2017 IEEE Computer Society Annual Symposium on VLSI

Parallel Simulation-Based Verification of RC Power Grids

Mohammad Fawaz
ECE Department
University of Toronto
Toronto, Canada
mohammad.fawaz@mail.utoronto.ca

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

***Abstract*—The power delivery network (PDN) must undergo a sequence of verification steps throughout the integrated circuit (IC) design flow. Typically, this is done by performing a transient simulation of the grid under certain input current traces, and checking that the resulting node voltages are within some user-specified thresholds. Existing tools require solving a large number of linear systems making them slow for modern power grids with billions of nodes. We propose a parallel simulation-based tool for RC grid verification that generates envelope waveforms on the true voltage drop waveforms. The resulting waveforms capture the peaks of the voltage drops quite accurately and require solving a much smaller number of linear systems than traditional tools.**

***Keywords*-Power grid, simulation, verification, multi-threading**

I. Introduction

To guarantee a correct and reliable functionality of an integrated circuit (IC), the power grid must undergo a series of checks to make sure that the right voltage levels are achieved at each node in the grid. Unwanted voltage fluctuations often lead to a decline in the performance of the chip both in terms of functionality and speed. Therefore, there is a clear need for efficient power grid verification.

The two main causes for voltage fluctuations are IR drops and Ldi/dt noise. In many cases, the inductive effect in the grid is ignored in order to simplify the analysis. This leads to a pure RC model of the grid where only *undershoots* are observed. To check the safety of an RC power grid, one must verify that the voltage drop at every node never exceeds a certain user-defined threshold.

The verification processes is typically done using a *transient simulation* of the grid where some of the nodes are assumed to be loaded by a set of user-defined current waveforms that represent the currents drawn by the underlying logic circuitry. Typically, this is done by solving the ordinary differential equation (ODE) resulting from applying Nodal Analysis [1]. Unfortunately, state-of-the-art power grid verification tools suffer from serious performance issues due to the size of modern grids containing billions of nodes, and due to the large number of linear systems that need to be solved for each simulation.

Many algorithms have been proposed for transient simulation by exploiting different tradeoffs. Traditional approaches use the standard LU factorization, Cholesky factorization [2], or the preconditioned conjugate gradient (PCG) method [3]. Other tools use random walks [4] and multigrid techniques [5]. Hierarchical approaches are proposed in [6]. Parallelization of forward/backward substitution is proposed in [7]. A solution utilizing the matrix exponential kernel is proposed in [8].

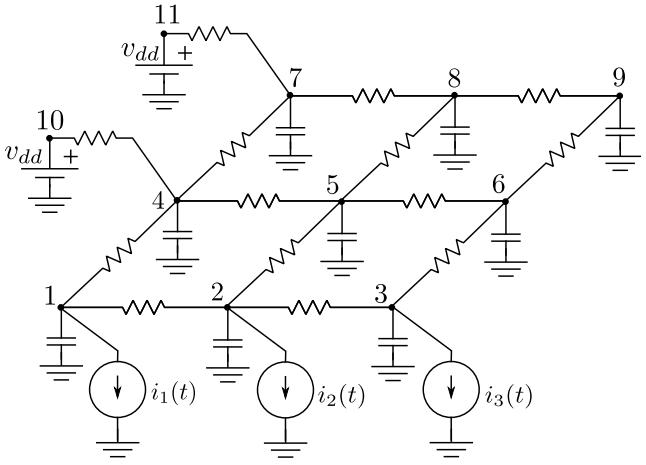

Figure 1: The RC grid model

In fact, for verification purposes, the *true* voltage drop waveforms are not necessary, and any set of *upper bound* waveforms accurately capturing the peaks of the voltage drops can be just as useful. In this paper, we present two *parallel* algorithms for power grid verification that efficiently generate *envelope* waveforms for the true voltage drop waveforms. Generating these envelope waveforms is much faster than finding the true waveforms because the envelope waveforms only consider the breakpoints in the input current traces. In other words, the total number of linear systems that need to be solved is reduced dramatically. A preliminary version of this work appeared in [9]. Here, we also develop a *multi-threaded* version of both algorithms and show that they present significant speedups over exact simulation.

The remainder of the paper is organized as follows. In section II, we review some background material on the power grid model and the current waveforms. Sections III and IV, explain how the envelope waveforms are generated. Section V explains some implementation details and section VI discusses how our methods are parallelized. Finally, section VII provides the test results and section VIII concludes the paper.

II. Background

A. Power Grid Model

The power grid is a full-chip multiple-layer mesh of metal lines that connect the external supply pins on the package to the underlying chip circuitry, providing the required electrical power to all circuit components.

2159-3477/17 $31.00 © 2017 IEEE
DOI 10.1109/ISVLSI.2017.84

Consider a power grid similar to the one shown in Fig. 1 where each node is one of three types:

1) **Type 1:** nodes that are connected to ideal current sources to ground, in parallel with capacitors to ground.
2) **Type 2:** nodes that are connected to resistors to other grid nodes and capacitors to ground.
3) **Type 3:** nodes that are connected to resistors to other grid nodes and ideal voltage sources to ground.

In the power grid of Fig 1:

- Nodes 1, 2, and 3 are of type 1.
- Nodes 4, 5, 6, 7, 8, and 9 are of type 2.
- Nodes 10 and 11 are of type 3.

The current sources with their parallel capacitors represent the currents drawn by the underlying logic blocks, and the ideal voltage sources represent the external voltage supply v_{dd}. Notice that this model assumes that *every* capacitor is connected to ground. Models that include node-to-node capacitors are usually much harder to analyze, and are not considered in this work. In fact, there is, in general, little interest in RC models that are not identical to the one described here.

Assume that the grid has q voltage sources, m current sources, and $n+q$ nodes (excluding the ground node) with $n \geq m$. Also, let the nodes be numbered $1, 2, \ldots, n+q$ such that:

- Nodes $1, 2, \ldots, n$ are the nodes not connected to any voltage source.
- Nodes $(n+1), (n+2), \ldots, (n+q)$ are the nodes where the q voltage sources are connected.
- Nodes $1, 2, \ldots, m$ are the nodes where the m current sources are connected. The positive (reference) direction of each of the source currents is assumed to be from node-to-ground.

The nodes in the power grid of Fig 1 are numbered according to the convention above with $q = 2$, $m = 3$, and $n = 9$.

Let $i_s(t) \geq 0$ be the $n \times 1$ vector of all source currents such that the kth entry $i_{s,k}(t)$ is a 1-dimensional function of time $t \in \mathbb{R}$ and corresponds to current source k. If a node is not connected to a current source, its corresponding entry in $i_s(t)$ is set to zero.

Let $u(t)$ be the $n \times 1$ vector of node voltages relative to ground, such that the kth entry $u_k(t)$ is a 1-dimensional function of time $t \in \mathbb{R}$, and corresponds to node k. Nodes $n+1, \ldots, n+q$ have been left out because their voltage is already known to be v_{dd}. By superposition, the vector $u(t)$ can be computed in three steps:

1) Open-circuit all the current sources, keep all the voltage sources, and find the resulting voltage response, which we call $u^{(1)}(t)$. Clearly, $u^{(1)}(t) = v_{dd}$ because at steady state, each capacitor will act as an open-circuit, and so, there will be no current running through the resistors, thus keeping the voltage at every node at v_{dd}.
2) Short-circuit all the voltage sources, keep all the current sources, and find the resulting voltage response, which we call $u^{(2)}(t)$.
3) Find $u(t) = u^{(1)}(t) + u^{(2)}(t)$.

To find $u^{(2)}(t)$, Kirchoff's Current Law (KCL), at every node $k = 1, \ldots, n$ provides, via Nodal Analysis (NA) [1], the following matrix formulation

$$Gu^{(2)}(t) + C\dot{u}^{(2)}(t) = -i_s(t) \tag{1}$$

where:

- The matrix G is the *sparse* $n \times n$ *conductance* matrix containing information about the resistive components of the grid. Essentially, the (j,k)th entry of G is the coefficient of the kth component of $u^{(2)}(t)$ in the jth KCL equation. It is known that G is symmetric, diagonally dominant with positive diagonal entries and non-positive off-diagonal entries. Under the standard assumptions that the resistive mesh of the grid is connected and that grid has at least one voltage source, the matrix G becomes irreducibly diagonally dominant [1]. With this, it can be shown that G is a symmetric $\mathcal{M}$-matrix, which leads to the crucial result that G^{-1} exists and $G^{-1} \geq 0$. Moreover, being a symmetric $\mathcal{M}$-matrix, G is also a symmetric positive definite matrix [10].
- The matrix C is the $n \times n$ *capacitance* matrix containing information about all the capacitive components in the grid. Due to the fact that only node-to-ground capacitors are considered, the matrix C is a positive diagonal matrix where the (j,j)th entry is the value of the capacitor connected from node j to ground. Notice that, because every node has a capacitor to ground, C is invertible.

For voltage integrity checking, we are mostly interested in the *voltage drop* at each node which is defined as $v_k(t) \triangleq v_{dd} - u_k(t)$ for nodes $k = 1, \ldots, n$. Voltage drops are more useful than actual node voltages because they represent how *far* the node voltages are from their nominal value v_{dd} regardless of the value of v_{dd}. Moreover, most power grid voltage constraints are provided as limits on the voltage drops. Let $v(t)$ be the vector of all voltage drops so that $v(t) = v_{dd} - u(t) = u^{(1)}(t) - u(t) = -u^{(2)}(t)$. This allows rewriting (1) as

$$G(-v(t)) + C(-\dot{v}(t)) = -i_s(t) \tag{2}$$

or equivalently

$$Gv(t) + C\dot{v}(t) = i_s(t). \tag{3}$$

A curious observation is that, based on (3), $v(t)$ can be found directly as the vector of node voltages resulting from an analysis of the RC mesh modified such that all voltage sources are short-circuited and all current source directions are reversed.

Effectively, (3) is a system of *first order differential equations* that describes the dynamics of the power grid given a certain time-varying *stimulus* vector $i_s(t)$. Solving (3) is most commonly done by first discretizing time using a finite-difference approximation of the derivative. Using the Backward Euler (BE) numerical scheme [11], the derivative is approximated as follows

$$\dot{v}(t) \approx \frac{v(t) - v(t-h)}{h} \tag{4}$$

for some time step h, which we assume to be fixed. This leads to the following *discretized* version of (3)

$$Gv(t) + C\frac{v(t) - v(t-h)}{h} \approx i_s(t) \tag{5}$$

or equivalently

$$\left(G + \frac{C}{h}\right) v(t) \approx \frac{C}{h} v(t-h) + i_s(t). \tag{6}$$

Recall that the BE scheme is motivated by the following Taylor expansion of a one dimensional time function $y(t)$

$$y(t-h) = y(t) - h\dot{y}(t) + \frac{h^2}{2} y(\xi) \tag{7}$$

for some $\xi \in [t-h, t]$. Thus, the choice of h relates to only the spectral properties of the node voltage signals. We assume that the time step is chosen such that (6) is accurate enough irrespective of the current stimulus $i_s(t)$. Accordingly, (6) leads to the following

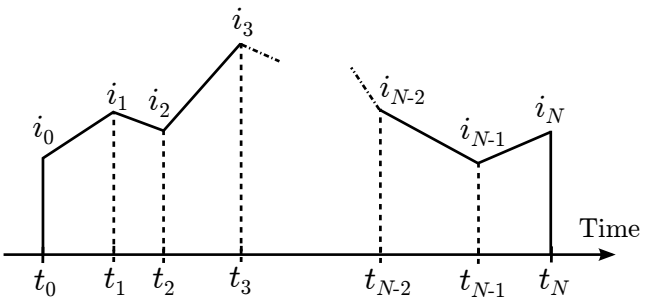

Figure 2: Piece-wise linear current waveform

recurrence relation that captures the evolution of the system over time:

$$Av(t) = Bv(t-h) + i_s(t) \tag{8}$$

where $A = G + \frac{C}{h}$ and $B = \frac{C}{h}$.

Similarly to G, the matrix A is an $\mathcal{M}$-matrix, meaning the inverse A^{-1} of A exists and $A^{-1} \geq 0$. This allows rewriting the recurrence in (8) as

$$v(t) = A^{-1}Bv(t-h) + A^{-1}i_s(t) \tag{9}$$

which is a very useful form for computing the voltage drops at time t given the voltage drops at time $t-h$ and the current stimulus at time t.

It is worth mentioning that other methods for discretizing (3) exist such as the second order Trapezoidal Rule (TR), which leads to the following recurrence:

$$\left(\frac{C}{h} + \frac{G}{2}\right) v(t) = \left(\frac{C}{h} - \frac{G}{2}\right) v(t-h) + \frac{i_s(t-h) + i_s(t)}{2}. \tag{10}$$

The trapezoidal method is generally more accurate than BE and is the method often employed in state-of-the-art simulation tools.

B. Current Waveforms

Generally, the current sources are characterized as piecewise-linear waveforms that result from simulating the underlying logic blocks for a certain period of time. In this work, we assume that the currents are given by the user in the form of a set of pairs

$$\mathcal{I} = \{(t_0, i_0), (t_1, i_1), \ldots, (t_N, i_N)\} \tag{11}$$

where $0 = t_0 < t_1 < \ldots < t_N$ are the time points at which the current vector $i_s(t)$ is available (breakpoints), and $i_0, i_1, \ldots, i_N$ are $n \times 1$ vectors containing the current values at $t_0, t_1, \ldots, t_N$ respectively. We also assume that $i_s(t) = 0$ for all $t < 0$ and $t > t_N$. Moreover, let T be the set of all time points

$$T = \{t_0, t_1, \ldots, t_N\}, \tag{12}$$

and define the intervals

$$T_k = [t_k, t_{k+1}], \quad \forall k \in \{0, \ldots, N-1\}. \tag{13}$$

III. Transient Envelope Waveforms

For the development of our proposed approach, we are going to use the BE discretization scheme. The trapezoidal scheme will only be used in the experimental results section to evaluate our tools against it, both in terms of accuracy and speed.

Applying the recurrence (9) at $t-h$

$$v(t-h) = A^{-1}Bv(t-2h) + A^{-1}i_s(t-h), \tag{14}$$

and substituting this for $v(t-h)$ back in (9) gives

$$v(t) = (A^{-1}B)^2 v(t-2h) + A^{-1}BA^{-1}i_s(t-h) + A^{-1}i_s(t), \tag{15}$$

and in general, for any integer $p \geq 1$, we can write

$$v(t) = (A^{-1}B)^p v(t-h) + \textstyle\sum_{q=0}^{p-1} (A^{-1}B)^q A^{-1} i_s(t-qh). \tag{16}$$

Recall that the *spectral radius* of a matrix Y, denoted by $\rho(Y)$, is the magnitude of the largest eigenvalue of Y. From [12], we know that $\rho(A^{-1}B) < 1$. This implies [13] that

$$\lim_{p \to \infty} (A^{-1}B)^p = 0. \tag{17}$$

Therefore, because $v(t-ph)$ is always bounded (the grid being a stable system with bounded inputs), we also have

$$\lim_{p \to \infty} (A^{-1}B)^p v(t-ph) = 0. \tag{18}$$

Thus, we can let $p \to \infty$ in (16) to get

$$v(t) = \textstyle\sum_{q=0}^{\infty} (A^{-1}B)^q A^{-1} i_s(t-qh). \tag{19}$$

We are going to develop DC and transient bounds on $v(t)$ in the form of envelope waveforms. For that, we need to define the following extreme-value operator.

Definition 1. (emax). Let $f(x) : \mathbb{R} \to \mathbb{R}^n$ be a vector function whose components will be denoted $f_1(x), \ldots, f_n(x)$, and let $\mathcal{A} \subset \mathbb{R}$. We define the operator $\underset{x \in \mathcal{A}}{\text{emax}}[f(x)]$ as one that provides the $n \times 1$ vector $y = \underset{x \in \mathcal{A}}{\text{emax}}[f(x)]$ such that, for every $j \in \{1, 2, \ldots, n\}$,

$$y_j = \max_{x \in \mathcal{A}} [f_j(x)] \tag{20}$$

Thus, $\text{emax}[\cdot]$ performs element-wise maximization over the domain of the one dimensional variable x. The $\text{emax}[\cdot]$ operator can also be used to perform element-wise maximization of a countable set of real valued vectors, i.e. if $\{z_1, z_2, \ldots\}$ is a (finite or countably infinite) set of vectors in $\mathbb{R}^n$, one can write $\text{emax}[z_1, z_2, \ldots]$ to denote the vector containing the element-wise maximum of these vectors.

Let $\psi \in \mathbb{R}$ be the time duration that represents the *largest significant duration* of the impulse response function of any node on the grid, and let τ be the smallest multiple of h that is larger than ψ. If τ and ψ are chosen properly as we will see in Section V-A, then one can approximate the exact solution in (19) as follows

$$v(t) \approx \textstyle\sum_{q=0}^{\tau/h} \left(A^{-1}B\right)^q A^{-1} i_s(t-qh) \triangleq v_\tau(t). \tag{21}$$

For any $t \in \mathbb{R}$, let $w(t) = A^{-1}i_s(t)$ and

$$\overline{w}(t) = \underset{\theta \in [t-\tau, t]}{\text{emax}} [w(\theta)]. \tag{22}$$

As was shown in [9], we have

$$v_\tau(t) \leq G^{-1} A \overline{w}(t) \triangleq \overline{v}(t). \tag{23}$$

The waveform $\overline{v}(t)$ is effectively an *envelope waveform* representing an upper bound on the voltage drop at any time t.

When the loading currents are piecewise linear, $w(t)$ must also be piece-wise linear because $w(t)$ is the result of applying a *linear* transformation A^{-1} to $i_s(t)$. Accordingly, $w(t)$ can be found by finding $w_0 = w(t_0), w_1 = w(t_1), \ldots, w_N = w(t_N)$ using the following linear systems

$$Aw_k = i_k, \quad \forall k \in \{0, 1, \ldots, N\}. \tag{24}$$

The result would be a piece-wise linear function with the set of breakpoints $\mathcal{W} = \{(t_0, w_0), (t_1, w_1), \ldots, (t_N, w_N)\}$. Notice that for $t < 0$ and $t > t_N$, we have $w(t) = 0$.

Once $\mathcal{W}$ is found, $\overline{w}(t)$ can be computed using the $\mathrm{emax}[\cdot]$ operator applied at *every* time point $t \in \mathbb{R}$ as in (22), and so, finding $\overline{w}(t)$ *exactly* is not very practical. Instead, we proposed the following.

For any time point $t_k \in T$, let $t_{\underline{k}}$ be the largest time point in T smaller than or equal to $t_k - \tau$ if $t_k - \tau \geq t_0$, and t_0 otherwise. In other words

$$t_{\underline{k}} = \begin{cases} t_0 & \text{if } t_k - \tau < t_0; \\ \max\limits_{\substack{t \in T \\ t \leq t_k - \tau}} (\ t\ ) & \text{if } t_k - \tau \geq t_0. \end{cases} \tag{25}$$

Now, let

$$\overline{w}'_k = \begin{cases} w(t_0) & \text{if } k = 0; \\ \underset{\theta \in \{t_{\underline{k-1}}, \ldots, t_k\}}{\mathrm{emax}}[w(\theta)] & \text{if } k \in \{1, \ldots, N\}, \end{cases} \tag{26}$$

and let $\overline{w}'(t)$ be piece-wise linear with the set of breakpoints $\overline{\mathcal{W}}' = \left\{\left(t_0, \overline{w}'_0\right), \left(t_1, \overline{w}'_1\right), \ldots, \left(t_N, \overline{w}'_N\right)\right\}$, and such that

$$\overline{w}'(t) = \begin{cases} 0 & \text{if } t < 0; \\ \overline{w}'_N & \text{if } t > t_N. \end{cases} \tag{27}$$

We now present two lemmas that will help us in developing the final solution.

Lemma 1. *For every $p, q \in \{0, \ldots, N\}$, $p < q$, we have*

$$\underset{\theta \in [t_p, t_q]}{\mathrm{emax}}[w(\theta)] = \underset{\theta \in \{t_p, \ldots, t_q\}}{\mathrm{emax}}[w(\theta)]. \tag{28}$$

Proof: For every $k \in \{0, \ldots, N-1\}$, we have

$$\underset{\theta \in T_k}{\mathrm{emax}}[w(\theta)] = \mathrm{emax}[w(t_k), w(t_{k+1})] \tag{29}$$

because w is linear in $[t_k, t_k + 1]$. Therefore

$$\begin{aligned} \underset{\theta \in [t_p, t_q]}{\mathrm{emax}}[w(\theta)] &= \underset{\theta \in T_p \cup \ldots \cup T_{q-1}}{\mathrm{emax}}[w(\theta)] \\ &= \underset{k \in \{p, \ldots, q-1\}}{\mathrm{emax}} \left[\underset{\theta \in T_k}{\mathrm{emax}}[w(\theta)]\right] \\ &= \underset{k \in \{p, \ldots, q-1\}}{\mathrm{emax}} [\mathrm{emax}[w(t_k), w(t_{k+1})]] \ \text{(by (29))} \\ &= \underset{\theta \in \{t_p, \ldots, t_q\}}{\mathrm{emax}}[w(\theta)]. \end{aligned}$$

■

Lemma 1 is useful to prove Lemma 2 below. It is also useful for proving the result of section IV.

Lemma 2. *For every $t \in \mathbb{R}$, $\overline{w}(t) \leq \overline{w}'(t)$.*

Proof: The result is obvious for $t < 0$ because when $t < 0$, $\overline{w}(t) = \overline{w}'(t) = 0$. On the other hand, If $t > t_N$, then

$$\begin{aligned} \overline{w}(t) &= \underset{\theta \in [t-\tau, t]}{\mathrm{emax}}[w(\theta)] = \underset{\theta \in [t-\tau, t_N]}{\mathrm{emax}}[w(\theta)] \\ &\leq \underset{\theta \in [t_{\underline{N-1}}, t_N]}{\mathrm{emax}}[w(\theta)] \ \ \text{(because } t_{\underline{N-1}} < t_N - \tau < t - \tau) \\ &= \underset{\theta \in \{t_{\underline{N-1}}, \ldots, t_N\}}{\mathrm{emax}}[w(\theta)] \ \ \text{(due to Lemma 1)} \\ &= \overline{w}'_N = \overline{w}'(t). \end{aligned}$$

Finally, if $t_0 \leq t \leq t_N$, then there exists a value for k for which $t \in T_k$. Because $\overline{w}'(t)$ is linear for $t \in T_k = [t_k, t_{k+1}]$, then we have:

$$\overline{w}'(t) = \alpha \overline{w}'(t_k) + (1 - \alpha)\overline{w}'(t_{k+1}) \tag{30}$$

where $\alpha = \frac{t_{k+1} - t}{t_{k+1} - t_k}$. Observe that for the same value of α, and because $w(t)$ is linear between t_k and t_{k+1}, we have

$$w(t) = \alpha w(t_k) + (1 - \alpha) w(t_{k+1}). \tag{31}$$

Let

$$\overline{w}^{\dagger}(t_k) = \underset{\theta \in [t_{\underline{k}}, t_k]}{\mathrm{emax}}[w(\theta)]. \tag{32}$$

If $k > 0$, then by Lemma 1, we have

$$\underset{\theta \in [t_{\underline{k-1}}, t_k]}{\mathrm{emax}}[w(\theta)] = \underset{\theta \in \{t_{\underline{k-1}}, \ldots, t_k\}}{\mathrm{emax}}[w(\theta)] = \overline{w}'(t_k) \tag{33}$$

and

$$\underset{\theta \in [t_{\underline{k}}, t_{k+1}]}{\mathrm{emax}}[w(\theta)] = \underset{\theta \in \{t_{\underline{k}}, \ldots, t_{k+1}\}}{\mathrm{emax}}[w(\theta)] = \overline{w}'(t_{k+1}). \tag{34}$$

Because $t_{\underline{k-1}} \leq t_{\underline{k}} < t_k < t_{k+1}$, then $[t_{\underline{k}}, t_k] \subseteq [t_{\underline{k-1}}, t_k]$ and $[t_{\underline{k}}, t_k] \subset [t_{\underline{k}}, t_{k+1}]$ so that we have

$$w^{\dagger}(t_k) \leq w'(t_k) \tag{35}$$

and

$$w^{\dagger}(t_k) \leq w'(t_{k+1}). \tag{36}$$

Otherwise, if $k = 0$, then $\overline{w}^{\dagger}(t_k) = w(t_0) = \overline{w}'(t_k) \leq \overline{w}'(t_{k+1})$, and so the above inequalities (35) and (36) are also true. Accordingly, $\forall k \in \{0, \ldots, N\}$, we have

$$\overline{w}^{\dagger}(t_k) \leq \alpha \overline{w}'(t_k) + (1 - \alpha)\overline{w}'(t_{k+1}) = \overline{w}'(t). \tag{37}$$

Now, observe that we can write $[t - \tau, t] = [t - \tau, t_k] \cup [t_k, t]$, and hence

$$\overline{w}(t) = \underset{\theta \in [t-\tau, t]}{\mathrm{emax}}[w(\theta)] = \mathrm{emax}\left[\underset{\theta \in [t-\tau, t_k]}{\mathrm{emax}}[w(\theta)], \underset{\theta \in [t_k, t]}{\mathrm{emax}}[w(\theta)]\right]. \tag{38}$$

But $[t - \tau, t_k] \subseteq [t_{\underline{k}}, t_k]$, hence (38) becomes

$$\begin{aligned} \overline{w}(t) &\leq \mathrm{emax}\left[\underset{\theta \in [t_{\underline{k}}, t_k]}{\mathrm{emax}}[w(\theta)], \underset{\theta \in [t_k, t]}{\mathrm{emax}}[w(\theta)]\right] \\ &= \mathrm{emax}\left[\overline{w}^{\dagger}(t_k), \underset{\theta \in [t_k, t]}{\mathrm{emax}}[w(\theta)]\right]. \end{aligned} \tag{39}$$

Using the fact that w is linear in $[t_k, t]$, we have

$$\underset{\theta \in [t_k, t]}{\mathrm{emax}}[w(\theta)] = \mathrm{emax}\left[w(t_k), w(t)\right] \tag{40}$$

Therefore, (39) implies

$$\overline{w}(t) \leq \mathrm{emax}\left[\overline{w}^{\dagger}(t_k), w(t_k), w(t)\right]. \tag{41}$$

But $w(t_k) \leq \overline{w}^{\dagger}(t_k) = \underset{\theta \in [t_{\underline{k}}, t_k]}{\mathrm{emax}}[w(\theta)]$, and hence (41) gives

$$\begin{aligned} \overline{w}(t) &\leq \mathrm{emax}\left[\overline{w}^{\dagger}(t_k), \overline{w}^{\dagger}(t_k), \alpha w(t_k) + (1 - \alpha) w(t_{k+1})\right] \\ &= \mathrm{emax}\left[\overline{w}^{\dagger}(t_k), \alpha w(t_k) + (1 - \alpha) w(t_{k+1})\right], \end{aligned} \tag{42}$$

where we have used (31) to replace $w(t)$ in (41).

Now, because $w(t_k) \leq \overline{w}'(t_k)$ and $w(t_{k+1}) \leq \overline{w}'(t_{k+1})$, we can obtain from (42)

$$\overline{w}(t) \leq \mathrm{emax}\left[\overline{w}^{\dagger}(t_k), \alpha \overline{w}'(t_k) + (1 - \alpha)\overline{w}'(t_{k+1})\right], \tag{43}$$

which allows us to write using (37) and (30)

$$\overline{w}(t) \leq \mathrm{emax}\left[\overline{w}'(t), \overline{w}'(t))\right] = \overline{w}'(t).$$

■

Algorithm 1 GENERATE_TRANSIENT_ENVELOPES

Input: G, C, $\mathcal{I}$, τ
Output: Envelope waveform $\overline{\mathcal{V}}'$

1: Solve $Aw_0 = i_0$ for w_0, and set $\overline{w}'_0 = w_0$
2: **for** $k = 1, \ldots, N$ **do**
3: Solve $Aw_k = i_k$ for w_k
4: Set $t_{\underline{k-1}} = \begin{cases} t_0 & \text{if } t_k - \tau < t_0 \\ \max\limits_{\substack{t \in T \\ t \le t_{k-1} - \tau}} (\ t\ ) & \text{if } t_k - \tau \ge t_0 \end{cases}$
5: Set $\overline{w}'_k = \max\limits_{\theta \in \{t_{\underline{k-1}}, \ldots, t_k\}} [w(\theta)]$
6: Solve $G\overline{v}'_k = A\overline{w}'_k$ for $\overline{v}'_k$
7: **end for**
8: **return** $\overline{\mathcal{V}}' = \left\{ \left(t_0, \overline{v}'_0\right), \left(t_1, \overline{v}'_1\right), \ldots, \left(t_N, \overline{v}'_N\right) \right\}$

Let I_n be the $n \times n$ identity matrix. Using the result of Lemma 2, and the fact that $G^{-1}A = G^{-1}\left(G + \frac{C}{h}\right) = I_n + G^{-1}\frac{C}{h} \ge 0$ since $G^{-1} \ge 0$ and $C \ge 0$, we can write using (23)

$$v_\tau(t) \le G^{-1}A\overline{w}(t) \le G^{-1}A\overline{w}'(t) \triangleq \overline{v}'(t), \tag{44}$$

which means that $\overline{v}'(t)$ is an envelope waveform representing an upper bound on the voltage drop at any time t. Because $\overline{w}'(t)$ is piece-wise linear, $\overline{v}'(t)$ must also be piece-wise linear because $\overline{v}'(t)$ is the result of applying a linear transformation $G^{-1}A$ to $\overline{w}'(t)$. Accordingly, one can find $\overline{v}'(t)$ by finding $\overline{v}'_0 = \overline{v}'(t_0)$, $\overline{v}'_1 = \overline{v}'(t_1)$ …, $\overline{v}'_N = \overline{v}'(t_N)$ by solving the following set of linear systems

$$G\overline{v}'_k = A\overline{w}'_k, \quad \forall k \in \{0, 1, \ldots, N\}, \tag{45}$$

so that the result is piece-wise linear with the set of breakpoints

$$\overline{\mathcal{V}}' = \left\{ \left(t_0, \overline{v}'_0\right), \left(t_1, \overline{v}'_1\right), \ldots, \left(t_N, \overline{v}'_N\right) \right\}. \tag{46}$$

Notice that for $t < 0$, we have $\overline{v}'(t) = 0$ and for $t > t_N$, we have $\overline{v}'(t) = \overline{v}'(t_N)$.

Algorithm 1 presents a high level description of how to find the envelope waveform $\overline{v}'(t)$. The algorithm requires $2N + 2$ system solves which can be done using a Cholesky factorization [13] of each of A and G. Cholesky factorization can be used because both A and G are positive definite matrices.

IV. DC Envelope Waveforms

In many cases, simple DC bounds on the true voltage drop waveforms are sufficient for power grid dynamic verification. In this section, we will show how such DC bounds can be generated and used for verification. These bounds turn out to be easier to compute.

Recall that $w(t) = A^{-1}i_s(t)$, and consider the vector

$$\overline{W} = \max_{\theta \in \mathbb{R}} [w(\theta)]. \tag{47}$$

As was shown in [9],

$$v(t) \le G^{-1}A\overline{W} \triangleq \overline{V}. \tag{48}$$

Effectively, $\overline{V}$ is a DC envelope waveform representing an upper bound on the voltage drop at any time t.

In the case of piece-wise linear loading currents, we first find $w(t)$ as in (24), and then we compute

$$\overline{W}' = \max_{\theta \in T} [w(\theta)]. \tag{49}$$

Algorithm 2 GENERATE_DC_ENVELOPES

Input: G, C, $\mathcal{I}$
Output: DC envelope $\overline{V}'$

1: **for** $k = 0, \ldots, N$ **do**
2: Solve $Aw_k = i_k$ for w_k
3: **end for**
4: $\overline{W}' = \max\limits_{\theta \in T} [w(\theta)]$
5: Solve $G\overline{V}' = A\overline{W}'$ for $\overline{V}'$
6: **return** $\overline{V}'$

Using Lemma 1 with $p = 0$ and $q = N$, and because $w(t) = 0$ for $t < t_0$ and $t > t_N$, it should be clear that $\overline{W} = \overline{W}'$. Therefore, a good way of finding DC bounds is by computing $\overline{W}'$ using (49), and then solving

$$G\overline{V}' = A\overline{W}' \tag{50}$$

for $\overline{V}'$, which is equal to $\overline{V}$ because $\overline{W} = \overline{W}'$.

Algorithm 2 presents a high level description of how to find the DC envelope $\overline{V}'$. The algorithm requires $N + 1$ system solves which can be done, as before, using a Cholesky factorization of each A and G.

V. Implementation Details

A. Computing the Duration ψ

The duration ψ is a parameter that depends on the dynamics of the power grid as a linear system. For any value of ψ, the exact solution, at any time t, of the differential equation (3) is

$$v(t) = e^{-C^{-1}G\psi}v(t - \psi) + \int_{t-\psi}^{t} e^{-C^{-1}G(t-\nu)}C^{-1}i_s(\nu)d\nu. \tag{51}$$

For (21) to be accurate, ψ should be chosen such the term $e^{-C^{-1}G\psi}v(t - \psi)$ in (51) is *negligible*. If this is true, then $v(t)$ can be approximated by the integral term of (51), which only requires information about the input in the duration $[t - \psi, t]$. The error vector introduced by this approximation is

$$e(t) = e^{-C^{-1}G\psi}v(t - \psi). \tag{52}$$

Let $\eta > 0$ be a user-defined error tolerance on the voltage drop at every node. If ψ is chosen such that

$$\|e(t)\|_\infty \le \eta, \tag{53}$$

then this guarantees that every entry of the integral term is at most η away from the corresponding entry of $v(t)$. The lemma below, first introduced in [9], shows how to choose ψ to guarantee $\|e(t)\|_\infty \le \eta$. It requires an upper bound Υ on $\|v(t)\|_2$ for all t which can be obtained from the DC envelope waveforms of section IV or using the simple bound v_{dd} on the voltage drop of each node. Note that the matrix $C^{-1}G$ has real positive eigenvalues because both C^{-1} and G are symmetric [14].

Lemma 3. *Let $\lambda_{\min}$ be the smallest eigenvalue of $C^{-1}G$, and $c_{\max}$ and $c_{\min}$ be the largest and the smallest diagonal elements of C, respectively. If ψ is chosen such that*

$$\psi \ge \frac{1}{\lambda_{\min}} \ln\left(\frac{\sqrt{c_{\max}/c_{\min}}}{\eta/\Upsilon}\right), \tag{54}$$

then

$$\|e(t)\|_\infty \le \eta. \tag{55}$$

Algorithm 3 FIND_ψ

Input: G, C, η, ϵ
Output: ψ

$x_0 = [1\ 1\ \ldots\ 1] \in \mathbb{R}^n$, $k = 0$
Solve $Gx_1 = Cx_0$ for x_1
$\lambda_d^{(1)} = \frac{x_1^T x_0}{x_0^T x_0}$
while $\left(\left|\frac{\lambda_d^{(k)} - \lambda_d^{(k-1)}}{\lambda_d^{(k-1)}}\right| \geq \epsilon\right)$ **do**
  $k = k + 1$
  Solve $Gx_{k+1} = Cx_k$ for x_{k+1}
  $\lambda_d^{(k)} = \frac{x_{k+1}^T x_k}{x_k^T x_k}$
end while
$\lambda_{\min} = \frac{1}{\lambda_d^{(k)}}$
Find $c_{\min}$, $c_{\max}$, and Υ
return $\frac{1}{\lambda_{\min}} \ln\left(\frac{\sqrt{c_{\max}/c_{\min}}}{\eta/\Upsilon}\right)$

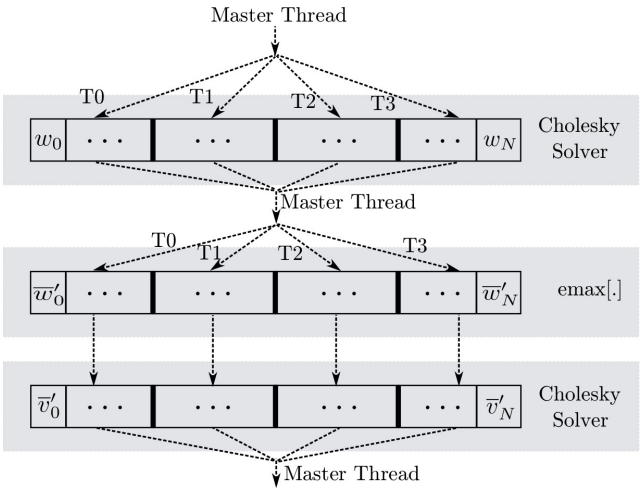

Figure 3: A parallel implementation of Algorithm 1

Proof: The proof requires the notion of the *Logarithmic norm* [15] of a square matrix X with respect to the 2-norm, defined as

$$\mu_2(X) = \max\left\{\lambda | \lambda \in \sigma\left(\frac{X^T + X}{2}\right)\right\} \tag{56}$$

where $\sigma\left(\frac{X^T+X}{2}\right)$ is the set of eigenvalues of $\frac{X^T+X}{2}$.

We also require the notion of the *square root of a matrix* defined as follows. For any matrix X, the square root of X, denoted $\sqrt{X}$ is a matrix satisfying $\sqrt{X}\sqrt{X} = X$ [10]. The square root of a matrix may or may not exist. If X is a diagonal matrix, then $\sqrt{X}$ exists and is a diagonal matrix with diagonal elements equal to the square root of the diagonal elements of X. Let

$$S \triangleq -\sqrt{C^{-1}}G\sqrt{C^{-1}}, \tag{57}$$

and notice that

$$-C^{-1}G = \sqrt{C^{-1}}(-\sqrt{C^{-1}}G\sqrt{C^{-1}})\sqrt{C} = \sqrt{C^{-1}}S\sqrt{C} \tag{58}$$

because $\sqrt{C^{-1}}\sqrt{C} = I_n$. In essence, the matrices $-C^{-1}G$ and S are related by a *similarity* transformation. This implies that $\sigma(-C^{-1}G) = \sigma(S)$ and $e^{-C^{-1}G\psi} = \sqrt{C^{-1}}e^{S\psi}\sqrt{C}$ [10]. Accordingly,

$$\begin{aligned}\|e(t)\|_\infty &= \|e^{-C^{-1}G\psi}v(t-\psi)\|_\infty \\ &= \|\sqrt{C^{-1}}e^{S\psi}\sqrt{C}v(t-\psi)\|_\infty \\ &\leq \|\sqrt{C^{-1}}e^{S\psi}\sqrt{C}v(t-\psi)\|_2 \\ &\leq \|\sqrt{C^{-1}}\|_2\|e^{S\psi}\|_2\|\sqrt{C}\|_2\|v(t-\psi)\|_2. \end{aligned} \tag{59}$$

Clearly,

$$\|v(t-\psi)\|_2 \leq \Upsilon. \tag{60}$$

Also, because C is a positive diagonal matrix, and because the 2-norm of a diagonal matrix is the entry with the largest magnitude [10], then $\|\sqrt{C}\|_2 = \sqrt{c_{\max}}$. Similarly, $\|\sqrt{C^{-1}}\|_2 = \sqrt{\frac{1}{c_{\min}}}$. Furthermore, the following is true from [15]

$$\|e^{S\psi}\|_2 \leq e^{\mu_2(S)\psi}. \tag{61}$$

But G is an $\mathcal{M}$ matrix, i.e. a symmetric positive definite matrix, and $\sqrt{C^{-1}}$ is a diagonal matrix. Therefore, $\sqrt{C^{-1}}G\sqrt{C^{-1}}$ is symmetric, meaning

$$\sigma\left(\frac{S^T + S}{2}\right) = \sigma(S) = \sigma(-C^{-1}G). \tag{62}$$

Hence,

$$\begin{aligned}\mu_2(S) &= \max\left\{\lambda | \lambda \in \sigma(-C^{-1}G)\right\} \\ &= -\min\left\{\lambda | \lambda \in \sigma(C^{-1}G)\right\} = -\lambda_{\min} \leq 0.\end{aligned}$$

Accordingly, using (54), we have

$$\mu_2(S)\psi = -\lambda_{\min}\psi \leq -\ln\left(\frac{\sqrt{c_{\max}/c_{\min}}}{\eta/\Upsilon}\right) \tag{63}$$

$$= \ln\left(\frac{\eta/\Upsilon}{\sqrt{c_{\max}/c_{\min}}}\right), \tag{64}$$

and combining (59), (60), (61), and (64), we get

$$\|e(t)\|_\infty \leq \sqrt{\frac{1}{c_{\min}}}\frac{\eta/\Upsilon}{\sqrt{c_{\max}/c_{\min}}}\sqrt{c_{\max}}\Upsilon = \eta, \tag{65}$$

which completes the proof. ■

Computing $\lambda_{\min}$ can be easily done using the *power method* [13] as was proposed in [14]. The full procedure is shown in Algorithm 3. Effectively, the algorithm computes the *dominant* eigenvalue of the matrix $G^{-1}C$, which is the same as its largest eigenvalue because all the eigenvalues of $G^{-1}C$ are real and positive. The smallest eigenvalue of $C^{-1}G$ is simply the inverse of the result. Moreover, the algorithm requires a tolerance ϵ on the relative error between two consecutive estimates of $\lambda_{\min}$.

B. Time-Step

The choice of the time step h is done based on the algorithm proposed in [14]. The algorithm requires $\lambda_{\min}$ which is already being computed to find ψ. The resulting time step is simply $h = \frac{1}{\lambda_{\min}}$. This choice of h was proven to be useful in the case of *vectorless* RC verification in [14] where we have used a similar upper bound to the one we are using in this paper. As we will see in the results section, the same choice of h turns out to be adequate in this context as well.

VI. Parallelization

Observing Algorithms 1 and 2, one can notice that the most expensive operations are solving the linear systems. However, one can also notice that, unlike traditional transient simulation approaches, the linear systems in our approaches can be solved *independently*. This is a key point because it implies that our algorithms are *embarrassingly parallel*. Thus, we implement multi-threaded versions of our algorithms. Fig 3 shows how Algorithm 1 can be parallelized. Algorithm 2 can be parallelized in a similar fashion. In the figure, The master thread branches off into 4 threads T0, T1, T2, and T3 where each thread takes care of a portion of the input. The threads branch off and merge appropriately as to respect data dependencies.

In the first step, each thread uses the Cholesky solver to compute a portion of the w_k's. Once all the w_k's are computed, the master thread collects the results and combines them into a big set of vectors. Given the parameter τ and the set of vectors generated, each thread will then compute a portion of the $\overline{w}'_k$'s using the $\mathrm{emax}[\cdot]$ operator. Finally, the Cholesky solver is used once again by each thread to compute the $\overline{v}'_k$'s by solving $G\overline{v}'_k = A\overline{w}'_k$.

VII. Experimental Results

We implement all the algorithms proposed in C++. All the linear systems are solved using CHOLMOD [16] from SuiteSparse [17]. The tested grids were generated based on user-defined industrial geometry parameters consistent with 1V 45nm CMOS technology. The sizes of the grids generated range from 250K to 4M nodes with around 20% of the nodes attached to current sources. The input current waveforms are piece-wise linear with breakpoints that are generated randomly. All the computations were done on a 3.4GHz quad-core Linux machine with 32GB of RAM.

In order to verify the accuracy of our approaches, we compare the resulting envelopes with the exact voltage drop waveforms resulting from performing the traditional trapezoidal (TR) method with a fixed time step. We understand that the traditional TR method is not the most efficient method out there. However, we decided to compare our method to TR for two reasons: 1) TR provides a baseline that is easy to implement and that every method out there can compare to. Knowing the speedups with respect to TR, one can easily figure out the relative speedups. 2) The runtime of other published methods is only meaningful on the specific machines that were used to simulate them. Comparing our data to their data is not particularly fair because we are running our simulations on a different platform.

The current waveforms are chosen such that the minimum distance between two consecutive breakpoints is 10ps, and so, the time step used to compute the exact solution is also chosen to be 10ps. The duration of the simulation is set to 100ns, so there are 10,000 pairs of forward and backward substitutions that have to be done during the process of exact transient simulation. The number of breakpoints is much smaller than that as the distance between two consecutive breakpoints is chosen randomly between 10ps and 1ns. It should be noted that using a *dynamic time step* for the TR method (adaptive stepping) leads to multiple matrices that need to be factorized, thus degrading the performance of the TR method. For that reason, a fixed time step is used.

In Table I we report the maximum and average overestimation errors of the DC envelopes as compared to the peaks of the exact voltage drop waveforms. In the table, "TR", "DC-ENV", and "TRAN-ENV" refer to the trapezoidal method, the DC envelope method, and the transient envelope method respectively. We observe that the overestimation is always below 1mV with an average that is never exceeding 300μV. Moreover, it can be shown that **the DC envelopes are upper bounds on the transient envelopes**, and so, the transient envelopes will also capture the peaks of the exact voltage drop quite accurately. To make this clearer, Fig 4 shows the exact voltage drop waveform and the resulting envelope waveforms for node indexed 20,000 in G1. We can observe how well the transient envelope waveform follows the peaks of the exact waveform, making it very useful for verification purposes. Using the transient envelope, one can figure out if a given node is safe or not. If the node is not safe, the envelope waveform provides the user an idea of the time intervals at which the node is becoming unsafe. This could potentially be very useful for debugging.

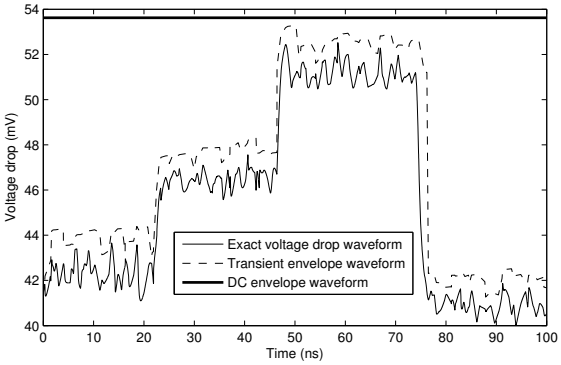

Figure 4: Envelope waveform at node indexed 20,000 in G1

In terms of speed, we first compare the runtime of the *sequential* implementation of our approaches to the TR method. For each of the three methods (TR, DC-ENV, and TRAN-ENV), the runtime is divided into two portions: DC and transient. The DC portion represents all the fixed cost that has to be done no matter how long the simulation is. For example, this includes any matrix computations and factorizations. The speedups reported are computed using the second portion of the runtime (tran) of each method for the given current waveforms because the fixed costs for the three methods is roughly the same for a given grid. We can see how well our algorithms perform in comparison with the TR method as the speedups can reach 228.79× for DC-ENV and 75.78× for TRAN-ENV. This shows that our approaches are quite practical and present significant performance advantages over the traditional TR method.

Multi-threaded implementations of our algorithms are also implemented in C++ using OpenMP with 4 threads. The implementations were tested on the quad-core machine described earlier. The overall runtimes and speedups in comparison with TR are reported in Table II. In the table, "MT-DC-ENV" and "MT-TRAN-ENV" refer to the multi-threaded versions of DC-ENV and TRAN-ENV respectively. The speedups over the sequential implementations are up to 3.26× for MT-DC-ENV and 2.42× for MT-TRAN-ENV. The overall speedups obtained are up to 732.87× for MT-DC-ENV and 181.59× for MT-TRAN-ENV. Notice that the parallelism invoked is across time. This cannot be done for TR due to the inherent data dependencies that exist in the recurrence (10).

Table I: Speed and Accuracy of the Sequential Implementations of the Proposed Approaches

Grid			TR		DC-ENV					TRAN-ENV		
Name	Nodes	ψ (ns)	DC(s)	Tran(s)	DC(s)	Tran(s)	Speedup	Maximum Error (μV)	Average Error (μV)	DC (s)	Tran (s)	Speedup
G1	251,122	30.15	1.45	425.32	1.67	2.22	**191.59**×	598	221	1.61	6.99	**60.85**×
G2	506,488	12.67	3.73	946.16	4.25	4.58	**206.56**×	735	272	4.18	12.69	**74.56**×
G3	1,007,064	13.13	9.57	2,038.45	10.68	9.51	**214.35**×	778	215	10.6	27.03	**75.41**×
G4	2,030,668	13.32	24.47	4,482.43	26.93	20.72	**216.33**×	774	187	26.59	59.15	**75.78**×
G5	4,022,448	13.77	63.25	9,593.28	69.54	41.93	**228.79**×	736	159	68.73	128.06	**74.91**×

Table II: Performance of the Multi-threaded Implementations of the Proposed Approaches

Grid	MT-DC-ENV			MT-TRAN-ENV		
	Tran (s)	Speedup vs. DC-ENV	Speedup v.s. TR	Tran (s)	Speedup vs. TRAN-ENV	Speedup vs. TR
G1	0.69	**3.21**×	**616.41**×	3.33	**2.10**×	**127.72**×
G2	1.45	**3.16**×	**652.52**×	5.92	**2.14**×	**159.82**×
G3	2.92	**3.26**×	**698.10**×	12.26	**2.20**×	**166.27**×
G4	6.41	**3.23**×	**699.29**×	25.82	**2.29**×	**173.60**×
G5	13.09	**3.20**×	**732.87**×	52.83	**2.42**×	**181.59**×

VIII. Conclusion

In this paper, we proposed a parallel simulation-based approach for checking RC power grids. The algorithms we proposed generate envelope upper bound waveforms on the exact voltage drop waveforms. The overestimation of the envelopes was shown to be minimal and the speedup over the traditional trapezoidal approach was shown to be dramatic. The generated envelopes can be very useful for checking the safety of the grid, as well as for debugging purposes in the cases where particular nodes were found to be unsafe.

References

[1] F. N. Najm, *Circuit Simulation*. Hoboken, NJ: John Wiley & Sons, Inc, 2010.

[2] Z. Zeng, T. Xu, Z. Feng, and P. Li, "Fast static analysis of power grids: Algorithms and implementations," in *ACM/IEEE International Conference on Computer-Aided Design (ICCAD-12)*, San Jose, CA, Nov. 7-10 2011.

[3] C.-H. Chou, N.-Y. Tsai, H. Yu, C.-R. Lee, Y. Shi, and S.-C. Chang, "On the preconditioner of conjugate gradient method - A power grid simulation perspective," in *ACM/IEEE International Conference on Computer-Aided Design (ICCAD-11)*, San Jose, CA, Nov. 7-10 2011.

[4] H. Qian, S. R. Nassif, and S. S. Sapatnekar, "Power grid analysis using random walks," *IEEE Trans. on Computer-Aided Design (TCAD)*, vol. 24, no. 8, August 2005.

[5] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "A multigrid-like technique for power grid analysis," *IEEE Trans. on Computer-Aided Design (TCAD)*, vol. 21, no. 10, Oct. 2002.

[6] M. Zhao, R. V. Panda, S. S. Sapatnekar, and D. Blaauw, "Hierarchical analysis of power distribution networks," *IEEE Trans. on Computer-Aided Design (TCAD)*, vol. 21, no. 2, Feb. 2002.

[7] X. Xiong and J. Wang, "Parallel forward and back substitution for efficient power grid simulation," in *ACM/IEEE International Conference on Computer-Aided Design (ICCAD-12)*, San Jose, CA, Nov. 5-8 2012.

[8] H. Zhuang, S.-H. Weng, J.-H. Lin, and C.-K. Cheng, "MATEX: A distributed framework for transient simulation of power distribution networks," in *ACM/IEEE Design Automation Conference (DAC-14)*, San Francisco, CA, June 1-5 2014.

[9] M. Fawaz and F. N. Najm, "Fast simulation-based verification of RC power grids," in *IEEE Canadian Conference on Electrical and Computer Engineering*, Vancouver, Canada, May 15-18 2016.

[10] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge University Press, 2012.

[11] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. Chichester, U.K.: Wiley, 1991.

[12] I. A. Ferzli, F. N. Najm, and L. Kruse, "A geometric approach for early power grid verification using current uncertainties," in *ACM/IEEE International Conference on Computer-Aided Design (ICCAD-07)*, San Jose, CA, Novermber 5-8 2007.

[13] Y. Saad, *Iterative methods for sparse linear systems*, 2nd ed. SIAM, 2003.

[14] M. Fawaz and F. N. Najm, "Accurate verification of RC power grids," in *IEEE Design, Automation and Test in Europe (DATE-16)*, Dresden, Germany, March 14-18 2016.

[15] G. Söderlind, "The Logarithmic norm. History and modern theory," *BIT Numerical Mathematics*, vol. 46, no. 3, 2003.

[16] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate," *ACM Trans. on Mathematical Software*, vol. 35, no. 3, pp. 22:1–22:14, 2008.

[17] T. A. Davis. Suitesparse 4.4.6. [Online]. Available: http://faculty.cse.tamu.edu/davis/suitesparse.html