A Multilevel $\mathcal{H}$-matrix-based Approximate Matrix Inversion Algorithm for Vectorless Power Grid Verification[*]

Wei Zhao, Yici Cai, and Jianlei Yang
Tsinghua National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, Beijing, P. R. China
zhao-w10@mails.tsinghua.edu.cn

Abstract - Vectorless power grid verification technique makes it possible to estimate the worst-case voltage fluctuations of the on-chip power delivery network at the early design stage. For most of the existing vectorless verification algorithms, the subproblem of linear system solution which computes the inverse of the power grid matrix takes up a large part of the computation time and has become a critical bottleneck of the whole algorithm. In this paper, we propose a new algorithm that combines the $\mathcal{H}$-matrix-based technique and the multilevel method to construct a data-sparse approximate inverse of the power grid matrix. Experimental results have shown that the proposed algorithm can obtain an almost linear complexity both in runtime and memory consumption for efficient vectorless power grid verification.

I INTRODUCTION

As supply voltages of integrated circuits are lowered to reduce the power consumption while subthreshold voltages are decreased for better performance, the impact of voltage fluctuations is becoming increasingly significant. A robust power grid must provide sufficient voltage at each gate in order to guarantee the proper logic functionality at the intended design speed. So power grid verification is becoming a critical procedure to guarantee a functional and robust chip design. However, the increasing complexity and decreasing feature size of modern integrated circuits has made power grid verification a challenging task.

Most of the existing power grid verification techniques use DC or transient simulation methods to simulate the power grid and evaluate the power supply noises. But the simulation-based technique has two major drawbacks. First, it is usually too expensive to enumerate all possible waveform combinations or to determine the worst-case waveform pattern. Second, the early stage verification cannot be performed since the detailed current waveform information is still unknown.

To address the requirement of early stage power grid verification, a class of vectorless verification algorithms has been proposed in [1], and studied in [2–8]. These algorithms use the notion of current constraints to specify a feasible space in which currents can vary during circuit operation, and estimate the worst-case voltage fluctuations by solving linear programming problems under these current constraints. The power grid model used in these verification algorithms has been extended from the RC model in [1] to the RLC model in [2, 7]. And the current constraints have also been extended from DC constraints to transient constraints [8]. Most of the vectorless verification algorithms consist of two major sub-problems: the linear system solution and the linear programming solution. Sparse approximate inverse methods (SPAI [3] and AINV [6]) and a hierarchical matrix inversion algorithm [5] are proposed to speed up the linear system solution. A convex dual algorithm [4] and a network simplex method [6] are proposed to speed up the linear programming solution.

The sub-problem of linear system solution in vectorless power grid verification aims to find out the functions that relate the voltage fluctuations and current excitations at each node of the power grid. We need to compute the whole inverse of the power grid matrix, or equivalently, to solve the corresponding linear equations at each node. The computation is quite expensive due to the extremely large size of the power grid and usually takes up a large part of the whole computation cost.

In this paper, we propose a new approximate matrix inversion algorithm for vectorless power grid verification. Major contributions of the paper are as follows.

1. Different from the sparse approximate inverse methods such as SPAI and AINV used in [3] and [6] respectively, we use the $\mathcal{H}$-matrix-based technique to construct a “data sparse” approximate inverse of the power grid matrix. Both of the theoretical analysis and experimental results have shown that the $\mathcal{H}$-matrix-based approximate inverse method can achieve a higher degree of approximate accuracy with a certain amount of memory footprint.
2. We propose an improved multilevel matrix inversion scheme which can be combined with the $\mathcal{H}$-matrix-based technique to further accelerate the computation speed and reduce the memory footprint.
3. Since $\mathcal{H}$-matrix is constructed based on a hierarchical block structure instead of the row-by-row manner in SPAI, the approximate accuracy of each row may have some fluctuations. In order to ensure the robustness of the $\mathcal{H}$-matrix-based approximate inverse method, we introduce an iterative refinement scheme to control the approximate accuracy by a user-specified parameter according to the actual accuracy requirement.

The reminder of this paper is organized as follows. The problem formulation and previous approaches are summarized in Section II. The details of the proposed algorithm are presented in Section III. Experimental results are given in Section IV. Concluding remarks are given in Section V.

II. BACKGROUND

A. *Problem Formulation*

[*]This work is supported by National Natural Science Foundation of China (NSFC) No.61274031.

A power grid consists of several metal layers, with each layer containing either horizontal or vertical wires. In a realistic design, some wires may be missing or truncated, and the periodicity and density of the wires may also vary.

An RC model of the power grid for vectorless power grid verification is introduced in [1]. In this model, each branch of the power grid is represented by a resistor and there is a capacitor from every grid node to ground. Let $\mathbf{v}(t)$ be the vector of voltage drops, then the system equation can be written as:

$$G\mathbf{v}(t) + C\mathbf{v}'(t) = \mathbf{i}(t) \tag{1}$$

where G is an $n \times n$ conductance matrix and C is an $n \times n$ diagonal matrix of node capacitances.

A more complicated RLC model is introduced in [2, 7], in which each branch is represented either by a resistor, referred to as an r-branch, or by a resistor in series with an inductor, referred to as an rl-branch. The presence of inductances can cause the voltage on a given node of the power grid to fluctuate in both directions, either increasing or decreasing. The system equations can be written as:

$$G\mathbf{v}(t) + C\mathbf{v}'(t) - M\mathbf{i}(t) = \mathbf{i}_{\mathbf{S}}(t) \tag{2}$$

and

$$M^T\mathbf{v}(t) + L\mathbf{i}'(t) = 0 \tag{3}$$

where G is an $n \times n$ conductance matrix, C is an $n \times n$ diagonal matrix of node capacitances, M is an $n \times m$ incidence matrix whose elements are either ±1 or 0, and L is an $m \times m$ diagonal matrix of inductance values.

There are two kinds of current constraints used in most of the existing vectorless power grid verification algorithms: local constraints

$$0 \leq \mathbf{i}(t) \leq I_L \; \forall t \geq 0 \tag{4}$$

and global constraints

$$U\mathbf{i}(t) \leq I_G \tag{5}$$

where U is an $m \times n$ matrix which contains only 0s and 1s. Local constraints define upper bounds on individual current sources and global constraints define upper bounds for certain groups of current sources. [6] uses additional equality constraints for the special P/G grid model in which the ground grid is not symmetric to the power grid. These constraints give DC upper bounds on transient waveforms. If the upper bounds are also defined by a set of transient waveforms, the results of vectorless power grid verification may be more realistic while the problem will be more difficult to solve. [8] uses the hierarchical current and power constraints to represent the transient constraints approximately.

B. *Previous Methods*

Vectorless power grid verification aims to verify the safety of the power grid by estimating the worst-case voltage fluctuations under given current constraints. The current constraints define the feasible space of current excitations and the system equations describe the functional relationship between voltage fluctuations and current excitations implicitly. But it still needs a lot of work to convert the original problem to a set of linear programming problems. In simple terms, the functional relationship between voltage fluctuations and current excitations at each node of the power grid needs to be converted to an explicit form. Details of the theoretical derivations can be found in [3] and [7]. Finally, most of the existing vectorless power grid verification algorithms boil down to two major sub-problems: the linear system solution which finds out the function that relates the voltage fluctuations and the current excitations by computing the inverse of the power grid matrix, and the linear programming solution which computes the worst-case voltage fluctuations at each node of the power grid subject to the current constraints.

Since the linear system solution problem needs not just one, but n linear equation solutions, the time complexity is quite large. If we use preconditioned conjugate gradient (PCG) methods to compute the corresponding row of the inverse matrix for each node, the whole linear system solution may take up about 80% of the computation time [4].

The linear system solution in vectorless power grid verification has some special properties compared with the similar problem in power grid simulation. First, it is a multiple right-hand sides problem. Second, the calculation accuracy needs to be user-specified and is usually lower than the required accuracy in DC or transient simulation. For the multiple right-hand sides problem, the Cholesky factorization based direct solver is a good choice since the resulting factor matrix is reusable. But the memory usage of the direct solver is usually too large due to the fill-in when computing the factor matrix. The iterative solvers can take advantage of the relatively low accuracy requirements. But the runtime is still quite long when dealing with the multiple right-hand sides problem. So the sparse approximate inverse based method [3, 6] which can take advantage of the two properties at the same time seems to be a more promising choice.

C. *Sparse Approximate Inverse Methods*

Sparse approximate inverse methods such as SPAI and AINV have been widely studied as a class of preconditioning techniques [10]. These methods try to approximate the inverse A^{-1} of a sparse matrix A using a sparse matrix M. The basis for sparse approximate inverse approaches is the assumption that although the inverse of a sparse matrix is generally dense, the majority of elements in the inverse are small enough to be neglected. A well-known result that supports this assumption is that if a banded matrix A is positive definite, the following bound can be established: $|b_{ij}| \leq C\gamma^{|i-j|}$, where $\gamma < 1$, $C > 0$ and b_{ij} is the element of A^{-1} at the (i,j)-th location [11]. This result indicates that values of the elements in the inverse matrix will decay exponentially away from the diagonal.

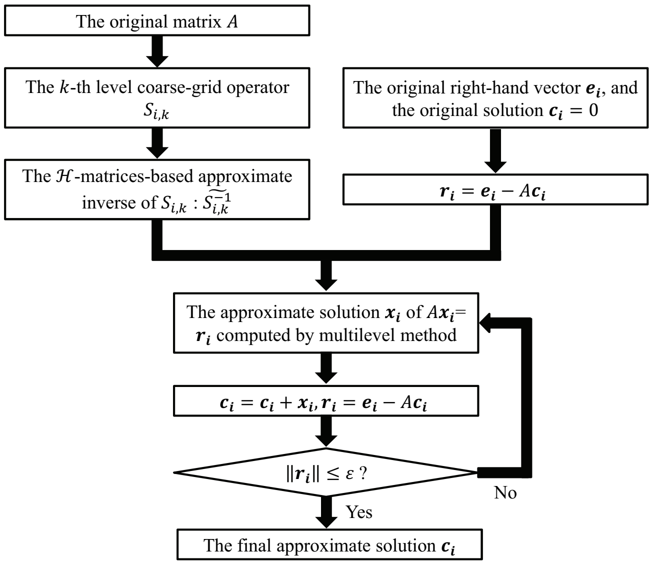
However, there are still some limitations of these sparse approximate inverse methods. For example, a very large constant C in the estimate may lead to unacceptable slow decay in practice [10], and as the problem size increases, the approximate accuracy of the sparse approximations with a certain sparsity pattern will always decline.

The power grid analysis problem shows a global coupling property. But the sparse approximate inverse can only contain some local information since there are only a fixed number of nonzero elements in each row of the approximate inverse. The fact that AINV has a better approximate performance than SPAI [6] can be explained as it is a factorization based method which can store more nonzero elements implicitly. So if we want to get a better sparse approximation, we have to find a method which can bring in more global information with a certain amount of memory footprint.

III. PROPOSED APPROACH

A. Algorithm Overview

Based on the analysis above, we propose a new approximate inverse algorithm which combines the $\mathcal{H}$-matrix-based technique and the multilevel method. Both of the two techniques can bring in more global information and help us to get a better approximation to the inverse matrix. Fig.1 shows the overall approximate inverse computation flow. The algorithm is under a multilevel framework on the whole, and the $\mathcal{H}$-matrix-based technique is used to compute the approximate inverse of the coarse-grid operator. In order to ensure the robustness of the algorithm, we also introduce an iterative refinement scheme to control the approximate accuracy.



The original matrix A
The k-th level coarse-grid operator $S_{i,k}$
The original right-hand vector $\boldsymbol{e_i}$, and the original solution $\boldsymbol{c_i} = 0$
The $\mathcal{H}$-matrices-based approximate inverse of $S_{i,k}$: $\widetilde{S_{i,k}^{-1}}$
$\boldsymbol{r_i} = \boldsymbol{e_i} - A\boldsymbol{c_i}$
The approximate solution $\boldsymbol{x_i}$ of $A\boldsymbol{x_i}= \boldsymbol{r_i}$ computed by multilevel method
$\boldsymbol{c_i} = \boldsymbol{c_i} + \boldsymbol{x_i}, \boldsymbol{r_i} = \boldsymbol{e_i} - A\boldsymbol{c_i}$
$\|\boldsymbol{r_i}\| \le \varepsilon$?
No
Yes
The final approximate solution $\boldsymbol{c_i}$

Fig.1. The approximate inverse computation flow.

B. Hierarchical Matrices

Hierarchical matrices, or $\mathcal{H}$-matrices in short, provide a data-sparse way to approximate fully populated matrices [14, 15]. Given an $\mathcal{H}$-matrix, the standard matrix operations such as matrix-vector multiplication, matrix-matrix addition and multiplication as well as approximate matrix inversion can all be defined for this $\mathcal{H}$-matrix format. The application of $\mathcal{H}$-matrix for power grid simulation has been studied in [9].

An $\mathcal{H}$-matrix approximation to a given dense matrix is based on a certain hierarchical block structure of the matrix in which some off-diagonal blocks are represented by low-rank approximations. Let $M \in \mathbb{R}^{n\times m}$ with $rank(M) = k$. Then there exist matrices $B \in \mathbb{R}^{n\times k}$ and $C \in \mathbb{R}^{m\times k}$ such that $M = BC^T$. The storage required to store the matrix M is now $k(n + m)$ instead of nm, which is significantly smaller if $k \ll min\{n, m\}$. If M is not of rank k but can be approximated by a matrix $\widetilde{M}$ which satisfies $rank(\widetilde{M}) = k$ and $\|M - \widetilde{M}\| \le \varepsilon$ ($\varepsilon \ge 0$ is small enough), we can still replace M by the low-rank approximation.

The blocks which allow for such low rank representations are selected from the hierarchy of partitions organized in the so-called cluster tree. And the accuracy of an $\mathcal{H}$-matrix approximation depends on how well the individual blocks in the partition can be approximated by low rank matrices. To obtain a suitable block partition, we construct a hierarchy of partitions from which the coarsest one that satisfies a certain admissibility condition which shall ensure the approximability by a low rank matrix is chosen.

There are two ways to construct the cluster tree: geometric clustering and algebraic clustering [9]. Geometric clustering methods need the detailed geometric information of the problem. These methods are based on the observation that for many differential and integral operators, if the parts of the geometry associated with the index sets I and J are well separated, then the matrix block $M \in \mathbb{R}^{I\times J}$ can be approximated by a low-rank matrix. They are usually used for solving problems arising from the finite element method (FEM) or the boundary element method (BEM). But since the power grid may have an irregular structure, geometric clustering methods are not applicable. Algebraic clustering methods are performed purely algebraically by using the connectivity relation of the indices stored in the matrix itself. So they are more suitable for the approximate representation of the matrix induced by the power grid.

Given the sparse matrix A arising from the vectorless power grid verification problem, we want to compute the approximate inverse matrix $\widetilde{A^{-1}}$ using the $\mathcal{H}$-matrix-based technique. But since we only have the original matrix A at this stage, we will convert A into the $\mathcal{H}$-matrix format firstly. And then we can use the corresponding $\mathcal{H}$-matrix arithmetic to compute the $\mathcal{H}$-matrix-based inverse directly or to perform the $\mathcal{H}$-Cholesky factorization and using the resulting factor matrix to represent the approximate inverse implicitly.

In terms of complexity, most of the standard $\mathcal{H}$-matrix operations can be performed in almost optimal complexity, i.e., $O(n\log^{\alpha}n)$ with moderate parameter α [14–16]. And the space complexity of $\mathcal{H}$-matrix is also nearly optimal (about $O(n\log n)$). The time complexity of the $\mathcal{H}$-matrix inversion and the $\mathcal{H}$-Cholesky factorization are both $O(n\log^2 n)$, but the $\mathcal{H}$-Cholesky factorization can be computed significantly faster than the direct $\mathcal{H}$-matrix inversion due to the relatively small constants involved in the complexity estimation. So the factorization based method is usually a better choice for the approximate inverse algorithm.

C. Multilevel Methods

Although most of the $\mathcal{H}$-matrix arithmetic can be performed in nearly optimal complexity, from the experimental results in [9] we can see that the constants in these complexity estimates are still a bit large. In this section, we will introduce the multilevel matrix inversion scheme that can be combined with the $\mathcal{H}$-matrix-based method to get further improvement in both of run time and space requirements.

The basic idea of multilevel matrix inversion methods comes from the block preconditioning technique [12]. We can reorder the nodes of the power grid in a red-black manner, so the original matrix can be transformed into a 2×2 block form:

$$A = \begin{bmatrix} D_1 & B \\ B^T & D_2 \end{bmatrix} \tag{6}$$

in which the submatrices D_1 and D_2 are both diagonal matrices. The block LU factorization of A is:

$$A = \begin{bmatrix} D_1 & B \\ B^T & D_2 \end{bmatrix} = \begin{bmatrix} D_1 & 0 \\ B^T & S \end{bmatrix} \begin{bmatrix} I & D_1^{-1}B \\ 0 & I \end{bmatrix} \tag{7}$$

in which $S = D_2 - B^T D_1^{-1} B$ is the Schur complement. The linear equation

$$\begin{bmatrix} D_1 & B \\ B^T & D_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{8}$$

can be rewritten as:

$$\begin{bmatrix} D_1 & 0 \\ B^T & S \end{bmatrix} \begin{bmatrix} I & D_1^{-1}B \\ 0 & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{9}$$

and solved by the block forward and backward substitution:

Algorithm 1. Block Matrix Inversion (BMI)

```
1. $x_1 := D_1^{-1} b_1$
2. $x_2 := S^{-1}(b_2 - B^T x_1)$
3. $x_1 := x_1 - D_1^{-1} B x_2$
```

The Schur complement S is still a sparse matrix although it usually will be denser than the original matrix A. If we just want an approximate solution, we can use the $\mathcal{H}$-matrix-based technique to approximate the inverse of S:

$$M_s^{-1} \cong S^{-1} \tag{10}$$

and get an implicit expression of the approximate inverse:

Algorithm 2. Approximate Block Matrix Inversion (ABMI)

```
1. $x_1 := D_1^{-1} b_1$
2. Compute $M_s^{-1} \cong S^{-1}$, $x_2 := M_s^{-1}(b_2 - B^T x_1)$
3. $x_1 := x_1 - D_1^{-1} B x_2$
```

The block matrix inversion algorithm can be viewed as a 2-level scheme. The multilevel matrix inversion methods are based on some interesting similarities between the block matrix inversion algorithm and the algebraic multigrid methods. Let

$$A^h = A \tag{11}$$

be the fine-grid operator. The coarse grid selected by the classical algebraic multigrid method is just the red-black coarsening if every connection in A is viewed as a strong dependence. The corresponding restriction operator and prolongation operator are

$$I_h^{2h} = \begin{bmatrix} -B^T D_1^{-1} & I \end{bmatrix} \tag{12}$$

and

$$I_{2h}^h = \begin{bmatrix} -D_1^{-1}B \\ I \end{bmatrix} \tag{13}$$

Then we can see that the coarse-grid operator

$$\begin{aligned} A^{2h} = I_h^{2h} A^h I_{2h}^h &= \begin{bmatrix} -B^T D_1^{-1} & I \end{bmatrix} \begin{bmatrix} D_1 & B \\ B^T & D_2 \end{bmatrix} \begin{bmatrix} -D_1^{-1}B \\ I \end{bmatrix} \\ &= D_2 - B^T D_1^{-1} B = S \end{aligned} \tag{14}$$

is just the Schur complement. So we can rewrite the block matrix inversion algorithm using the expansion

$$\begin{aligned} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} D_1^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} -D_1^{-1}B \\ I \end{bmatrix} S^{-1} \begin{bmatrix} -B^T D_1^{-1} & I \end{bmatrix} * \\ \left(\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} - \begin{bmatrix} D_1 & B \\ B^T & D_2 \end{bmatrix} \begin{bmatrix} D_1^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right) \end{aligned} \tag{15}$$

The block matrix inversion algorithm can be viewed as another expression of the *coarse-grid correction* scheme: The vector

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} D_1^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{16}$$

is the original approximate solution on the fine grid, and the residual is

$$r = b - Au = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} - \begin{bmatrix} D_1 & B \\ B^T & D_2 \end{bmatrix} \begin{bmatrix} D_1^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{17}$$

Then the residual equation is solved on the coarse grid, and the result is used to correct the original solution.

In the multilevel approximate matrix inversion scheme, the second step in algorithm 2 is replaced by a recursive solution:

Algorithm 3. Multilevel Approximate Matrix Inversion (MAMI)

```
1. $x_1 := D_1^{-1} b_1$
2. If $k = Level_{max}$
3.     Compute $M_s^{-1} \cong S^{-1}$
           $x_2 := M_s^{-1}(b_2 - B^T x_1)$
4. Else
5.     MAMI$(S,\ x_2,\ k+1)$
6. End If
7. $x_1 := x_1 - D_1^{-1} B x_2$
```

When $k > 2$，the k-th level coarse-grid operator S computed by the algebraic multigrid method is an approximation to the corresponding Schur complement.

The multilevel method described above has been studied in [13]. But we can extend the block matrix inversion algorithm to multilevel approach in a different way. The block matrix inversion algorithm can also be expressed in the following form:

$$\begin{aligned} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} D_1^{-1} & 0 \\ -D_2^{-1} B^T D_1^{-1} & D_2^{-1} \end{bmatrix} \begin{bmatrix} 0 & -B \\ 0 & 0 \end{bmatrix} * \\ \begin{bmatrix} -D_1^{-1}B \\ I \end{bmatrix} S^{-1} \begin{bmatrix} -B^T D_1^{-1} & I \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \\ \begin{bmatrix} D_1^{-1} & 0 \\ -D_2^{-1} B^T D_1^{-1} & D_2^{-1} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \end{aligned} \tag{18}$$

This can be interpreted as a *nested iteration* scheme. We first solve the problem on the coarse grid, and the solution is

$$x^{2h} = S^{-1} \begin{bmatrix} -B^T D_1^{-1} & I \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{19}$$

Then we use the coarse-grid result to obtain a better initial guess on the fine grid:

$$x_0^h = I_{2h}^h x^{2h} = \begin{bmatrix} -D_1^{-1}B \\ I \end{bmatrix} S^{-1} \begin{bmatrix} -B^T D_1^{-1} & I \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{20}$$

and relax one time using the Gauss–Seidel method. The corresponding algorithm can be described as:

Algorithm 4. Modified Block Matrix Inversion (MBMI)

```
1. $x_1 := D_1^{-1} b_1$
2. $u := b_2 - B^T x_1$
3. Solve $S x_2 = u$
4. $x_1 := x_1 - D_1^{-1} B x_2$
5. $x_2 := D_2^{-1}(B^T D_1^{-1} B x_2 + u)$
```

And then the 2-level scheme can be extended to the multi-level version by a recursive solution of the third step just like algorithm 3. In this 2-level scheme, if we solve the equation $Sx_2 = u$ accurately, algorithm 4 is essentially equivalent to algorithm 1. However, if this equation is solved by using an approximate inverse of S like algorithm 2, experimental results show that the new algorithm may be more accurate.

In fact, the algorithms above are not really related to the multigrid idea in the sense that these algorithms are not based on the fundamental multigrid principles of smoothing and coarse-level correction. So if the number of levels we used is too large, the approximation error of the Schur complement in each level will cause the decline in accuracy of the approximate solution.

Most of the calculations involved in these algorithms are sparse matrix operations such as matrix-vector multiplication and matrix-matrix addition. It seems that the matrix-matrix multiplication used in the calculation of the Schur complement is the most complicated step and needs a large amount of computation. If we just use the sparse matrix-matrix multiplication to calculate $B * B^T$ directly, the time complexity is $O(d * nnz^2)$ in which d is the largest number of nonzero elements in each row of B $(d \leq 4)$. But we can take advantage of the special meaning of the product $B * B^T$ on the corresponding graph. The (i,j)-th element of $B * B^T$ is nonzero means that the coarse-grid nodes i and j can be connected by a path with length $l \leq 2$. So we can use the breadth first search (BFS) strategy to find the positions of all the nonzero elements in $B * B^T$ and then compute their exact values. The time complexity can be reduce to $O(d^2 * nnz)$ which is actually $O(n)$ since that $d \leq 4$ and $nnz = O(n)$.

D. Iterative Refinement Scheme

In most cases, the algorithm described above can give an approximate inverse with quite high accuracy. However, since the $\mathcal{H}$-matrix is constructed based on a hierarchical block structure, we cannot ensure that approximate accuracy of each row can all meet the requirement even though the whole matrix has achieved a high approximate accuracy. The iterative refinement scheme is used to enhance the robustness of the $\mathcal{H}$-matrix-based algorithm.

A feasible strategy is to use the approximate inverse as a preconditioner of the PCG method. But the relatively large computational cost is the major drawback of this strategy. Since the required accuracy of the linear system solution in vectorless power grid verification algorithms is usually not very high, and we have already constructed a relatively good approximate inverse, a simpler iterative refinement scheme can be used: let $\widetilde{A^{-1}}$ be the approximate inverse, we can use the linear iteration

$$x_0 = 0, x_{i+1} = x_i + \widetilde{A^{-1}}(e_i - Ax_i) \qquad (21)$$

to compute a more accurate solution. The iterative refinement scheme has a convergence rate of $\left\| I - \widetilde{A^{-1}}A \right\|$. The extra computational cost is mainly the computation of the residual $r_i = e_i - Ax_i$ which can also be used in the user-specified error tolerance control procedure introduced in [4].

IV. EXPERIMENTAL RESULTS

We implement the proposed multilevel $\mathcal{H}$-matrix-based approximate matrix inversion algorithm using C++. The sparse $\mathcal{H}$-matrix arithmetic is implemented based on the $\mathcal{H}$-matrices library, HLIBpro [16]. The multilevel method is implemented based on algorithm 4. For comparison, we also use two classical methods, Cholmod [17] and ICCG, to solve the linear systems arising from vectorless power grid verification problems. The ICCG solver is implemented based on incomplete Cholesky factorization with zero fill-in. All these experiments are performed on a 64-bit Linux machine with 2.33GHz Intel Xeon E5345 processor and 8GB memory.

The test cases are generated based on the power grids used in [4]. The original power grids are mainly small and medium sized 3-D regular grids. We expand these power grids to larger size, and introduce some irregularities to these test cases randomly. The current constraints are also generated based on the benchmarks used in [4].

For each of the test cases, we choose 1000 nodes of the power grid randomly to test the run time and memory usage. The experimental results are presented in Table I. The setup time corresponds to the time spent on $\mathcal{H}$-matrix construction and $\mathcal{H}$-Cholesky factorization in $\mathcal{H}$-matrix-based method, the k-th level coarse-grid operator construction in multilevel method, and the Cholesky factorization in Cholmod. These setup operations need to be done only once. In fact, the time spent on the preconditioner construction in ICCG can be also viewed as setup time, and it is not counted in the solve time of ICCG. The solve time is the average time spent in computing each column of the inverse matrix. This is the main factor that affects the total run time since the number of columns that need to be computed is extremely large. For ICCG method, the error tolerance control procedure introduced in [4] is used to ensure that the iterative solver can achieve about the same approximate accuracy as the multilevel $\mathcal{H}$-matrix-based method, so the convergence tolerance is larger than the usual setting in DC or transient simulation, and the solve time also becomes smaller. The approximate error is measured by the average 1-norm of error vectors. The 1-norm is used instead of the 2-norm mainly for the convenience of the error estimation which is introduced below.

Fig.2 and Fig.3 show the results of solve time and memory usage comparison respectively. We use the power function fitting method to get a rough estimate of the time and space complexity of each algorithm being tested. The results are consistent with the theoretical analysis that $\mathcal{H}$-matrices can be computed and stored in almost linear complexity.

From the experimental results we can also see that the proposed multilevel matrix inversion scheme is an effective way to improve the performance of the $\mathcal{H}$-matrix-based method. The algorithm that combines the two techniques achieves 20X speed up over the ICCG solver on the largest power grid with about 1.45M nodes. It is even faster than the forward and backward substitution performed by Cholmod while the memory usage is less than half of the memory used by the direct solver.

In terms of approximate accuracy, since the exact solution has been computed by Cholmod, we can get the error vector of the approximate inverse algorithm directly instead of the residual vector. So the error estimation becomes much easier. Denote the objective function of the linear programming problem at some node by $f = \boldsymbol{c}^{\boldsymbol{T}}\mathbf{i}$. From the local current constraints we can know that the maximum current excitation is less than 10mA, i.e., $\|\mathbf{i}\|_\infty < 0.01$. It is also easy to see that $|\delta f| \leq \|\mathbf{i}\|_\infty * \|\boldsymbol{\delta c}\|_1$, where $\boldsymbol{\delta c}$ is the error vector and δf is the error of the final result caused by $\boldsymbol{\delta c}$. So on average, the error introduced by the proposed matrix inversion algorithm is only about 0.01mV~0.1mV.

V. CONCLUSIONS

In this paper, we present a multilevel $\mathcal{H}$-matrix-based

TABLE I. Experimental results of linear system solution problem

Grid Size	$\mathcal{H}$-matrix				$\mathcal{H}$-matrix + Multilevel				Cholmod			ICCG
	Setup (s)	Solve (s)	Memory	Avg. Error	Setup (s)	Solve (s)	Memory	Avg. Error	Setup (s)	Solve (s)	Memory	Solve (s)
5875	0.62	0.02	7.50MB	4.9E-4	0.46	0.01	3.80MB	1.3E-3	0.18	0.03	5.42MB	0.02
22939	3.48	0.08	33.74MB	2.5E-4	2.72	0.05	19.89MB	3.9E-4	0.76	0.12	30.09MB	0.13
35668	6.17	0.13	53.55MB	1.2E-3	4.25	0.08	29.14MB	6.6E-4	0.93	0.2	52.42MB	0.23
51195	9.55	0.19	83.01MB	9.7E-4	6.57	0.12	43.79MB	1.2E-3	1.36	0.31	84.37MB	0.36
90643	18.83	0.35	161.37MB	2.5E-3	14.37	0.22	87.05MB	2.1E-3	2.61	0.54	176.05MB	0.78
141283	31.94	0.58	254.48MB	2.0E-3	21.87	0.34	127.76MB	2.7E-3	4.54	0.89	302.26MB	1.60
203725	65.97	0.89	479.94MB	1.2E-3	37.53	0.50	196.14MB	1.2E-3	6.92	1.28	469.77MB	2.73
277559	94.71	1.22	670.13MB	3.4E-3	55.40	0.66	295.32MB	2.3E-3	8.74	1.64	687.82MB	4.82
562363	206.24	2.56	1.39GB	1.1E-3	155.79	1.42	671.49MB	1.2E-3	26.39	3.87	1.63GB	12.07
681265	344.76	3.29	2.04GB	1.0E-3	220.04	1.76	910.42MB	1.2E-3	31.68	4.54	2.09GB	16.48
953245	443.93	4.54	2.72GB	9.9E-4	371.59	2.50	1.33GB	2.0E-3	45.57	6.38	3.08GB	32.87
1446655	802.83	7.16	4.60GB	4.4E-3	1833.54	4.01	2.45GB	4.7E-3	81.13	9.82	5.61GB	87.29

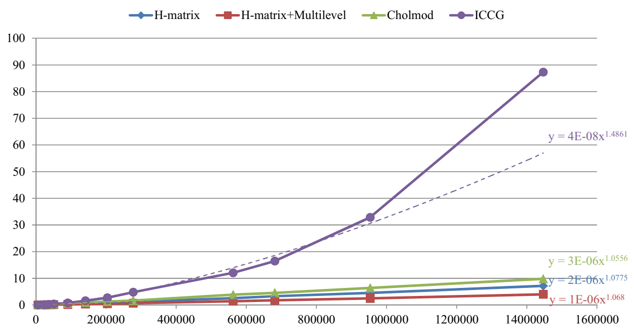


Fig.2. Solve time comparison.

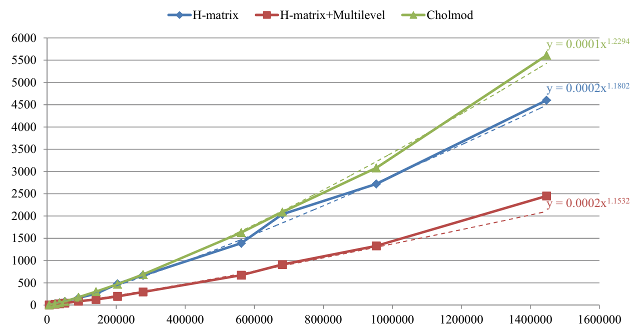


Fig.3. Memory usage comparison.

approximate matrix inversion algorithm. We combine the $\mathcal{H}$-matrix-based technique and the multilevel method to compute an approximate inverse of the power grid matrix which can be adopted in efficient vectorless power grid verification. Experimental results have shown that the combination of the two different techniques is successful. The resulting algorithm can accelerate the linear system solution significantly with relatively small memory footprint.

REFERENCES

[1] D. Kouroussis and F. N. Najm, "A static pattern-independent technique for power grid voltage integrity verification," In DAC, pages 99–104, 2003.

[2] N. H. Abdul Ghani and F. N. Najm, "Handling inductance in early power grid verification," In ICCAD, pages 127–134, 2006.

[3] N. H. Abdul Ghani and F. N. Najm, "Fast vectorless power grid verification using an approximate inverse technique," In DAC, pages 184–189, 2009.

[4] X. Xiong and J. Wang, "An efficient dual algorithm for vectorless power grid verification under linear current constraints," In DAC, pages 837–842, 2010.

[5] X. Xiong and J. Wang, "A hierarchical matrix inversion algorithm for vectorless power grid verification," In ICCAD, pages 543–550, 2010.

[6] M. Avci and F. N. Najm, "Early P/G grid voltage integrity verification," In ICCAD, pages 816–823, 2010.

[7] N. H. Abdul Ghani and F. N. Najm, "Fast vectorless power grid verification under an RLC model," IEEE Trans. Computer-Aided Design, vol. 30, no. 5, pages 691–703, May 2011.

[8] Chung-Kuan Cheng, Peng Du, Andrew B. Kahng, Grantham K.H. Pang, Yuanzhe Wang and Ngai Wong, "More realistic power grid verification based on hierarchical current and power constraints," In ISPD, pages 159-166, 2011.

[9] J. M. S. Silva, Joel R. Phillips, and L. Miguel Silveira, "Efficient simulation of power grids," ACM Trans. on CAD, 29(10):1523–1532, October 2010.

[10] M. Benzi and M. Tuma, "A comparative study of sparse approximate inverse preconditioners," Appl. Numer. Math., 1999.

[11] S. Demko, W. F. Moss and P. W. Smith, "Decay rates for inverses of band matrices," Mathematics of Computation, 43(168):491–499, October 1984.

[12] E. Chow and Y. Saad, "Approximate inverse techniques for block-partitioned matrices," SIAM J. Sci. Comput., pages 1657–1675, 1997.

[13] Y. Saad, "Iterative Methods for Sparse Linear Systems," SIAM, Philadelphia, second edition, 2003.

[14] W. Hackbusch, "A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices," Computing, pages 89-108, 1999.

[15] W. Hackbusch and B.N. Khoromskij, "A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems," Computing, pages 21-47, 2000.

[16] http://www.hlibpro.com

[17] http://www.cise.ufl.edu/research/sparse/cholmod