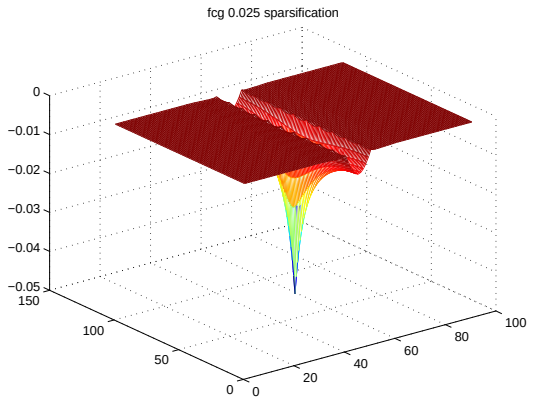

Fig. 3. Example 1, FCG with 0.025 sparsification

nonsingular. For diagonal preconditioning, left sparsification becomes unnecessary, and one sets $\texttt{sptol2} = 0$ in this case.

For *SSOR preconditioning*, one chooses

$$\mathbf{M} = (\mathbf{D} + \mathbf{L})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{L}^{\mathrm{T}}).$$

Again, $\mathbf{M}$ is guaranteed to be symmetric positive definite and nonsingular. Note that the factors $(\mathbf{D} + \mathbf{L})$ and $(\mathbf{D} + \mathbf{L}^{\mathrm{T}})$ are both triangular and that $\mathbf{D}^{-1}$ is diagonal. Therefore, each solve with the SSOR preconditioner $\mathbf{M}$ requires only two triangular solves and one multiplication with the diagonal matrix $\mathbf{D}$. In practice, SSOR preconditioning is more effective when it is combined with an appropriate reordering technique. This means that, instead of (27), one uses a corresponding splitting for the reordered matrix $\mathbf{\Pi}^{\mathrm{T}}\mathbf{A}\,\mathbf{\Pi}$, where Π is a permutation matrix produced by a reordering routine. The numerical results with SSOR preconditioning reported in Section VII were obtained with symmetric reverse Cuthill-McKee reordering. For SSOR preconditioning, we only use left sparsification, and thus $\texttt{sptol1} = 0$ in this case.

VII. Numerical Examples

In this section, we report some results of numerical experiments for three examples. Example 1 is a single-layer anisotropic power grid with $N = 9081$ nodes. It was run with digaonal preconditioning. In Figure 2, we show the nodal voltages obtained by solving the linear system $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ with a direct method and the anisotropic nature of the network is apparent from the solution.

Figure 3 depicts the solution of the same linear system with FCG and a very aggressive sparsification factor $\texttt{sptol1} = 0.05$. The two solutions are indistinguishable and the sparsification led to a reasonable increase in the number of iterations to convergence. The computational saving in manipulating sparse vectors is significant.

Figure 4 shows the sparsity of the FCG iteration vectors vs. the iteration number n for different choices of the sparsification factor $\texttt{sptol1}$. The plot also shows the total number of iterations to convergence.

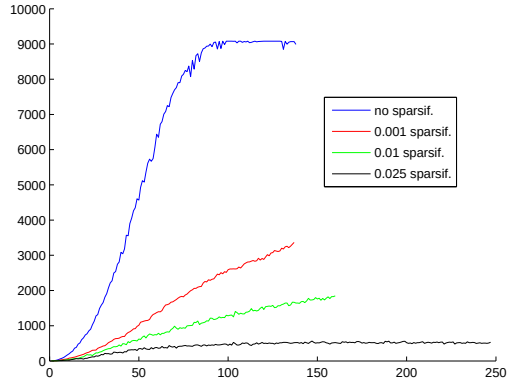

Fig. 4. Example 1, sparsity of vectors vs. iterations

The next example exploits the sparsity on the observability side. Here we analyze a 6 layer power grid with 30000 nodes shown in Figure 5. We compute the voltage drop at a particular point marked with X on the figure.

The result is shown in Figure 6 and again we observe significant sparsification at the expense of modest increase in the number of iterations.

Finally, the last Examples applies FCG to a regular DC analysis of a 6 layer powergrid with about 30000 nodes with currents drawn in numerous locations, in other words, non-local excitation, and seeking the complete solution. Even in this situation our algorithm detects and exploits a significant amount of sparsity resulting in great computational and memory savings. Figure 6 shows

VIII. Concluding Remarks

The paper has introduced a novel flexible conjugate gradient algorithm as a tool for the solution of the extremely large electrical networks that arise in the analysis of VLSI power distribution networks. The main feature of the algorithm is its ability to change and adapt its preconditioner at every iteration. This feature is used successfully to sparsify and eliminate numerically irrelevant portions of the basis vectors and thus

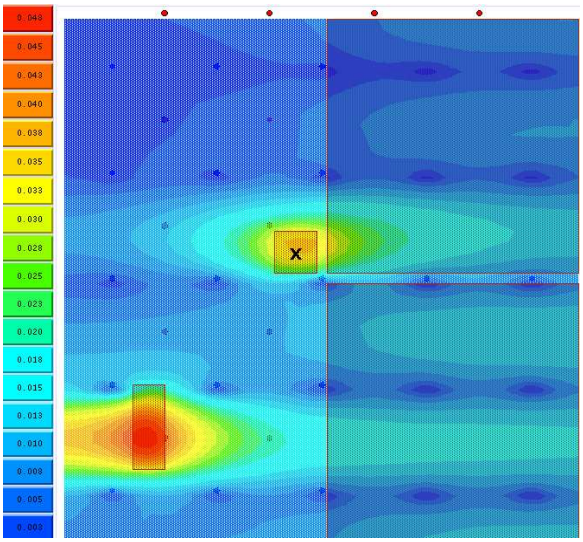

Fig. 5. Example 2, volt5

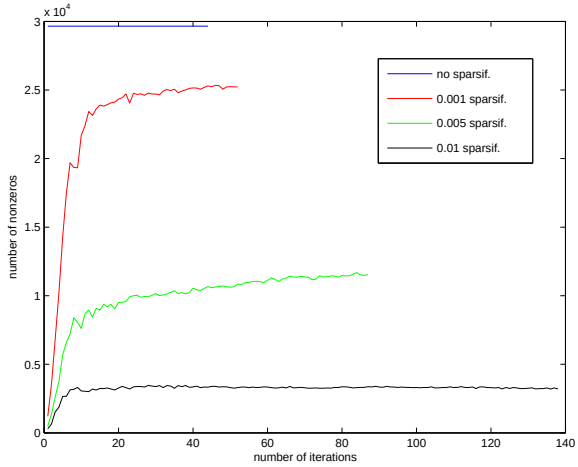

Fig. 6. Example 2, sparsity of vectors vs. iterations

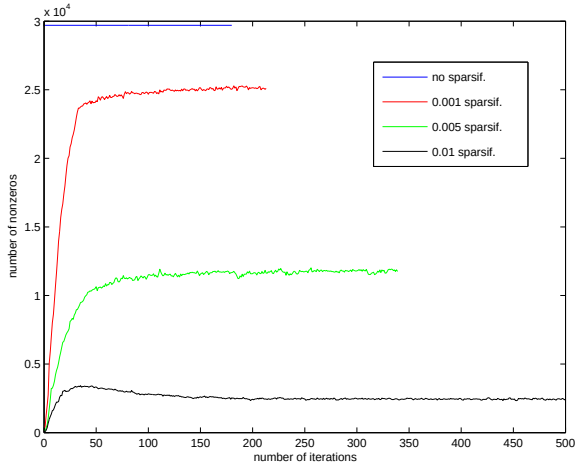

Fig. 7. Example 3, sparsity of vectors vs. iterations

exploit for great computational savings the natural sparsity of the solution. As a consequence, the extremely large cost of modeling distant and irrelevant portions of the network can also be avoided. The examples demonstrate the robustness of the algorithm even in the presence of very aggressive sparsification.

REFERENCES

[1] J. Kozhaya, S. Nassif, and F. Najm, "A multigrid-like technique for power grid analysis," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 10, pp. 1148–1160, 2002.

[2] H. Qian, S. Nassif, and S. Sapatnekar, "Power grid analysis using random walks," *IEEE Trans. Computer-Aided Design*, vol. 24, no. 8, pp. 1204–1224, 2005.

[3] H. Qian and S. S. Sapatnekar, "A hybrid linear equation solver and its application in quadratic placement," in *Tech. Dig. 2005 IEEE/ACM International Conference on Computer-Aided Design*. Los Alamitos, California: IEEE Computer Society Press, 2005, pp. 904–908.

[4] M. R. Hestenes and E. L. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Standards*, vol. 49, pp. 409–436, 1952.

[5] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," *SIAM J. Sci. Comput.*, vol. 14, pp. 461–469, 1993.

[6] G. H. Golub and Q. Ye, "Inexact preconditioned conjugate gradient method with inner-outer iteration," *SIAM J. Sci. Comput.*, vol. 21, pp. 1305–1320, 1999.

[7] Y. Notay, "Flexible conjugate gradients," *SIAM J. Sci. Comput.*, vol. 22, pp. 1444–1460, 2000.