Fast Vectorless Power Grid Verification Using an Approximate Inverse Technique*

Nahi H. Abdul Ghani
Department of ECE
University of Toronto
Toronto, Ontario, Canada
nahi@eecg.utoronto.ca

Farid N. Najm
Department of ECE
University of Toronto
Toronto, Ontario, Canada
f.najm@utoronto.ca

ABSTRACT

Power grid verification in modern integrated circuits is an integral part of early system design where adjustments can be most easily incorporated. In this work, we describe an early verification approach under the framework of current constraints where worst-case node voltage drops are computed via linear programs proportional to the grid size. We propose an efficient method based on a sparse approximate inverse technique to greatly reduce the size of such linear programs while ensuring a user-specified over-estimation margin (in volts) on the exact solution.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Performance, Algorithms, Verification

Keywords

Power grid, voltage drop, approximate inverse

1. INTRODUCTION

A well-designed power grid in integrated circuits (ICs) should guarantee the proper logic functionality at the intended design speed. As the supply and threshold voltages decrease with technology scaling, full-chip verification is getting more and more challenging. A key concern is the fact that logic circuits slow down under reduced supply voltages thus putting overall circuit timing performance at risk. Hence, power grid analysis and verification has become central to reliable high-speed chip design.

Today, grid verification is typically done by simulation. Such an approach requires full knowledge of the current waveforms drawn by every logic block attached to the grid. These waveforms would then be used to simulate the grid and get the voltage drop at every node. Verifying the grid in this manner proves to be problematic. For one thing, the number of current traces needed to cover the space of voltage drops exhibited on the grid is intractable for modern designs where grids can consist of several million nodes. Another major drawback is that a simulation based flow does not allow for *early* grid verification, when grid modifications can be most easily incorporated. The need, then, is for a verification approach that does not depend on simulation, *i.e.*, a *vectorless* approach. Therefore, we adopt the framework of partial current specifications, in the form of *current constraints* [1], which will be detailed in section 2.2. Under such constraints, power grid verification becomes a problem of computing the maximum voltage drops subject to current constraints.

In [1], the authors adopted a DC grid model and formulated the verification problem as a set of linear programs (LPs). This approach identifies coarse problems with the grid but is oblivious to violations related to the grid dynamics. In [2], the authors considered an *RC* model of the grid and focused on developing upper bounds on the worst-case voltages using a convergent iterative process that requires the solution of LPs while stepping repeatedly through time. A later work [3] gave a closed form bound which required a single LP per node. For all these formulations, finding the worst-case voltage drop everywhere on the grid would require the application of as many LPs as there are nodes, in a sequence, where every LP is proportional to the size of the grid. This becomes prohibitive for large designs; we often refer to this as the *sequentiality* problem. In this paper, we propose a novel approach to alleviate sequentiality based on an approximate inverse technique. This is a rigorous method that takes advantage of *locality* on the grid. It captures, for any given node, the current sources that are most influential, on that node, in a fast automated way, and then it formulates a reduced-size optimization problem while ensuring a *user-specified* over-estimation margin (in volts) on the solution of the exact LPs. Experimental results in section 5 show that even for small over-estimation values, the improvement in runtime is dramatic.

2. BACKGROUND

2.1 The Power Grid Model

Consider an *RC* model of the power grid, where each branch is represented by a resistor and where there exists a capacitor from every node to ground. This work is not applicable in its present form to *RLC* grids, because it takes advantage of special properties of circuit matrices in the *RC* case. We are working on extending this to the more general case. In a power grid, some nodes have ideal current sources (to ground) representing the currents drawn by the circuits tied to the grid at those nodes, while other nodes may be connected to ideal voltage sources representing the connection to the external voltage supply.

Let the power grid consist of $n+p$ nodes, where nodes $1, 2, \ldots, n$ have no voltage sources attached, and the remaining nodes $(n+1), (n+2), \ldots, (n+p)$ are the nodes where the p voltage sources are connected. Let $i(t)$ be the element-wise non-negative vector of all current sources connected to the grid. We assume that $\forall k = 1, \ldots, n$, $i_k(t)$ is well-defined, so that nodes with no current source attached have $i_k(t) = 0$. With these assumptions, we can write the RC model for the power grid as [2]:

$$C\dot{v}(t) + Gv(t) = i(t) \tag{1}$$

*This work has been supported in part by the Semiconductor Research Corporation (SRC) (`www.src.org`).

where $v(t)$ is the $n \times 1$ vector of time varying voltage drops (difference between V_{dd} and node voltages). C is the $n \times n$ *diagonal* capacitance matrix, since all capacitors were assumed to be node-to-ground. G is the $n \times n$ conductance matrix, which is known to be a diagonally-dominant symmetric positive definite $\mathcal{M}$-matrix (so that $G^{-1} \geq 0$). Using a finite difference approximation, a discrete-time version of (1) can be written as:

$$Av(t) = \frac{C}{\Delta t} v(t - \Delta t) + i(t) \tag{2}$$

where $A = (G + \frac{C}{\Delta t})$ is also a symmetric positive definite $\mathcal{M}$-matrix, so that $A^{-1} \geq 0$. For properties of $\mathcal{M}$-matrices, the reader is refered to [4].

2.2 Current Constraints

In our approach, we perform a verification of the grid in the absence of complete information about currents drawn by the underlying circuit, what may be called a *vectorless* approach. We do this because the currents are typically hard to specify, for at least two reasons: 1) there is a large variety of possible current waveforms, so that the worst case is hard to determine up-front and simulation of a large set of waveforms is prohibitively expensive, and 2) grid design and verification cannot wait until the chip design is nearly-complete, and is typically done early in the design flow, so that the details of the underlying circuitry may not yet be available or complete.

However, while users may not know the exact circuit currents, there is always some knowledge from previous design projects which users can bring to bear. It is such knowledge that one aims to capture with the concept of *current constraints*, originally introduced in [1]. Current constraints capture the uncertainty about the circuit currents arising from both unknown circuit behaviors and the fact that one is uncertain about circuit details early in the design flow. The aim is to verify that the grid is safe (*i.e.*, its voltages remain within certain bounds), under all possible transient current waveforms which satisfy these constraints. Two types of constraints are defined: *local constraints* and *global constraints*. Local constraints are upper bounds on individual current sources, where a current source can represent a single logic gate or cell, but more typically should represent a larger block. They can be expressed as:

$$0 \leq i(t) \leq i_L, \quad \forall t \geq 0 \tag{3}$$

where i_L represents the peak value of current that this current source can draw - it is a "DC" upper bound on the transient waveform $i(t)$. It is possible to also consider upper bounds that are themselves transient waveforms over time (transient constraints), but in this paper we restrict our work to the case of DC constraints. It is important to remember, however, that it is only the constraints that are DC; the currents themselves are transient. To ensure that these constraints are well-defined for every node of the grid, we enforce the condition that any node with no current source connected would have a zero i_L component.

If *only* local constraints are provided, the problem is much simplified, but the results would be overly pessimistic, because it is never the case that all chip components simultaneously draw their maximum currents, hence the need for global constraints, which are upper bounds on the sums of groups of current sources. They represent the peak total power dissipation of a group of circuit blocks. Assuming we have a total of κ global constraints, they can be expressed in matrix form as:

$$0 \leq Si(t) \leq i_G, \quad \forall t \geq 0 \tag{4}$$

where S is a $\kappa \times n$ matrix that consists only of 0s and 1s which indicate which current sources are present in each global constraint. As for the case of local constraints, note that i_G is a fixed time-independent upper bound, a DC constraint, but the currents themselves are transient waveforms over time.

How would one obtain/specify these constraints in practice? If a logic block is available and small enough to simulate, then one can generate the constraints by an "offline" process of simulation, which can be viewed as a characterization of that block. If the block is not yet available or is too large to simulate, then one would need to rely on design expertise and engineering judgement (how big it is, what its power needs were in a previous technology and how scaling would affect those needs, etc.). If, early in the design flow, nothing is known about this block, not even its detailed functionality, one typically is able to come up with an area budget for it. From that, and from the projected power density (Watts/μm^2) for the target process technology, one can generate a rough current constraint for it. The bottom line: *something* is typically known about that block, which with good engineering judgement can be formulated into constraints. Chip designers typically use "power budgets" early in the design flow to help verify the grid, often making use of simple spreadsheet applications. The proposed constraint-based approach is a scientific and reliable approach for making use of such budgets, and it depends on good designer expertise and judgement. After all, if truly nothing is known about the circuit currents, then the grid simply cannot be verified.

Another possibility is that the current constraints can be used to implement a "spec-based" design flow; a chip-level designer would simply specify the constraints based on design expertise, and the grid is verified under these constraints. The constraints now become *design guidelines* to be observed in subsequent design activity. If all design teams follow these guidelines and verify their blocks, the end result would be a grid that is *safe by construction*.

Together, the local and global constraints define a *feasible space* of currents, which we denote by $\mathcal{F}$, so that $i(t) \in \mathcal{F}$ if and only if it satisfies (3) and (4). Later in the paper, we will define algorithms that operate on a vector $i(t)$ and which are applicable at any value of time t. In that context, we will use the shorthand $i \in \mathcal{F}$ to denote the fact that $i(t)$ is feasible, for any given t.

3. PROBLEM DEFINITION

In this section, we will show how grid verification boils down to solving linear programs that depend on a certain matrix inverse. The aim is to lay the ground-work for the rest of the paper, in which this inverse will be conservatively approximated using a very efficient approach. The novelty of this paper is in adapting a previous inverse approximation technique to the power grid verification problem. However, we do need to provide some detailed background on grid verification, to motivate the need for finding an approximation to the inverse matrix.

We are interested in the vector of maximum node voltage drops, over all possible currents in $\mathcal{F}$. Using (2), we can write:

$$v(t) = A^{-1} \frac{C}{\Delta t} v(t - \Delta t) + A^{-1} i(t) \tag{5}$$

Consider the special case where the grid had no stimulus for all $t \leq 0$ so that $v(0) = 0$. At time $t = \Delta t$, we can write:

$$\begin{aligned} v(\Delta t) &= A^{-1} \frac{C}{\Delta t} v(0) + A^{-1} i(\Delta t) \\ &= A^{-1} i(\Delta t) \end{aligned} \tag{6}$$

At $2\Delta t$ and at $3\Delta t$, we have:

$$\begin{aligned} v(2\Delta t) &= A^{-1} \frac{C}{\Delta t} v(\Delta t) + A^{-1} i(2\Delta t) \\ &= A^{-1} \frac{C}{\Delta t} A^{-1} i(\Delta t) + A^{-1} i(2\Delta t) \end{aligned} \tag{7}$$

$$\begin{aligned} v(3\Delta t) &= A^{-1} \frac{C}{\Delta t} v(2\Delta t) + A^{-1} i(3\Delta t) \\ &= \left(A^{-1} \frac{C}{\Delta t}\right)^2 A^{-1} i(\Delta t) + A^{-1} \frac{C}{\Delta t} A^{-1} i(2\Delta t) \\ &\quad + A^{-1} i(3\Delta t) \end{aligned} \tag{8}$$

This can be repeated, so that at any future time $p\Delta t$, we have:

$$v(p\Delta t) = \sum_{k=0}^{p-1} \left(A^{-1} \frac{C}{\Delta t}\right)^k A^{-1} i((p-k)\Delta t) \tag{9}$$

At every point in time $t \in [0, p\Delta t]$, the current vector $i(t)$ must be feasible, *i.e.*, we must have $i(t) \in \mathcal{F}$ and, under that condition, we are interested in the worst-case voltage attained (separately) by each component of $v(p\Delta t)$. In order to compactly capture this notion, we introduce the following notation.

Let $f(x) : \mathbb{R}^n \to \mathbb{R}^n$ be a vector function whose components will be denoted $f_1(x), \dots, f_n(x)$, and let $\mathcal{A} \subset \mathbb{R}^n$. Now, define a vector $z \in \mathbb{R}^n$, such that z_i is the maximum of $f_i(x)$ over all $x \in \mathcal{A}$. We will denote this by the following operator:

$$z = \underset{\forall x \in \mathcal{A}}{\text{emax}}(f(x)) \tag{10}$$

where the "emax($\cdot$)" notation is introduced to denote the fact that this is an *element-wise* maximization. Notice that each component z_i may be found separately by solving the optimization problem:

$$\begin{aligned} &\text{Maximize: } f_i(x) \\ &\text{Subject to: } x \in \mathcal{A} \end{aligned} \tag{11}$$

Using this notation, we can write the worst-case voltage drops at all nodes at time $\tau = p\Delta t$ as:

$$v_{max}(\tau) = \underset{\forall i(t) \in \mathcal{F}}{\text{emax}} \left(\sum_{k=0}^{p-1} \left(A^{-1} \frac{C}{\Delta t} \right)^k A^{-1} i((p-k)\Delta t) \right) \tag{12}$$

where the notation "$\forall i(t) \in \mathcal{F}$" means that, for every time point $t \in [0, \tau]$, the current $i(t)$ satisfies all the (local and global) constraints.

In practice, we are interested in the steady state solution where the system becomes independent of the initial condition ($i(t) = 0, \forall t \leq 0$). Since the RC grid model is a dynamical system with a limited memory of its past, then the steady state solution can be obtained by evaluating (12) at points far away from the initial condition, *i.e.*, as $p \to \infty$. Thus, the general solution to the exact voltage drop maximization problem is:

$$v_{max}(t) = \lim_{p \to \infty} \underset{\forall i(t) \in \mathcal{F}}{\text{emax}} \left(\sum_{k=0}^{p-1} \left(A^{-1} \frac{C}{\Delta t} \right)^k A^{-1} i((p-k)\Delta t) \right) \tag{13}$$

Although the RC model is dynamic, *i.e.*, its currents and voltages vary with time, the constraints are DC and do not depend on time. Hence, $\mathcal{F}$ is the same for each time step. With this, the components of (13) can be "decoupled," leading to:

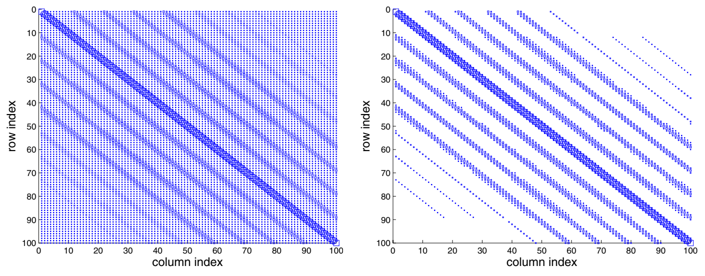
$$v_{max}(t) = \lim_{p \to \infty} \sum_{k=0}^{p-1} \underset{\forall i \in \mathcal{F}}{\text{emax}} \left[\left(A^{-1} \frac{C}{\Delta t} \right)^k A^{-1} i \right] \tag{14}$$

where i is simply an $n \times 1$ vector of variables that satisfies the (local and global) constraints, without reference to any particular point in time. This is a major simplification of the problem, as it has the advantage that the number of constraints for each maximization is fixed and does not span all previous time-points.

Unfortunately, both (13) and (14) are of theoretical interest only. They cannot be directly computed, as they stand, because they have to be evaluated for a large number of time steps until convergence and because the element-wise maximization requires *linear programs* (LPs) proportional to the number of nodes in the grid which for modern designs is in the thousands or even the millions. In previous work, various approaches to simplify (14) have been proposed, two of which will be relevant to our work, and they are as follows.

In [1], a DC model of the grid was verified subject to the current constraints. The approach employed a resistive grid model and offered a major simplification of (14). In this case, the exact maximum voltage drop vector can be computed with a single emax operation as follows:

$$v_{max}(t) = \underset{\forall i \in \mathcal{F}}{\text{emax}} \left(G^{-1} i \right) \tag{15}$$



(a) Matrix view of G^{-1} (b) Matrix view of $\hat{M}$

Figure 1: Structure of G^{-1} and $\hat{M}$ for a 100-node grid

In another approach, and given an RC grid, the authors in [3] proposed an upper bound on the exact worst-case voltage drop vector, under transient currents. As $t \to \infty$, the bound was found to be:

$$v_{max}(t) = \left(I + G^{-1} \frac{C}{\Delta t} \right) \underset{\forall i \in \mathcal{F}}{\text{emax}} \left(A^{-1} i \right) \tag{16}$$

where I is the identity matrix. The result in (16) has a run-time advantage over (14) as it requires a single emax operation. Once that is computed, the evaluation of (16) is relatively cheap. It involves a single vector scaling and a single standard linear solve (LU factorization of G).

Thus, we see that both approaches [1] and [3] give a formulation that involves the inverse of a large matrix, G in one case, and A in the other. More generally, both problem formulations involve solving the same/following generic problem:

$$v^\star = \underset{\forall i \in \mathcal{F}}{\text{emax}} \left(E^{-1} i \right) \tag{17}$$

where E is understood to be either A in the case of transient analysis [3] or G in the case of DC analysis [1]. Note that E^{-1} is symmetric, because E (be it A or G) is symmetric. In the mentioned previous work, the matrix inverse E^{-1} was not found explicitly. Instead, the problem was transformed into the voltage domain, so that it makes use of the original sparse matrix E. However, even then, the problem remained quite expensive, limiting the size of grids that may be verified. As an alternative, and in the rest of the paper, we propose an efficient approach to compute a <u>tight</u> conservative approximation of $v^\star$, based on a sparse approximation of the matrix inverse, followed by a solution of (17) using linear programming.

4. PROPOSED SOLUTION

Let M be an $n \times n$ matrix and let the vector m_k denote the kth column of M. Let e_k be the $n \times 1$ vector consisting of all zeros, except for its kth component which is 1. Then, consider the $n \times 1$ vector, called the kth *residual*, defined by:

$$r_k = E m_k - e_k, \qquad \forall k = 1, 2, \dots, n \tag{18}$$

It is clear that, if we choose $M = E^{-1}$, then $r_k = 0, \forall k$. In general, the norm of the residual $\|r_k\|_2$ (the Euclidean or l-2 norm in this case) is positive, and becomes zero only when $M = E^{-1}$. Thus, the norms of the residuals, for all k, provide a measure of how far M is from being equal to the desired matrix inverse E^{-1}. Let $M^\star \triangleq E^{-1}$, so that when M takes the value $M^\star$, then $\|r_k\|_2 = 0, \forall k$, and we want:

$$v^\star = \underset{\forall i \in \mathcal{F}}{\text{emax}} \left(M^\star i \right) \tag{19}$$

Typical power grids have a mesh structure, where a node has a small number of neighboring nodes. Such a structure gives rise to a matrix E (be it G or A) that is sparse, symmetric, positive definite, and *banded*. For such matrices, it is well known in the literature that their inverse has entries whose values decay exponentially as one moves away from the diagonal [5]. We illustrate this property in Fig. 1(a), which shows the structure of the G^{-1} matrix for a 100-node grid. In Fig. 1(a), every entry is represented, at the appropriate location, by a square whose area reflects that entry's magnitude, and very small valued entries are represented by dots. It is clear that the majority of the entries are extremely small with the exception of few significant bands in the neighborhood of the main diagonal. As expected, G^{-1} entries decay as one moves away from the diagonal. This is a *key* observation that has been the motivation for published work [6] on finding approximations to the inverse matrix that basically ignore the extremely small values in the inverse. A matrix that has a large number of entries whose values are extremely small is refered to as being *practically* sparse. In this case, $M^\star = E^{-1}$ is practically sparse. Therefore, every component of $v^\star$, which we will denote $v_k^\star$, is mainly the result of the contribution of very few current sources, specifically those that correspond to the significant entries in row k of $M^\star$. This is not surprising, because one expects that the voltage at a node depends mostly on the nearby current sources. This is well-known in power grid research, as in [7] which finds that a current source has a localized contribution on the grid. This is often refered to as the *locality effect.* The difficulty, with locality, is to develop rigorous methods for determining exactly where the neighborhood is, for a given node, and how far it extends. By using methods based on discovering the significant entries of a practically sparse matrix, we believe that we have found a rigorous approach for taking advantage of locality.

Our contribution, described in the next two sub-sections, is as follows. Given a *user-specified* over-estimation margin (in volts) for $v^\star$, we devise a fast automated method to formulate a smaller-size optimization problem, based only on the current sources that are most influential on each node, and we solve it to provide a solution that meets the over-estimation margin specified by the user. We start by briefly describing the existing approach of inverse matrix approximation and then, in the following section, we describe how we have adapted this to solve the power grid verification problem.

4.1 Inverse Approximation Method (SPAI)

A sparse approximate inverse technique (SPAI) technique is given in [6]. Given some user-defined tolerance η, SPAI computes an approximation $\hat{M}$ to the actual inverse matrix $M^\star$. It operates by finding an $\hat{M}$ matrix that gives a very small residual norm, as an approximation to the unknown $M^\star$ (which, as we saw above, would give a zero residual). The technique starts with an arbitrary initial matrix M and iteratively refines its columns by minimizing the Frobenius norm $\|EM - I\|_F$ subject to η. Recall that for any $n \times n$ matrix H, the Frobenius norm is defined as [4]:

$$\|H\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |h_{ij}|^2} \tag{20}$$

where h_{ij} are the entries of H, which can be re-written as:

$$\|H\|_F = \sqrt{\sum_{k=1}^{n} \|He_k\|_2^2} = \sqrt{\sum_{k=1}^{n} \|h_k\|_2^2} \tag{21}$$

where h_k is the kth column of H. Obviously, the minimum of (21) (which is zero) occurs when $H = 0$. Likewise, the minimum of $\|EM - I\|_F$ (which is zero) occurs when $M = E^{-1} = M^\star$. By reducing $\|EM - I\|_F$ until it is less than η, SPAI finds an $\hat{M}$ that is a good approximation to $M^\star = E^{-1}$. To see how SPAI works, consider that:

$$\begin{aligned}\|EM - I\|_F^2 &= \sum_{k=1}^{n} \|(EM - I)e_k\|_2^2 \\ &= \sum_{k=1}^{n} \|Em_k - e_k\|_2^2 = \sum_{k=1}^{n} \|r_k\|_2^2\end{aligned} \tag{22}$$

In this way, the problem of finding an approximate inverse $\hat{M}$ separates into n *independent* least-squares problems:

> *For each $k = 1, 2, \ldots, n$, vary m_k so as to minimize $\|r_k\|_2^2$ until an m_k is found for which $\|r_k\|_2 \le \eta$.*

While $\|r_k\|_2 > \eta$, the iterative refinement of m_k proposed by [6] attempts to improve on the column by appending the most profitable entries, *i.e.*, the ones that will result in the largest reduction in $\|r_k\|_2$. In this manner, the technique automatically captures the sparsity pattern by including the significant entries of the inverse while avoiding fill-ins.

The numerical study in [6] shows that SPAI captures the main entries of E^{-1} extremely well. Moreover, it is an extremely efficient technique. It is inherently parallel as the columns of $\hat{M}$ can be computed independently of one another. Moreover, the computation of a column is cheap; it requires a matrix-vector product and several QR factorizations of small submatrices of the original sparse matrix E.

4.2 An Upper Bound Vector

The $\hat{M}$ obtained by SPAI can be used to give an approximate solution to the exact optimization problem in (19). However, for power grid verification, we are interested in, not simply *any* approximation to the $v^\star$, but in a *conservative* approximation to $v^\star$, *i.e.*, an upper-bound on it. In other words, we want an $\hat{M}$ which, if used instead of $M^\star$ in (19), would produce a $\hat{v}$ that we can use to construct an upper bound $\bar{v}$ on $v^\star$, so that $v^\star \le \bar{v}$. In this section, we describe how we adapt SPAI to our needs, by modifying its stopping criterion, and then how we generate our upper bound $\bar{v}$ based on the matrix $\hat{M}$ achieved by SPAI under the new stopping criterion. Crucially, we will show that the over-estimation $\|\bar{v} - v^\star\|$ is under control and can in fact be specified up-front.

4.2.1 Stopping criterion

Normally, SPAI terminates when $\|r_k\|_2$ is smaller than a user-supplied η, for every k. However, in order to arrive at predictable over-estimation bounds, as we will see below, a new stopping criterion will now be introduced. If $m_k^\star$ is the kth column of $M^\star = E^{-1}$, then clearly $Em_k^\star = e_k$, so that $r_k = Em_k - e_k = E(m_k - m_k^\star)$. This leads to:

$$\|m_k - m_k^\star\|_\infty = \|E^{-1}r_k\|_\infty \le \|E^{-1}\|_\infty \|r_k\|_\infty \tag{23}$$

where $\|\cdot\|_\infty$ is the infinity norm. We define the error tolerance $\epsilon_k > 0$, where $\epsilon_k \in \mathbb{R}$ is a scalar. As we will see below, ϵ_k is derived from another user-specified error-tolerance on voltages. We modify SPAI so that it stops when, for every k, we have:

$$\|r_k\|_\infty \le \frac{\epsilon_k}{\|E^{-1}\|_\infty} \tag{24}$$

which achieves the condition that, for every k:

$$\|\hat{m}_k - m_k^\star\|_\infty \le \epsilon_k \tag{25}$$

It remains to explain how $\|E^{-1}\|_\infty$ is found, when E^{-1} is unknown! This is easy to do, in fact, because E is an $\mathcal{M}$-matrix, as follows. Let u be the $n \times 1$ vector of all 1s, so that $u^T = [1\ 1\ \cdots\ 1]$. Then, $E^{-1}u$ is a vector whose every component is the sum of all the entries in the corresponding row of E^{-1}. Because $E^{-1} \ge 0$, then each component of $E^{-1}u$ is actually the sum of the absolute values of all the entries in that row of E^{-1}. Thus, the component with the largest value in $E^{-1}u$ is, in fact, the "largest absolute row sum" of E^{-1} which is its infinity norm. In other words, if x is a vector such that $Ex = u$, then it's clear that:

$$\|E^{-1}\|_\infty = \|E^{-1}u\|_\infty = \|x\|_\infty \tag{26}$$

Finding this x is easy, by performing one LU-factorization of E, which we have to do anyway as part of power grid verification, followed by a backward and a forward solve starting with u.

Algorithm 1 UPPER_BOUND

```
Input: $E$, $\delta$
Output: $\bar{v}$
 1: Compute $\|E^{-1}\|_\infty$ using (26):
        LU-factorize $E$ :  $E = L \cdot U$
        Forward solve: $Ly = u$
        Backward solve: $Ux = y$
        Get: $\|E^{-1}\|_\infty = \|x\|_\infty$
 2: $(\hat{M},\epsilon)$ = Modified_SPAI$(E, \|E^{-1}\|_\infty, \delta)$
 3: Compute $\hat{v}$ using (27):
        Maximize: $\hat{M}i$
        Subject to: $i \in \mathcal{F}$
 4: $\bar{v} = \hat{v} + (u^T i_L)\epsilon$
```

4.2.2 Upper bound

Once SPAI terminates with its approximation matrix $\hat{M}$, using the new stopping criterion, we run our optimization (19) using $\hat{M}$ instead of $M^\star$, to get:

$$\hat{v} = \underset{\forall i \in \mathcal{F}}{\text{emax}} \left(\hat{M}i\right) \tag{27}$$

and this optimization run is quite fast, as we will see in the results section. In this section, we will show how the $\hat{v}$ resulting from this optimization is used to generate a tight upper bound $\bar{v}$ on $v^\star$, in a way that the over-estimation $\|\bar{v} - v^\star\|$ is under control and predictable up-front. We will do this by first showing how (25) leads to an upper bound on $M^\star$, based on $\hat{M}$ and ϵ_k. We then show how this leads to an upper bound on $v^\star$, based on $\hat{v}$ and ϵ_k.

SPAI starts with an initial $M = I$, and tries to maintain sparsity and reduce fill-ins throughout the process. So, many of the entries of $\hat{m}_k$ will be equal to 0. Let $\mathcal{C}_k$ be the set of non-zero entries in $\hat{m}_k$. The number of such entries, denoted by $|\mathcal{C}_k|$, is often considerably smaller than n. To compute an upper bound on $m^\star_{jk}$, for all j, k, two cases have to be considered, according to whether $\hat{m}_{jk} \in \mathcal{C}_k$ or not. If $\hat{m}_{jk} \notin \mathcal{C}_k$, *i.e.*, $\hat{m}_{jk} = 0$, then (25) yields:

$$|-m^\star_{jk}| \le \epsilon_k \Rightarrow 0 \le m^\star_{jk} \le \epsilon_k \tag{28}$$

so that ϵ_k is an upper-bound on $m^\star_{jk}$, with a maximum over-estimation error of ϵ_k. In the second case, when $m_{jk} \in \mathcal{C}_k$, then, (25) gives:

$$|\hat{m}_{jk} - m^\star_{jk}| \le \epsilon_k \Rightarrow \hat{m}_{jk} - \epsilon_k \le m^\star_{jk} \le \hat{m}_{jk} + \epsilon_k \tag{29}$$

so that $\hat{m}_{jk} + \epsilon_k$ is an upper-bound on $m^\star_{jk}$, with a maximum over-estimation error of $2\epsilon_k$. Let ϵ be the $n \times 1$ vector such that $\epsilon^T = [\epsilon_1\ \epsilon_2\ \cdots\ \epsilon_n]$. We will now prove that these upper bounds on the entries of $M^\star$ lead to an upper bound $\bar{v}$ on $v^\star$, as follows:

$$\bar{v} = \hat{v} + (u^T i_L)\epsilon \tag{30}$$

where u is the same vector of all 1s introduced above, so that $u^T = [1\ 1\ \cdots\ 1]$. The proof of this result is given in the following claim, which also provides the measure of over-estimation between $v^\star$ and $\bar{v}$.

CLAIM 1. *For every $k = 1, 2, \ldots, n$, let u_k be the $n \times 1$ vector whose jth entry is 1 if $\hat{m}_{jk} \in \mathcal{C}_k$, and otherwise 0. Then, $\bar{v}$ in (30) is an upper-bound on $v^\star$, with a maximum over-estimation error of:*

$$(\bar{v}_k - v^\star_k) \le \delta_k \triangleq \left(u^T i_L + u_k^T i_L\right)\epsilon_k, \quad \forall k = 1, \ldots, n \tag{31}$$

PROOF. In the following, and for any $k \in \{1, 2, \ldots, n\}$, we will prove that $0 \le \bar{v}_k - v^\star_k \le \delta_k$. The proof addresses first the lower bound (of 0) and then the upper bound (of δ_k).

Algorithm 2 Modified_SPAI$(E, \|E^{-1}\|_\infty, \delta)$

```
 1: $M = I$ {initialize $M$ to the identity matrix}
 2: for $k = 1 : n$ do
 3:     $r_k = Em_k - e_k$
 4:     Compute $\epsilon_k$ in (32)
 5:     while ($\|r_k\|_\infty > \epsilon_k / \|E^{-1}\|_\infty$) do
 6:         Minimize $\|Em_k - e_k\|_2$ and update $m_k$ {this is the result
            of a $QR$ factorization of a sub-matrix of $E$ }
 7:         Update $r_k$ and $\epsilon_k$
 8:     end while
 9: end for
10: return ($M, \epsilon$)
```

The lower bound: For the kth component of $v^\star$, and using the bounds on the entries of $M^\star$ seen above, and because $M^\star$ and $\hat{M}$ are both symmetric, it is easy to see that:

$$\begin{aligned}
v^\star_k &= \max_{\forall i \in \mathcal{F}} \left(\sum_{j=1}^n m^\star_{kj} i_j\right) = \max_{\forall i \in \mathcal{F}} \left(\sum_{j=1}^n m^\star_{jk} i_j\right) \\
&\le \max_{\forall i \in \mathcal{F}} \left(\sum_{\forall j, \hat{m}_{jk} \neq 0} (\hat{m}_{jk} + \epsilon_k) i_j + \sum_{\forall j, \hat{m}_{jk} = 0} \epsilon_k i_j\right) \\
&= \max_{\forall i \in \mathcal{F}} \left(\sum_{\forall j, \hat{m}_{jk} \neq 0} \hat{m}_{jk} i_j + \epsilon_k \sum_{j=1}^n i_j\right) \\
&= \max_{\forall i \in \mathcal{F}} \left(\sum_{j=1}^n \hat{m}_{jk} i_j + \left(u^T i\right)\epsilon_k\right) \\
&\le \max_{\forall i \in \mathcal{F}} \left(\sum_{j=1}^n \hat{m}_{jk} i_j\right) + \max_{\forall i \in \mathcal{F}} \left(\left(u^T i\right)\epsilon_k\right) \\
&= \max_{\forall i \in \mathcal{F}} \left(\sum_{j=1}^n \hat{m}_{kj} i_j\right) + \max_{\forall i \in \mathcal{F}} \left(\left(u^T i\right)\epsilon_k\right) \\
&\le \hat{v}_k + \left(u^T i_L\right)\epsilon_k = \bar{v}_k
\end{aligned}$$

so that $\bar{v}_k - v^\star_k \ge 0$.

The upper bound: It is easy to see that, if the optimal $\hat{v}_k$ is achieved at some current vector i', then it must also be the case that $\sum_{j=1}^n \hat{m}_{kj}\hat{i}_j = \hat{v}_k$, where $\hat{i}$ is obtained from i' by setting to 0 every jth entry of i' for which $\hat{m}_{kj} = 0$. In the following, we make use of such a vector $\hat{i}$ at which $\hat{v}_k$ is achieved.

If we define $v^\dagger_k \triangleq \sum_{j=1}^n m^\star_{kj}\hat{i}_j$, then obviously $v^\dagger_k \le v^\star_k$, because $v^\star_k$ is optimal and $\hat{i} \in \mathcal{F}$, and $\bar{v}_k - v^\star_k \le \bar{v}_k - v^\dagger_k$. By making use of the fact that $|\hat{m}_{jk} - m^\star_{jk}| \le \epsilon_k$, due to (25), and because $M^\star$ and $\hat{M}$ are both symmetric, this expands as follows:

$$\begin{aligned}
\bar{v}_k - v^\star_k &\le \bar{v}_k - v^\dagger_k \\
&= \sum_{j=1}^n \hat{m}_{kj}\hat{i}_j + \left(u^T i_L\right)\epsilon_k - \sum_{j=1}^n m^\star_{kj}\hat{i}_j \\
&= \sum_{j=1}^n \hat{m}_{jk}\hat{i}_j + \left(u^T i_L\right)\epsilon_k - \sum_{j=1}^n m^\star_{jk}\hat{i}_j \\
&= \sum_{j=1}^n \left(\hat{m}_{jk} - m^\star_{jk}\right)\hat{i}_j + \left(u^T i_L\right)\epsilon_k \\
&\le \sum_{j=1}^n \epsilon_k \hat{i}_j + \left(u^T i_L\right)\epsilon_k = \left(u_k^T \hat{i}\right)\epsilon_k + \left(u^T i_L\right)\epsilon_k \\
&\le \left(u_k^T i_L + u^T i_L\right)\epsilon_k
\end{aligned}$$

□

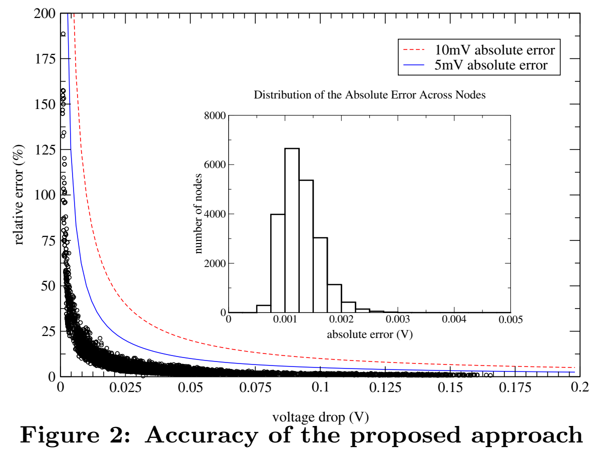


Figure 2: Accuracy of the proposed approach

Our algorithm accepts δ as a user-specified error-tolerance on node voltage drops. During the algorithm, as SPAI is iterating on a column, causing changes in u_k, we repeatedly update ϵ_k based on:

$$\epsilon_k = \frac{\delta}{u_k^T i_L + u^T i_L} \tag{32}$$

until such time when the stopping criterion (24) is met. By claim 1, this achieves the result that:

$$\|\bar{v} - v^\star\|_\infty \leq \delta \tag{33}$$

Algorithm 1 provides a high-level description of our approach which uses our modified version of SPAI (algorithm 2). The algorithm takes a user-defined δ, computes $\|E^{-1}\|_\infty$ and calls Modified_SPAI which returns $\hat{M}$ and ϵ. The algorithm will then compute $\hat{v}$ leading to $\bar{v}$.

5. EXPERIMENTAL RESULTS

The above algorithm, including the modified SPAI component, has been implemented in C++. It computes the column approximations of the matrix inverse, generates the reduced optimization problem and solves it using the Mosek optimization package [8]. A number of tests were conducted on a set of randomly-generated power grids and computations were carried on a 64-bit Linux machine with 8 GB of memory. The grids are based on user specifications, including grid dimensions, metal layers (M1-M4), pitch and width per layer, and C4 and current source distribution. Four global constraints were specified for the grids under study.

Fig. 1(b) shows $\hat{M}$ as a result of running Algorithm 2 on the 100-node grid mentioned earlier. By comparing this figure to Fig. 1(a), it is clear that Modified_SPAI captures the significant entries of the inverse extremely well while avoiding fill-ins. The resulting $\hat{M}$ is practically sparse with an average of 20 entries per row/column.

We ran Algorithm 1 on a 21,099-node grid for $\delta = 5$mV and monitored the relative over-estimation error for the kth entry as $(\bar{v}_k - v_k^\star)/v_k^\star$. Fig. 2 shows a scatter plot of such errors, in percent, versus the entries of $v^\star$. The figure also shows the curve corresponding to $\delta = 5$mV where a point on the curve represents a $v^\star$ entry over-estimated by exactly 5mV. On the other hand, a point with an over-estimation greater than 5mV will lie in the region above the curve. From the resulting data distribution, it is clear that our algorithm guarantees the specified over-estimation δ for all entries, as it should, by claim 1. Note that for some entries, especially those with magnitudes less than δ, the relative error can be high. This, however, is of no concern in practice, as the critical $v^\star$ entries are those that are significantly larger than δ. In that case, the relative error incurred is small. The absolute error is always, of course, guaranteed to be less than δ, as we have seen. In fact, the absolute errors can be much less than δ, as seen in the histogram embedded in the above figure, which shows that the actual over-estimation is often less than $\delta/2$.

Before proceeding with the results, we note that computing the exact $v^\star$ in (19), against which we will compare our results, does not have to involve the expensive computation of $M^\star$. The problem can be transformed as shown in [1] from an optimization in the current space to an optimization in the voltage space by defining a vector y, such that $Ey = i$, and then solving $v^\star = \text{emax}(y)$, $\forall Ey \in \mathcal{F}$. The exact solution, using this approach, remains expensive and involves solving one LP for every node on the grid. Our algorithm also solves an LP for every node, but each of these LPs is much smaller, because it benefits from locality (we retain only the most influential current sources, as determined by the modified SPAI routine). The result is an extremely fast optimization run.

Power Grid	**Runtime**			
	Our Approach			
Nodes	$\delta = 5mV$	$\delta = 10mV$	$\delta = 20mV$	$v^\star$ comp.
8,413	46 min.	38.2 min.	35.65 min.	3.39 h.
21,099	5.42 h.	4.85 h.	4 h.	26.5 h.
39,391	11.48 h.	8.56 h.	8.16 h	103.58 h.
50,444	15.67 h.	11.69 h.	10.8 h.	222.7 h.
72,692	25.7 h.	21.36 h.	19.38 h.	21.38 days
113,304	58.16 h.	47.0 h.	43.8 h.	66.4 days

Table 1: Speed comparisons of the proposed approach with the exact approach

Table 1 gives runtimes for computing $\bar{v}$ subject to different values of δ. The runtimes take into account the Modified_SPAI algorithm runtime. The table also gives the CPU time required for computing $v^\star$, using the above mentioned approach of converting the problem to the voltage domain. Our method was tested on all grid nodes. Given the size of the test grids, it was impossible to solve all n exact LPs associated with the computation of $v^\star$. For this purpose, we chose 1000 random nodes, solved the 1000 corresponding exact LPs, and estimated the runtime of the exact approach. Results show that even for relatively small δ's, our proposed approach runs several orders of magnitude faster than the exact computation which exhibits impractical runtimes. For instance, computing $\bar{v}$ for a 50,444-node grid subject to $\delta = 5mV$ takes around 15.67 hours while computing $v^\star$ requires around 222.7 hours. This is over 14X speedup, and the last test case in the table shows an even higher, 27X, speedup! Thus, our method makes checking a power grid under incomplete current specification a feasible and practical solution. It is also highly parallelizable, both in the modified-SPAI module and in the rest of the algorithm where the separate LPs can be run in parallel.

6. CONCLUSION

We describe an early verification approach under the framework of capturing circuit uncertainty via current constraints and maximizing node voltage drops over the constraint space. Our proposed method takes advantage of locality on the grid and formulates, for any given node, a reduced-size optimization problem while ensuring a user-specified over-estimation margin (in volts) on the exact solution. With this technique, verifying power grids under incomplete current specification becomes scalable and practical. The method shows a drastic improvement in runtime, doing in hours what required days in previous approaches.

7. REFERENCES

[1] D. Kouroussis and F. N. Najm. A static pattern-independent technique for power grid voltage integrity verification. In *ACM/IEEE DAC*, pages 99–104, Anaheim, CA, June 2-6 2003.

[2] M. Nizam, F. N. Najm, and A. Devgan. Power grid voltage integrity verification. In *ACM/IEEE ISLPED*, pages 239–244, San Diego, CA, August 8-10 2005.

[3] I. A. Ferzli, F. N. Najm, and L. Kruze. A geometric approach for early power grid verification using current constraints. In *ACM/IEEE ICCAD*, pages 40–47, San Jose, CA, November 5-8 2007.

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

[5] 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.

[6] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, May 1997.

[7] E. Chiprout. Fast flip-chip power grid analysis via locality and grid shells. In *ACM/IEEE ICCAD*, pages 485–488, San Jose, CA, Novembre 7-11 2004.

[8] Mosek - http://www.mosek.com/.