Fast Vectorless RLC Grid Verification

Mohammad Fawaz, *Student Member, IEEE*, and Farid N. Najm, *Fellow, IEEE*

***Abstract*—Checking the power distribution network of an integrated circuit must start early in the design process, when changes to the grid can be more easily implemented. Vectorless verification is a technique that achieves this goal by demanding limited information about the currents drawn from the grid. State of the art techniques that deal with RLC grids become prohibitive even for medium size grids. In this paper, we propose a novel technique that estimates the worst-case voltage fluctuations for RLC grids by carefully selecting the time step, in a way that significantly reduces the number of linear programs that need to be solved, and eliminates the need for other expensive computations, like dense matrix–matrix multiplications. Results show that our technique is accurate and scalable for large grids as it achieves over 19× speedup over existing methods.**

***Index Terms*—Linear programming, overshoot, power grid, RLC, undershoot, verification.**

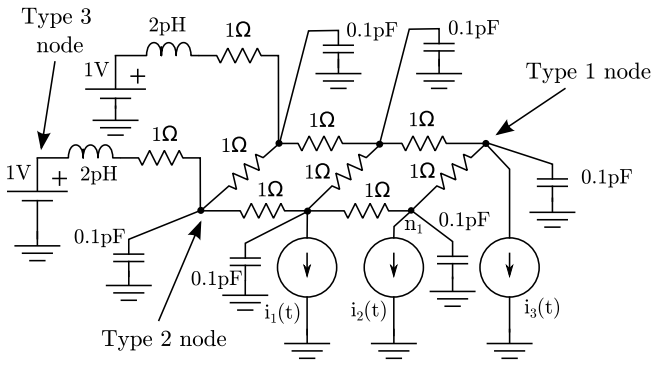

Fig. 1. Small RLC power grid.

I. Introduction

VERIFICATION of the on-die power grid has become a critical step in modern VLSI design. Recent advances in nanotechnology have led to a significant decrease in feature size and voltage levels, as well as to an increase in operating frequency, thus creating a substantial amount of IR drops and Ldi/dt noise across the chip. This is worrying because large voltage fluctuations can cause timing violations, and, in some cases, may create logic hazards.

Typically, a power grid is modeled as a large passive network of resistors, capacitors, and inductors. The topology of this network and the values of its elements are usually determined based on the geometrical parameters of the on-die metal lines and on the grid-package interconnections. When an RLC model of the grid is assumed, both overshoots and undershoots can arise at grid nodes. Thus, engineers must verify that fluctuations in both directions are within a certain user-defined safety threshold.

Today, most power grid verification tools are based on simulation. These tools assume that the grid is loaded by a certain set of transient current waveforms, and compute the resulting voltage fluctuations using direct circuit solving techniques. One must repeat the process several times for different sets of current traces until the grid is ensured to be safe with a certain level of confidence. State of the art simulation-based tools are very efficient and can handle industrial level power girds. Traditional approaches use the standard LU factorization, Cholesky factorization [1], or the preconditioned conjugate gradient (CG) method [2]. Other tools use random walks [3], [4] and multigrid techniques [5]. Hierarchical approaches are proposed in [6]. Also, and as part of TAU 2012 power grid simulation contest, Yang *et al.* [7] and Yu and Wong [8] have been proposed. Parallelization of forward/backward substitution is proposed in [9]. A solution utilizing the matrix exponential kernel is proposed in [10]. However, all these tools suffer from two main problems. First, it is not practical to expect the user to provide the exact currents drawn by the underlying logic blocks, because these currents depend on how the chip is being used and on which blocks are in operation at every point in time. Covering all possible scenarios is clearly impossible, and thus, the user can only simulate the grid for few sets of current traces that are deemed to be representative of worst-case behavior. This is a difficult process that is prone to all kinds of uncertainties and error. Second, verifying the grid must begin at early stages of the design process where only limited information about the logic circuitry is known, and where grid modifications can be more easily incorporated. However, simulation-based tools are not very useful early in the design flow because specifying the current traces requires detailed information about the position and the power dissipation of every logic block.

The problems of simulation-based tools are amplified when the inductance in the grid is considered because, when analyzing RLC power grid, only certain temporal arrangements of the loading currents may lead to an overshoot at certain nodes in the grid. For example, consider the small RLC network in Fig. 1. The circuit is simulated using HSPICE under two different sets of current waveforms, and the resulting voltage

Manuscript received October 9, 2015; revised March 6, 2016 and June 16, 2016; accepted June 28, 2016. Date of publication July 11, 2016; date of current version February 16, 2017. This work was supported by the Natural Sciences and Engineering Research Council of Canada. This paper was recommended by Associate Editor A. Srivastava.

The authors are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: mohammad.fawaz@mail.utoronto.ca; f.najm@utoronto.ca).

Digital Object Identifier 10.1109/TCAD.2016.2589899

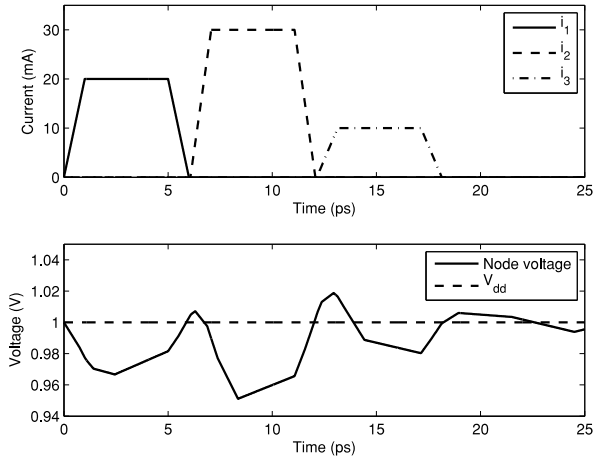

Fig. 2. Current configuration resulting in a large overshoot.

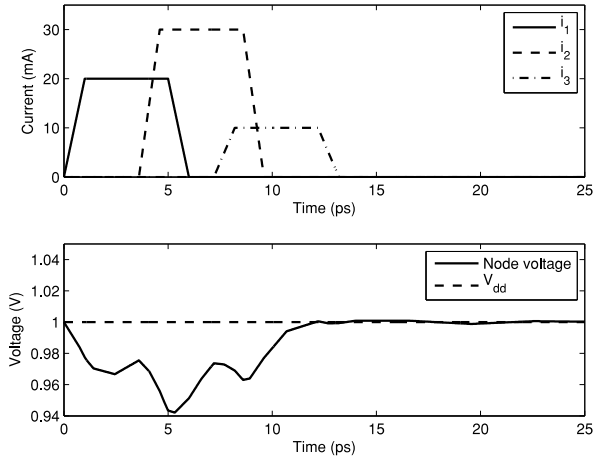

Fig. 3. Current configuration resulting in a small overshoot.

waveforms at node n_1 are shown in Figs. 2 and 3. In the first experiment, we observe large overshoots where the node voltage rises almost 20 mV above V_{dd}. In the second experiment, where the loading currents are only slightly modified, almost no overshoot is observed. The point is that it is expensive for the user to use trial and error to find out which situations lead to critical overshoots in the grid. The process becomes very complicated for grids with large numbers of current sources.

To overcome these problems, the constraint-based framework for verification was first introduced in [11]. The main advantage of this framework is that it allows for vectorless power grid verification which can be done early in the design process. As opposed to vector-based techniques, vectorless verification requires only a limited amount of information about the underlying logic circuit, in the form of current constraints. Given the the user-specified constraints, one can find the worst-case voltage fluctuations at every node in the grid, and compare them to the user-defined thresholds. Vectorless verification is well established for R and RC grids, and several approaches have been developed in the past few years to make the process as efficient and as accurate as possible. Approximate inverse techniques were used in [12] to reduce the size of the linear program (LP) problem for each node, and [13] developed a hierarchical matrix inverse algorithm for the same purpose, while [14] proposed a dual algorithm to speed-up the LP solution. Moreover, dominance relations among node voltage drops were exploited in [15] to reduce the number of LPs to be solved. To reduce the size of the grid, several model order reduction approaches were suggested such as in [16] and [17].

In the realm of RLC verification, Ghani and Najm [18] developed a technique to compute the worst-case voltage fluctuations using an iterative process that requires solving a large number of LPs while stepping through time. Later, a more efficient technique was developed in [19], which computes bounds on the worst-case fluctuations and which was shown to be more efficient than [18]. Xiong and Wang [20], [21] proposed a vectorless framework for RLC grids under transient constraints. Unfortunately, all the above approaches are highly inefficient as they require several weeks to verify a moderate size RLC grid.

In this paper, we improve upon the work of [19]. We show how a careful choice of the time step Δt leads to a new expression of the bounds that is much easier to compute and in fact more accurate than that of [19]. Basically, we show how to choose Δt to guarantee that a certain parameter r from [19] is equal to 1. This is very useful, as we will see later, because it reduces the number of LPs to be solved to 2 per node or per inductive branch. It also eliminates the need for the full inverse (only a subset of the columns is now required), and for any dense matrix–matrix multiplications. Moreover, the fact that r is not varying anymore means that the behavior of the algorithm is more predictable. This allowed us to obtain up to $32\times$ speed up over the algorithm of [19], with a bound that is less than 1 mV away from the exact solution.

The rest of this paper is organized as follows. Sections II and III provide some background material and notation. Section IV presents the problem definition and the exact solution of the vectorless RLC problem. Section V derives the bounds on the exact solution while Section VI shows how the time step must be chosen to guarantee $r = 1$. Finally, implementation details and experimental results and presented in Sections VII and VIII, while Section IX concludes this paper.

II. Background

A. Power Grid Model

The power grid is a large full-chip structure of connected metal lines, across multiple layers interconnected through vias and connected by C4 bumps to wiring in the package and on the board. Typically, a power grid is modeled as a linear circuit composed of a large number of lumped linear (RLC) elements.

Consider an RLC grid in which there are three types of nodes: 1) some nodes are connected to ideal current sources to ground, in parallel with capacitors to ground; 2) some (most) nodes are connected to resistors or inductors to other grid nodes and capacitors to ground; and 3) some nodes are connected to resistors or inductors to other grid nodes and ideal voltage sources to ground. Fig. 1 shows examples of each type of node. Note that in this paper, mutual inductances and branch

capacitances are ignored. That is, only self inductances are considered and all capacitances are assumed to be connected to ground. The current sources (with their parallel capacitors) represent the currents drawn by the logic circuits tied to the grid at these nodes. The ideal voltage sources represent the external voltage supply, V_{dd}. Excluding the ground node, let the power grid consist of $n_v + q$ nodes, where nodes $\{1, \ldots, n_v\}$ are the nodes not connected to a voltage source, while the remaining nodes $(n_v + 1), (n_v + 2), \ldots, (n_v + q)$ are the nodes where the q voltage sources are connected. Let m be the number of current sources connected to the grid, whose positive (reference) direction of current is from node-to-ground, assumed to be connected at nodes $1, 2, \ldots, m \leq n_v$, and let $i(t) \geq 0$ be the $m \times 1$ vector of all source currents. Also, let H be an $n_v \times m$ matrix of 0 and 1 entries that identifies (with a 1) which node is connected to which current source. Finally, let n_l be the number of inductors in the grid. It should be noted that n_l is typically much smaller than n_v. The reason is that Ldi/dt noise is mostly due to the inductance of interconnections between the grid and the package [22], and so, the number of inductors in a typical RLC model of the grid is in the order of the number of the C4 bumps, which is much smaller than the total number of nodes in the grid. This fact will become more apparent when we consider the IBM power grid benchmarks [23] in the experimental results section.

As in modern circuit simulators, such as HSPICE [24], we assume that the grid does not contain any purely inductive loops. In other words, for any node k, there is no path in the grid from k leading back to k that consists of only (ideal) inductors.

B. Constraints-Based Framework

As was done in [19], we use current constraints to capture the uncertainty about the circuit currents, arising from both unknown circuit behaviors or unknown circuit details early in the design flow. Two types of constraints are defined: 1) local constraints and 2) global constraints. Local constraints are upper and lower bounds on individual current sources, where a current source can represent a single logic gate or cell, but more typically might represent a larger block. They can be expressed as

$$i_{lb} \leq i(t) \leq i_{ub}, \quad \forall t \in \mathbb{R} \tag{1}$$

where i_{ub} and i_{lb} are $m \times 1$ vectors of the maximum and minimum values that the current sources can draw, respectively. Global constraints represent the maximum and minimum total power dissipation of a group of circuit cells or blocks. Assuming we have a total of κ global constraints, they can be expressed in matrix form as

$$i_g \leq Ui(t) \leq i_G, \quad \forall t \in \mathbb{R} \tag{2}$$

where U is a $\kappa \times m$ matrix that consists only of 0s and 1s which indicates (with a 1) which current sources are present in each global constraint. Together, the local and global constraints define a feasible space of currents, which we denote by $\mathcal{F}$, so that $i(t) \in \mathcal{F}$ for every $t \in \mathbb{R}$ if and only if it satisfies both (1) and (2).

III. Notation

We use the notation y_i to denote the ith entry of any vector y (unless defined otherwise) and J_{ij} to denote the (i, j)th entry of any matrix J. We will also use the notation $J \geq 0$ to indicate that $J_{ij} \geq 0$ for every i, j. Moreover, we will use the standard "dot above the symbol" notation to denote the time derivative, as in $\dot{u}(t)$, but we will also occasionally use $i'(t)$ to denote the derivative, whenever it helps maintain clarity of notation.

Given a vector $x \in \mathbb{R}^\lambda$, the 1-norm and the infinity norm of x are denoted as follows:

$$\|x\|_1 \triangleq \sum_{i=1}^{\lambda} |x_i| \text{ and } \|x\|_\infty \triangleq \max_{i \in \{1,\ldots,\lambda\}} |x_i|.$$

We use the notation I_λ to denote the identity matrix of size $\lambda \times \lambda$, the notation $\mathbf{0}_{\lambda \times \gamma}$ to denote the $\lambda \times \gamma$ matrix of zeroes

We also define some extreme-value operators to help express the worst-case voltage variations on the grid, as follows.

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

$$y_j = \max_{x \in \mathcal{A}} \big[f_j(x) \big]. \tag{3}$$

Thus, emax $[\cdot]$ performs element-wise maximization, and it may be computed using n LPs when $f(x)$ is linear and $\mathcal{A}$ is a convex polytope. We similarly define another operator emin $[\cdot]$ as follows.

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

$$y_j = \min_{x \in \mathcal{A}} \big[f_j(x) \big]. \tag{4}$$

We also combine emax and emin in the single $2n \times 1$ vector $\text{eopt}_{x \in \mathcal{A}}[f(x)] = \begin{bmatrix} \text{emax}_{x \in \mathcal{A}}[f(x)] \\ \text{emin}_{x \in \mathcal{A}}[f(x)] \end{bmatrix}$, which we define as follows.

Definition 3 (eopt): Let $f(x) : \mathbb{R}^m \mapsto \mathbb{R}^n$ be a vector function whose components will be denoted $f_1(x), \ldots, f_n(x)$, and let $\mathcal{A} \subseteq \mathbb{R}^m$. We define the operator $\text{eopt}_{x \in \mathcal{A}}[f(x)]$ as one that provides the $2n \times 1$ vector $y = \text{eopt}_{x \in \mathcal{A}}[f(x)]$ such that, for every $j \in \{1, 2, \ldots, n\}$

$$y_j = \max_{x \in \mathcal{A}} \big[f_j(x) \big] \tag{5}$$

$$y_{j+n} = \min_{x \in \mathcal{A}} \big[f_j(x) \big]. \tag{6}$$

If $\mathcal{A}$ is empty, then $\text{eopt}_{x \in \mathcal{A}}[f(x)]$ is undefined.

Moreover, we denote by $\mathbb{N}$ the set of all nonnegative integers, and by $\mathbb{Z}$ the set of all signed integers. In other words, $\mathbb{N} = \{0, 1, 2, \ldots\}$, and $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$. Also, let $\mathbb{C}$ denote the set of complex numbers.

IV. Problem Definition

In this section, we derive the exact solution to the vectorless RLC problem based on nodal analysis and backward

Euler (BE) discretization scheme. Aside from Lemma 1, the content of this section is not novel, but gives notation, and clearer and more compact description of the exact solution.

A. System Equations

Let G be the $n_v \times n_v$ conductance matrix [25] of the grid, which is symmetric and diagonally dominant with positive diagonal entries and nonpositive off-diagonal entries. If the graph consisting of all the resistances of the grid is a connected graph that contains at least all grid nodes $1, 2, \ldots, n_v$, and if it has at least one direct connection to a voltage source, then G is known to be irreducibly diagonally dominant [25]. With this, it can be shown that G is an $\mathcal{M}$-matrix, so that G^{-1} exists and is non-negative, $G^{-1} \geq 0$, and all the eigenvalues of G are real and positive [26]. If the resistive graph is not connected or does not cover all n_v nodes, or if there are no direct resistive connections to any voltage source, then there is an easy and practical "fix" that maintains all the above useful properties of G, which is to attach a large resistance from every grid node $1, 2, \ldots, n_v$ to ground. These large resistors have a negligible effect on the circuit solution, but they have the effect that G becomes strictly diagonally dominant, from which all the above properties of G automatically follow [25]. In fact, these resistors can be useful to model the leakage effects inside the chip.

Let M be an $n_v \times n_l$ incidence matrix consisting of ± 1 or 0 elements only, such that

$$m_{jk} = \begin{cases} 1, & \text{if node } j \text{ is connected to inductor } k \text{ and the} \\ & \text{reference current direction in } k \text{ is away from } j \\ -1, & \text{if node } j \text{ is connected to inductor } k \text{ and the} \\ & \text{reference current direction in } k \text{ is toward } j \\ 0, & \text{otherwise.} \end{cases}$$

Let $u(t)$ be the vector of node voltages, relative to ground. By superposition, $u(t)$ may be found in three steps: 1) open-circuit all the current sources and find the response, which would be $u^{(1)}(t) = V_{dd}$, 2) short-circuit all the voltage sources and find the response $u^{(2)}(t)$, and 3) find $u(t) = u^{(1)} + u^{(2)}$. To find $u^{(2)}(t)$, Kirchhoff's current law at every node $k \in \{1. \ldots, n_v\}$ provides

$$Gu^{(2)}(t) + C\dot{u}^{(2)}(t) + Mi_l(t) + Hi(t) = 0 \tag{7}$$

where $i_l(t)$ is the vector of all inductive branch currents and C is an $n_v \times n_v$ diagonal matrix of node capacitances. We are mainly interested in the voltage drop $v(t) \triangleq V_{dd} - u(t) = -u^{(2)}(t)$, so that

$$Gv(t) + C\dot{v}(t) - Mi_l(t) = Hi(t). \tag{8}$$

In addition, and in order to take into account the relationship between inductor currents and voltages, we have the familiar inductor branch equation $M^T u(t) = Li_l'(t)$, from which

$$M^T v(t) + Li_l'(t) = 0 \tag{9}$$

where L is an $n_l \times n_l$ diagonal matrix of inductance values. The pair of equations (8) and (9) represent the complete behavior of the power grid as a dynamical system.

B. Time Discretization

One approach to solving the dynamic systems (8) and (9) starts out by discretizing time and using a finite-difference approximation of the derivatives, essentially a BE numerical scheme $\dot{v}(t) = [v(t) - v(t - \Delta t)]/\Delta t$ and $i'(t) = [i(t) - i(t - \Delta t)]/\Delta t$. Assuming that Δt is small enough, we have

$$Av(t) - Mi_l(t) \approx Bv(t - \Delta t) + Hi(t) \tag{10}$$

$$M^T v(t) + Ei_l(t) \approx Ei_l(t - \Delta t) \tag{11}$$

where $B = C/\Delta t$, $A = G + B$, and $E = L/\Delta t$. In what follows, we assume that Δt is chosen such that (10) and (11) are accurate. Multiplying (11) by E^{-1} to get an expression for $i_l(t)$

$$i_l(t) = -E^{-1}M^T v(t) + i_l(t - \Delta t) \tag{12}$$

and substituting that in (10) as was done in [27], gives

$$Dv(t) = Bv(t - \Delta t) + Mi_l(t - \Delta t) + Hi(t) \tag{13}$$

where

$$D = G + \frac{C}{\Delta t} + M\left(\frac{L}{\Delta t}\right)^{-1} M^T.$$

Lemma 1: The matrix D is a symmetric $\mathcal{M}$-matrix.

Proof: Symmetry is easy to see as both G and C are symmetric, and $(ME^{-1}M^T)^T = M(E^{-1})^T M^T = ME^{-1}M^T$. It remains to prove that D is an $\mathcal{M}$-matrix. We do this by proving that D has nonpositive off-diagonal entries and eigenvalues that have positive real parts [26].

The matrices G and $(C/\Delta t)$ have nonpositive off-diagonal entries by construction. We now show that the same is true for $ME^{-1}M^T$. Notice that

$$\left(E^{-1}\right)_{ij} = \begin{cases} \dfrac{\Delta t}{L_{ii}} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, if $X = E^{-1}M^T$, we have

$$X_{ij} = \sum_{k=1}^{n_l} \left(E^{-1}\right)_{ik} \left(M^T\right)_{kj}, \quad \begin{matrix} i = 1, \ldots, n_l \\ j = 1, \ldots, n_v \end{matrix}$$

hence, $X_{ij} = (\Delta t/L_{ii})M_{ji}$. Also, if $W = ME^{-1}M^T$, then

$$W_{ij} = \sum_{k=1}^{n_l} M_{ik}X_{kj} = \sum_{k=1}^{n_l} M_{ik}\frac{\Delta t}{L_{kk}}M_{jk}, \quad \begin{matrix} i = 1, \ldots, n_v \\ j = 1, \ldots, n_v. \end{matrix}$$

By the definition, every column of the matrix M contains either one nonzero entry, or two nonzero entries where one of them is $+1$ and the other is -1. It follows that, for any $i \neq j$, we have $M_{ik}M_{jk} \leq 0$, for any k, so that $W_{ij} \leq 0, \forall i \neq j$, and the matrix $W = ME^{-1}M^T$ has nonpositive off-diagonal entries. Therefore, D has nonpositive off-diagonal entries.

It remains to show that D has eigenvalues with positive real parts. Notice that the symmetry of D implies that all the eigenvalues of D are real, so, all we need to do is to show that these eigenvalues are positive. We do this by showing that D is positive definite. For all vectors $z \in \mathbb{R}^{n_v}$, $z \neq 0$, we have

$$z^T Dz = z^T Gz + z^T \frac{C}{\Delta t} z + z^T ME^{-1}M^T z. \tag{14}$$

We know that $z^T Gz > 0$ because G is an $\mathcal{M}$-matrix, and hence symmetric positive definite. Also, $z^T Cz > 0$ because C is a positive invertible diagonal matrix. Finally, $z^T ME^{-1}M^T z = (M^T z)^T E^{-1}(M^T z) \geq 0$ because E^{-1} is also a positive invertible diagonal matrix. Therefore, $z^T Dz > 0$ for all $z \in \mathbb{R}^{n_v}$, $z \neq 0$, so that D is a symmetric positive definite matrix. Accordingly, D is a symmetric $\mathcal{M}$-matrix. ■

It follows from Lemma 1 that D^{-1} exists and $D^{-1} \geq 0$ [26]. Multiplying (13) by D^{-1} gives

$$v(t) = D^{-1}Bv(t-\Delta t) + D^{-1}Mi_l(t-\Delta t) + D^{-1}Hi(t) \quad (15)$$

then substituting this for $v(t)$ in (12), we get

$$i_l(t) = -E^{-1}M^T D^{-1}Bv(t-\Delta t) - E^{-1}M^T D^{-1}Hi(t) + \left(I_{n_l} - E^{-1}M^T D^{-1}M\right)i_l(t-\Delta t). \quad (16)$$

Combining (15) and (16) gives the system

$$x(t) = Fx(t-\Delta t) + RHi(t) \quad (17)$$

where

$$x(t) = \begin{bmatrix} v(t) \\ i_l(t) \end{bmatrix}, \quad R = \begin{bmatrix} D^{-1} \\ -E^{-1}M^T D^{-1} \end{bmatrix}$$
$$F = \begin{bmatrix} D^{-1}B & D^{-1}M \\ -E^{-1}M^T D^{-1}B & \left(I_{n_l} - E^{-1}M^T D^{-1}M\right) \end{bmatrix}. \quad (18)$$

Let $n = n_v + n_l$ so that $x(t)$ is $n \times 1$, F is $n \times n$, and R is $n \times m$.

C. Exact Solution

We are interested in the worst-case voltage drops (both maximum and minimum) under all currents $i(t)$ that satisfy the current constraints, i.e., $i(t) \in \mathcal{F}$, $\forall t \in \mathbb{R}$. Applying (17) at $(t-\Delta t)$ gives

$$x(t-\Delta t) = Fx(t-2\Delta t) + RHi(t-\Delta t)$$

and, substituting this for $x(t-\Delta t)$ back in (17) gives

$$x(t) = F^2 x(t-2\Delta t) + FRHi(t-\Delta t) + RHi(t)$$

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

$$x(t) = F^p x(t-p\Delta t) + \sum_{q=0}^{p-1} F^q RHi(t-q\Delta t). \quad (19)$$

If the grid is initialized with some state x_0 at some point in time and all current sources are kept at zero for all future time, then clearly the state would eventually go to $x = 0$, because the grid is a passive and stable system. The BE time-discretized model of the grid when $x(t-p\Delta t) = x_0$ and all current sources are off provides, from the above, that $x(t) = F^p x_0$. Because BE is absolutely stable for any time-step Δt [28], then the solution of the time-discretized system would also die down to zero eventually, for any initial state x_0, which means that

$$\lim_{p\to\infty} F^p = 0. \quad (20)$$

Recall that the spectral radius of a matrix X, denoted $\rho(X)$, is the maximum of the absolute values of the eigenvalues of X. According to [26], (20) is true if and only if

$$\rho(F) < 1. \quad (21)$$

Taking the limit as $p \to \infty$, this allows us to write

$$x(t) = \sum_{q=0}^{\infty} F^q RHi(t-q\Delta t). \quad (22)$$

Using the emax and emin notation given earlier, we can express the maximum and minimum worst-case voltage drops and inductive branch currents at all nodes at time t as $\text{emax}_{i\in\mathcal{F}}[x(t)]$ and $\text{emin}_{i\in\mathcal{F}}[x(t)]$, respectively. Here, the notation $i \in \mathcal{F}$ means that $i(\tau) \in \mathcal{F}$, for any $\tau \in \mathbb{R}$. Thus, at time t, the maximum voltage drop is available as the top (voltage) part of $\text{emax}_{i\in\mathcal{F}}[x(t)]$, and the minimum voltage drop is available as the top (voltage) part of $\text{emin}_{i\in\mathcal{F}}[x(t)]$. We also combine $\text{emax}_{i\in\mathcal{F}}[x(t)]$ and $\text{emin}_{i\in\mathcal{F}}[x(t)]$ in the single $2n \times 1$ vector $\text{eopt}_{i\in\mathcal{F}}[x(t)] = \begin{bmatrix} \text{emax}_{i\in\mathcal{F}}[x(t)] \\ \text{emin}_{i\in\mathcal{F}}[x(t)] \end{bmatrix}$, which was defined earlier.

With this, we can now define the vector of worst-case voltage drop and inductive branch current fluctuations, as was done in [19], as follows:

$$x^*(t) \triangleq \underset{i\in\mathcal{F}}{\text{eopt}}[x(t)]. \quad (23)$$

Although the currents and voltages vary with time, the constraints on $i(t)$ do not depend on time. Hence, $\mathcal{F}$ is the same for any time point t. Therefore, the result of the above application of eopt must be time-independent. The optimization needs to be performed at only one time point, any time point in fact, and so we define the time-independent vector x^*

$$x^* \triangleq \underset{i\in\mathcal{F}}{\text{eopt}}[x(t)] = \underset{i\in\mathcal{F}}{\text{eopt}}\left[\textstyle\sum_{q=0}^{\infty} F^q RHi(t-q\Delta t)\right]. \quad (24)$$

Because we are interested in the worst-case over all current waveforms that satisfy $i \in \mathcal{F}$, then the values of the source currents $i(\cdot)$ at any two time points must be treated as independent variables, and so we can "decouple" the components of (24) to obtain

$$x^* = \sum_{q=0}^{\infty} \underset{i\in\mathcal{F}}{\text{eopt}}\left[F^q RHi(t-q\Delta t)\right] \quad (25)$$

which simplifies to

$$x^* = \sum_{q=0}^{\infty} \underset{i\in\mathcal{F}}{\text{eopt}}\left[F^q RHi\right] \quad (26)$$

where we use the single vector variable i to denote the current vector for the purpose of the eopt computation. This expression (26) of x^* is the exact solution in the RLC case. It is an infinite sum whose every term requires the solution of $2n$ LPs. Clearly, it is prohibitively expensive to use this for computation. Instead, in the following, we give practical ways to estimate x^*. The estimates we will derive are related to the ones developed in [19]. Our contribution lies in carefully choosing the time step Δt as to make the parameter r in [19] always equal to 1. This has significant implications as it reduces the number of LPs to be solved by a large factor, and eliminates the need for the full inverse and for any dense matrix–matrix multiplications. Experimental results will show that the new choice of Δt also leads to more accurate bounds. Moreover, the fact that $r = 1$ and not varying means that our algorithm behaves in a more predictable way.

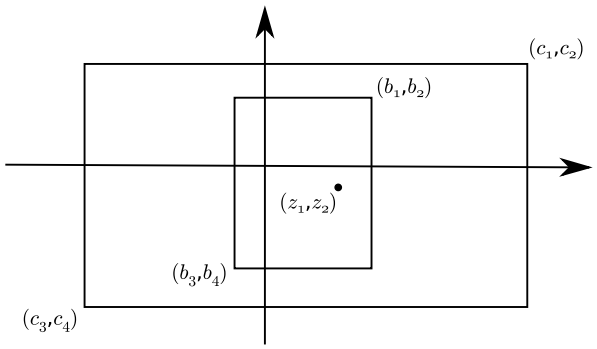

Fig. 4. Geometrical representation of extended vectors.

V. Proposed Solution

In this section, we derive conservative bounds on the exact solution (26). These bounds are much easier to compute than x^* as they require a much smaller number of LPs.

A. Extended Vectors and Matrices

The purpose of this technical section is to develop Theorems 1 and 2, which will be key to our proposed solution.

1) Extended Vectors: We give a few definitions and key results, starting with the belong ($\in$) relation among vectors. These will be used to express the bounds on the exact solution (26).

Definition 4: If z is an $n \times 1$ vector, and b is a $2n \times 1$ vector, then we say that $z \in b$ if, $\forall j \in \{1, \ldots, n\}$

$$z_j \leq b_j \quad \text{and} \quad z_j \geq b_{j+n}.$$

Thus, $z \in b$ means that z is upper-bounded (element-wise) by the top half of b and likewise lower-bounded by the bottom half of b. One can interpret this definition geometrically as shown in Fig. 4 where $n = 2$. The point (z_1, z_2) lies inside the rectangle defined by the two vertices (b_1, b_2) and (b_3, b_4), and so

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \in \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}. \tag{27}$$

We will refer to $2n \times 1$ vectors as extended vectors. We say that b is an empty extended vector if there does not exist any z for which $z \in b$. Note that b is nonempty if and only if $b_j \geq b_{j+n}$, $\forall j \in \{1, 2, \ldots, n\}$. If $f(i) : \mathbb{R}^m \mapsto \mathbb{R}^n$ is a vector function, and if $b = \text{eopt}_{i \in \mathcal{A}} [f(i)]$, where $\mathcal{A}$ is a nonempty set, then clearly b is a nonempty extended vector.

Lemma 2: If $f(i) : \mathbb{R}^m \mapsto \mathbb{R}^n$ and $\mathcal{A}$ is a nonempty subset of $\mathbb{R}^m$, then

$$f(z) \in \underset{i \in \mathcal{A}}{\text{eopt}} \big[f(i) \big], \quad \forall z \in \mathcal{A}. \tag{28}$$

Proof: Let $f_1(i), \ldots, f_n(i)$ be the n components of $f(i)$, and $y \triangleq \text{eopt}_{i \in \mathcal{A}} [f(i)]$. Then, by the definition of eopt, we have $y_j = \max_{i \in \mathcal{A}} [f_j(i)], \forall j \in \{1, 2, \ldots, n\}$. This implies $f_j(z) \leq y_j, \forall j, \forall z \in \mathcal{A}$. Similarly, $y_{j+n} = \min_{i \in \mathcal{A}} [f_j(i)], \forall j$, which implies $f_j(z) \geq y_{j+n}, \forall j, \forall z \in \mathcal{A}$. Therefore, $f(z) \in y$. ∎

We now introduce the subset ($\subseteq$) relation among vectors.

Definition 5: If b and c are $2n \times 1$ vectors, then we say that $b \subseteq c$ if, $\forall j \in \{1, 2, \ldots, n\}$

$$b_j \leq c_j \quad \text{and} \quad b_{j+n} \geq c_{j+n}.$$

The subset relation among vectors can also be interpreted geometrically: back to Fig. 4, the rectangle defined by the vertices (b_1, b_2) and (b_3, b_4) lies completely inside the rectangle defined by the vertices (c_1, c_2) and (c_3, c_4), and so

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \subseteq \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}. \tag{29}$$

A few simple properties can be stated without proof. First, it should be clear that, if $b \subseteq c$, then

$$\forall z, \quad z \in b \Longrightarrow z \in c. \tag{30}$$

The converse is true in the case where b is nonempty: if b is nonempty and is such that $\forall z, z \in b \Rightarrow z \in c$, then $b \subseteq c$. Second, it is clear that the subset relation is transitive

$$b \subseteq c \quad \text{and} \quad c \subseteq d \quad \Longrightarrow \quad b \subseteq d. \tag{31}$$

Third, it is also clear that subset relations may be combined by summation

$$a \subseteq b \quad \text{and} \quad c \subseteq d \quad \Longrightarrow \quad a + c \subseteq b + d. \tag{32}$$

Lemma 3: Let $f(i) : \mathbb{R}^m \mapsto \mathbb{R}^n$ be a vector function, $\mathcal{A}$ be a nonempty subset of $\mathbb{R}^m$, and b be a $2n \times 1$ vector. if $f(i) \in b, \forall i \in \mathcal{A}$, then $\text{eopt}_{i \in \mathcal{A}} [f(i)] \subseteq b$.

Proof: Let $f(i) \in b, \forall i \in \mathcal{A}$, and $y = \text{eopt}_{i \in \mathcal{A}} [f(i)]$. For every $j \in \{1, 2, \ldots, n\}$, we have $y_j = \max_{i \in \mathcal{A}} [f_j(i)]$, so that $y_j \leq b_j$, and $y_{j+n} = \min_{i \in \mathcal{A}} [f_j(i)]$, so that $y_{j+n} \geq b_{j+n}$. Then, $y \subseteq b$ and the proof is complete. ∎

2) Subset-Preserving: Consider the class of $2n \times 2n$ matrices. They map extended vectors into other extended vectors as follows: if b is an extended vector and N is a $2n \times 2n$ matrix, then N maps b to the extended vector $Nb \in \mathbb{R}^{2n}$.

Definition 6: A $2n \times 2n$ matrix N is said to be subset-preserving if, for any two extended vectors b, c, we have that

$$b \subseteq c \Rightarrow Nb \subseteq Nc.$$

Lemma 4: If N_1 and N_2 are subset-preserving matrices, then $N_1 N_2$ and $(N_1 + N_2)$ are also subset-preserving.

Proof: For any two extended vectors b and c such that $b \subseteq c$, we have $N_1 b \subseteq N_1 c$ and $N_2 b \subseteq N_2 c$, so that $(N_1 + N_2)b = N_1 b + N_2 b \subseteq N_1 c + N_2 c = (N_1 + N_2)c$ and so $(N_1 + N_2)$ is subset-preserving.

Also, because $N_2 b \subseteq N_2 c$, then $N_1(N_2 b)) \subseteq N_1(N_2 c)$, so that $N_1 N_2 b \subseteq N_1 N_2 c$ and $N_1 N_2$ is subset-preserving. ∎

Lemma 5: If N is a subset-preserving matrix, then $\lim_{k \to \infty} \sum_{j=0}^{k} N^j$ is subset-preserving, if it exists.

Proof: The results follows by repeated application of Lemma 4. ∎

3) Extended Matrices: We define the concept of the extension of a matrix, first introduced, but without development of its full potential, in [19].

Definition 7: Let J be an $n \times n$ matrix. We denote by $|J|$ the matrix consisting of the absolute values of the elements of J. In other words

$$|J|_{ij} = \big|J_{ij}\big|, \quad \forall i, j \in \{1, \ldots, n\}. \tag{33}$$

Definition 8: Let J be an $n \times n$ matrix, and let $J^+ = (1/2)(J+|J|)$ and $J^- = (1/2)(J-|J|)$. We define the extension of J as the $2n \times 2n$ matrix $\tilde{J}$, given by

$$\tilde{J} = \begin{bmatrix} J^+ & J^- \\ J^- & J^+ \end{bmatrix}. \tag{34}$$

Notice that $J^+ \geq 0$ consists of only the positive or zero elements of J while $J^- \leq 0$ consists of only the negative or zero elements of J, so that we have

$$J^+_{ij} = \begin{cases} J_{ij}, \text{ if } J_{ij} \geq 0 \\ 0, \quad \text{otherwise.} \end{cases} \qquad J^-_{ij} = \begin{cases} J_{ij}, \text{ if } J_{ij} \leq 0 \\ 0, \quad \text{otherwise.} \end{cases}$$

Note that if $J \geq 0$ then $\tilde{J} = \begin{bmatrix} J & 0 \\ 0 & J \end{bmatrix}$, while if $J \leq 0$, then $\tilde{J} = \begin{bmatrix} 0 & J \\ J & 0 \end{bmatrix}$. We will show shortly that $\tilde{J}$ is a subset-preserving matrix, which will lead to one of our main results.

We first give the following significant result that contributes to the solution.

Lemma 6: If J is an $n \times n$ matrix, z is an $n \times 1$ vector, and b is a $2n \times 1$ nonempty extended vector, then

$$\operatorname*{eopt}_{z \in b}(Jz) = \tilde{J}b. \tag{35}$$

Proof: Let $c = Jz$ and $d = \tilde{J}b$. For any $k \in \{1, \ldots, n\}$, we have

$$\begin{aligned} \max_{z \in b}[c_k] &= \max_{z \in b}\left[\sum_{j=1}^{n} J_{kj} z_j\right] = \sum_{j=1}^{n} \max_{z \in b}\big[J_{kj} z_j\big] \\ &= \sum_{J_{kj} \geq 0} J_{kj} \max_{z \in b}\big[z_j\big] + \sum_{J_{kj} \leq 0} J_{kj} \min_{z \in b}\big[z_j\big] \\ &= \sum_{j=1}^{n} J^+_{kj} b_j + \sum_{j=1}^{n} J^-_{kj} b_{j+n} = d_k. \end{aligned}$$

Likewise

$$\min_{z \in b}[c_k] = \textstyle\sum_{j=1}^{n} J^-_{kj} b_j + \sum_{j=1}^{n} J^+_{kj} b_{j+n} = d_{k+n}$$

which completes the proof. ∎

Corollary 1: If J is an $n \times n$ matrix, z is an $n \times 1$ vector, and b is a $2n \times 1$ vector, then

$$z \in b \quad \Longrightarrow \quad Jz \in \tilde{J}b.$$

Proof: Invoking Lemma 2, we have that $Jz \in \operatorname{eopt}_{z \in b}(Jz)$, which by Lemma 6 implies that $Jz \in \tilde{J}b$. ∎

We now arrive at our first main result.

Theorem 1: Let $f(i) : \mathbb{R}^m \mapsto \mathbb{R}^n$ be a vector function and $\mathcal{A}$ be a nonempty subset of $\mathbb{R}^m$. Also, let J be an $n \times n$ matrix. Then

$$\operatorname*{eopt}_{i \in \mathcal{A}}\big[Jf(i)\big] \subseteq \tilde{J} \operatorname*{eopt}_{i \in \mathcal{A}}\big[f(i)\big]. \tag{36}$$

Proof: By Lemma 2, we have

$$f(i) \in \operatorname*{eopt}_{i \in \mathcal{A}}\big[f(i)\big], \quad \forall i \in \mathcal{A}.$$

Using the corollary to Lemma 6, we can write

$$Jf(i) \in \tilde{J} \operatorname*{eopt}_{i \in \mathcal{A}}\big[f(i)\big], \quad \forall i \in \mathcal{A}.$$

Applying Lemma 3, we have that

$$\operatorname*{eopt}_{i \in \mathcal{A}}\big[Jf(i)\big] \subseteq \tilde{J} \operatorname*{eopt}_{i \in \mathcal{A}}\big[f(i)\big]$$

which completes the proof. ∎

We now derive our second main result.

Lemma 7: If J is an $n \times n$ matrix, then $\tilde{J}$ is subset-preserving.

Proof: Let b and c be $2n \times 1$ vectors such that $b \subseteq c$, and let b_1 and b_2 be the top and bottom halves of b, and c_1 and c_2 be the top and bottom halves of c, respectively. Because $b_1 \leq c_1$, then $J^+ b_1 \leq J^+ c_1$ and $J^- b_1 \geq J^- c_1$. Likewise, because $b_2 \geq c_2$, then $J^+ b_2 \geq J^+ c_2$ and $J^- b_2 \leq J^- c_2$. Making suitable additions, we have $J^+ b_1 + J^- b_2 \leq J^+ c_1 + J^- c_2$ and $J^- b_1 + J^+ b_2 \geq J^- c_1 + J^+ c_2$, so that $\tilde{J}b \subseteq \tilde{J}c$, which means that $\tilde{J}$ is subset-preserving. ∎

Lemma 7 leads to our second main result.

Theorem 2: If $\rho(\tilde{J}) < 1$, then $(I_{2n} - \tilde{J})^{-1}$ is subset-preserving.

Proof: If $\rho(\tilde{J}) < 1$, then $\sum_{k=0}^{\infty} \tilde{J}^k$ exists and $(I_{2n} - \tilde{J})$ is nonsingular with $(I_{2n} - \tilde{J})^{-1} = \sum_{k=0}^{\infty} \tilde{J}^k$ [26]. But, because $\tilde{J}$ is subset-preserving due to Lemma 7, then by Lemma 5 the infinite sum $\sum_{k=0}^{\infty} \tilde{J}^k$ is subset-preserving, which completes the proof. ∎

Theorems 1 and 2 are key results that will be used below to derive the bounds on x^*.

B. Bounds

Given that the values of the source currents $i(\cdot)$ at any two time points are independent variables, as in Section IV-C, we can use (17) to write

$$x^* = \operatorname*{eopt}_{i(t) \in \mathcal{F}}[x(t)] = \operatorname*{eopt}_{i(t) \in \mathcal{F}}[Fx(t - \Delta t)] + \operatorname*{eopt}_{i(t) \in \mathcal{F}}[RHi(t)] \tag{37}$$

because $x(t-\Delta t)$ depends on past values of $i(\cdot)$ only. Applying Theorem 1 with $f(i) = x(t - \Delta t)$ gives

$$x^* \subseteq \tilde{F} \operatorname*{eopt}_{i(t) \in \mathcal{F}}[x(t - \Delta t)] + \operatorname*{eopt}_{i \in \mathcal{F}}[RHi] \tag{38}$$

where we have replaced $i(t)$ by the dummy variable vector i for purpose of the eopt optimization. Then, because x^* is time-independent, we have $\operatorname{eopt}_{i(t) \in \mathcal{F}}[x(t)] = \operatorname{eopt}_{i(t) \in \mathcal{F}}[x(t - \Delta t)] = x^*$, and

$$x^* \subseteq \tilde{F}x^* + \operatorname*{eopt}_{i \in \mathcal{F}}[RHi] \tag{39}$$

so that

$$\big(I_{2n} - \tilde{F}\big)x^* \subseteq \operatorname*{eopt}_{i \in \mathcal{F}}[RHi]. \tag{40}$$

Algorithm 1 FIND_RLC_BOUND

Input: RLC circuit matrices, $\mathcal{F}$
Output: $\bar{x}$
1: Find Δt such that $\rho(\tilde{F}(\Delta t)) < 1$.
2: Compute the required columns of D^{-1} and find RH.
3: Find $z^* = \operatorname{eopt}_{i\in\mathcal{F}}[RHi]$.
4: Solve $(I_{2n} - \tilde{F})\bar{x} = z^*$ for $\bar{x}$ using CG.
5: **return** $\bar{x}$

If $\rho(\tilde{F}) < 1$, then Theorem 2 provides that $(I_{2n} - \tilde{F})^{-1}$ is subset-preserving, so that, using (40), we can write

$$\left(I_{2n} - \tilde{F}\right)^{-1}\left(I_{2n} - \tilde{F}\right)x^* \subseteq \left(I_{2n} - \tilde{F}\right)^{-1} \operatorname*{eopt}_{i\in\mathcal{F}}[RHi]. \quad (41)$$

Let

$$\bar{x} \triangleq \begin{bmatrix} x_{ub} \\ x_{lb} \end{bmatrix} \triangleq \left(I_{2n} - \tilde{F}\right)^{-1} \operatorname*{eopt}_{i\in\mathcal{F}}[RHi] \quad (42)$$

where x_{ub} and x_{lb} are $n \times 1$ vector. Hence, when $\rho(\tilde{F}) < 1$, we have

$$x^* \subseteq \bar{x}. \quad (43)$$

Note that $x_{lb} \leq x^* \leq x_{ub}$. It turns out that these bounds are in fact accurate as will be shown in the experimental results section. Thus, one can check the grid safety by comparing x_{ub} and x_{lb} to the upper and lower user-specified thresholds, respectively.

Unfortunately, the spectral radius $\rho(\tilde{F})$ might be greater than 1. One way of dealing with this is by using a different form of (22) where every r consecutive terms are grouped together. This leads to an expression of the bound that depends on r and that is more complicated and harder to compute. This approach is detailed in [19].

In the following, we propose a more efficient option that benefits from a careful choice of Δt that leads to $\rho(\tilde{F}) < 1$. In contrast with [19], our approach requires a much smaller number of LPs and matrix multiplications, and only needs certain columns of D^{-1} instead of the full inverse.

Notice that $\tilde{F}$ is in fact a function of Δt, and we will explicitly denote it by $\tilde{F}(\Delta t)$ in the rest of this paper. In Section VI we will prove that there always exists a value for Δt for which $\rho(\tilde{F}(\Delta t)) < 1$. We will also show how to find such a value and demonstrate that it gives accurate results in Section VIII.

Once Δt is chosen, the rest of the algorithm becomes relatively simple. We first compute $z^* \triangleq \operatorname{eopt}_{i\in\mathcal{F}}[RHi]$ by solving $2n$ LPs, each having m variables. Then, we solve the system $(I_{2n} - \tilde{F})\bar{x} = z^*$. We will show later how to do this iteratively using the CG method [26]. The overall procedure for finding $\bar{x}$ is described in Algorithm 1.

VI. Choice of the Time Step

Because F depends on Δt, the spectral radius of $\tilde{F}$ will also depend on Δt. In what follows, we will refer to F by $F(\Delta t)$ and to $\tilde{F}$ by $\tilde{F}(\Delta t)$ to emphasize their dependence on Δt.

A. Existence

In this section, we will prove an important result which will help us deal with the fact that $\rho(\tilde{F}(\Delta t))$ is not always smaller than 1. We will prove that for every $c \in (0, 1)$, there exists a value of $\Delta t > 0$ for which $\rho(|F(\Delta t)|) = c$. We will then use Lemma 8 below, proven in [19], to show that for this value of Δt, we have $\rho(\tilde{F}(\Delta t)) < 1$.

Lemma 8 [19]: For any square matrix J, we have $\rho(\tilde{J}) = \max(\rho(J), \rho(|J|))$.

Several key results are needed before presenting and proving the main theorem. We start by recalling an important result from linear algebra.

Lemma 9 [29]: The eigenvalues of a matrix Q depend continuously on the entries of Q.

Notice that, for any $\alpha, \beta \in \mathbb{R}$, we have $|\alpha| = |(\alpha-\beta)+\beta| \leq |\alpha - \beta| + |\beta|$ so that

$$|\alpha| - |\beta| \leq |\alpha - \beta|. \quad (44)$$

This property will be useful for the proof of several of the following results.

Lemma 10: The spectral radius of $|F(\Delta t)|$ is a continuous function of Δt for any $\Delta t > 0$.

Proof: Clearly, $B(\Delta t)$ and $E(\Delta t)$ are continuous functions of Δt for $\Delta t \in (0, \infty)$. Accordingly, $E^{-1}(\Delta t)$ is also continuous because the inverse operator is continuous on the set of invertible matrices [29]. Therefore, $D(\Delta t)$ is continuous because addition and multiplication are also continuous operators. Following the same argument, we can conclude that $D^{-1}(\Delta t)$, $D^{-1}(\Delta t)B(\Delta t)$, $D^{-1}(\Delta t)M$, $-E^{-1}(\Delta t)M^T D^{-1}(\Delta t)B(\Delta t)$, and $(I_{n_l} - E^{-1}(\Delta t)M^T D^{-1}(\Delta t)M)$ are all continuous functions of Δt, which leads to $F(\Delta t)$, as well as $|F(\Delta t)|$, being continuous. Therefore, using Lemma 9, the eigenvalues of $|F(\Delta t)|$ also depend continuously on Δt. In other words, for every $\Delta t_0 \in (0, \infty)$, and for every $\epsilon > 0$, there exists $\delta > 0$ such that if $|\Delta t - \Delta t_0| < \delta$, then

$$\lambda_i \in N_\epsilon(\lambda_{0,i}) \quad \forall i = 1, \ldots, n \quad (45)$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $|F(\Delta t)|$, $\lambda_{0,1}, \ldots, \lambda_{0,n}$ are the eigenvalues of $|F(\Delta t_0)|$, and $N_\epsilon(\lambda_{0,j})$ is the ϵ-neighborhood of $\lambda_{0,j}$ defined as the set $\{\lambda \in \mathbb{C} : |\lambda - \lambda_{0,i}| < \epsilon\}$. Using (44), we have $\forall i = 1, \ldots, n$

$$\begin{aligned} \lambda_i \in N_\epsilon(\lambda_{0,i}) &\Rightarrow |\lambda_i| - |\lambda_{0,i}| \leq |\lambda_i - \lambda_{0,i}| < \epsilon \\ &\Rightarrow |\lambda_i| < |\lambda_{0,i}| + \epsilon \\ &\Rightarrow \max_{i=1,\ldots,n} |\lambda_i| < \max_{i=1,\ldots,n} \left(|\lambda_{0,i}| + \epsilon\right) \\ &= \max_{i=1,\ldots,n} \left(|\lambda_{0,i}|\right) + \epsilon \\ &\Rightarrow \rho(|F(\Delta t)|) < \rho(|F(\Delta t_0)|) + \epsilon. \end{aligned} \quad (46)$$

In a similar fashion, we can show that

$$\rho(|F(\Delta t_0)|) < \rho(|F(\Delta t)|) + \epsilon. \quad (47)$$

Combining (46) and (47), we conclude that

$$\rho(|F(\Delta t_0|) - \epsilon < \rho(|F(\Delta t)|) < \rho(|F(\Delta t_0|) + \epsilon$$

meaning

$$|\rho(|F(\Delta t|) - \rho(|F(\Delta t_0)|)| < \epsilon.$$

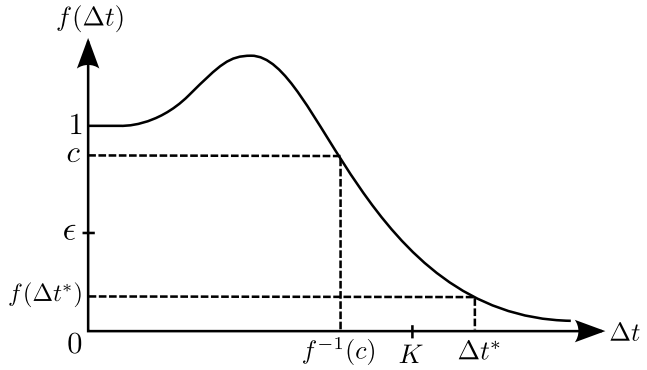

Fig. 5. Intermediate value theorem.

This being true for every ϵ and for every $\Delta t_0 \in (0, \infty)$, we conclude that $\rho(|F(\Delta t)|)$ is a continuous function of Δt on $(0, \infty)$. ■

The following lemmas are also useful to understand the behavior of $\rho(|F(\Delta t)|)$ for $\Delta t > 0$.

Lemma 11: We have

$$\lim_{\Delta t \to \infty} \rho(|F(\Delta t)|) = 0 \quad \text{and} \quad \lim_{\Delta t \to 0} \rho(|F(\Delta t)|) = 1.$$

The proof of Lemma 11 is available in Appendix A.

We now present the main theorem of this section.

Theorem 3: For every $c \in (0, 1)$, there exists $\Delta t > 0$ such that $\rho(|F(\Delta t)|) = c$.

Proof: Define the function $f : \mathbb{R}_+ \mapsto \mathbb{R}_+$ as follows:

$$f(\Delta t) = \begin{cases} \rho(|F(\Delta t)|) & \text{if } \Delta t > 0 \\ 1 & \text{if } \Delta t = 0. \end{cases} \tag{48}$$

Note that the function f is continuous on $\mathbb{R}_+$ due to Lemmas 10 and 11. For any $0 < \epsilon < c$, and because $\lim_{\Delta t \to \infty} f(\Delta t) = \lim_{\Delta t \to \infty} \rho(|F(\Delta t)|) = 0$ due to Lemma 11, then $\exists K > 0$ such that if $\Delta t > K$, then $|f(\Delta t)| = f(\Delta t) < \epsilon$. In other words there exists $\Delta t^* > K$ such that $f(\Delta t^*) < c$. Knowing that f is continuous on the closed interval $[0, \Delta t^*]$, and that $f(\Delta t^*) < c < 1$ we conclude, using the intermediate value theorem, that $\exists \Delta t \in (0, \Delta t^*)$ such that $f(\Delta t) = c$ (see Fig. 5), and hence the statement of the theorem. ■

Corollary 2: There exists $\Delta t > 0$ such that $\rho(\tilde{F}(\Delta t)) < 1$.

Proof: Recall that $\rho(F(\Delta t)) < 1$ for any $\Delta t > 0$ by (21). Therefore, if Δt was found such that $\rho(|F(\Delta t)|) = c$ for some $c \in (0, 1)$ as per Theorem 3, then for that Δt, and using Lemma 8, we have $\rho\big(\tilde{F}(\Delta t)\big) = \max\{\rho(F(\Delta t)), c\} < 1$. ■

Of course, it is not enough for such a Δt to exist, it must also be small enough to allow for an accurate time-discretization in (10) and (11). We will see in the results section that accuracy is indeed preserved, with Δt computed as we next describe.

B. Method

Now that we have proved that there exists a Δt such that $0 < \rho(\tilde{F}(\Delta t)) < 1$, we need to show how such a Δt can be found. We propose an iterative method that solves the equation $\rho(|F(\Delta t)|) = c$ for some $c \in (0, 1)$. The method uses the bisection method for root finding, and a variant of the power method to find the spectral radius of $|F(\Delta t)|$ at every step of the bisection method.

Algorithm 2 FIND_TIME_STEP

Input: Interval $[a, b]$, RLC circuit matrices
Output: Δt

```
a_1 = a, b_1 = b, p_0 = a, p_1 = 0.5(a_1 + b_1), i = 1
while |p_i − p_{i−1}| > ξ or |g(p_i) − c| > ξ do
    if (g(p_i) − c)(g(a_i) − c) > 0 then
        a_{i+1} = p_i, b_{i+1} = b_i
    else
        a_{i+1} = a_i, b_{i+1} = p_i
    end if
    p_{i+1} = 0.5(a_{i+1} + b_{i+1})
    i = i + 1
end while
return Δt = g(p_i)
```

1) Bisection Method: For improved numerical performance, we have found that scaling of the problem is useful. So, let $\gamma = \log_{10}(\Delta t)$, and define $g(\gamma) = \rho(|F(10^\gamma)|)$. We want to solve $g(\gamma) = c$, for some $0 < c < 1$ A simple way of doing this is by solving the equation $g(\gamma) - c = 0$ using the bisection method [30], which is a root-finding method that takes as input the function g, an interval $[a, b]$, and a tolerance ξ. It then keeps dividing the initial interval in halves in a similar fashion to binary search. The process is shown in Algorithm 2.

2) Variant of the Power Method: Algorithm 2 requires evaluating the function g several times. This is equivalent to finding the spectral radius of $|F(\Delta t)|$ for several values of Δt. Typically, finding the spectral radius of a matrix is done using the power method [26] which finds the dominant eigenvalue of a matrix. But because $|F|$ is not symmetric, its eigenvalues might be complex, in which case the power method might fail [31]. Alternatively, one can use a variant of the power method that is based on the theorem below.

Theorem 4 [31]: Let $Q \geq 0$ be an $n \times n$ irreducible matrix, and q_0 be an arbitrary positive n-dimensional vector. Defining $q_\nu = Qq_{\nu-1} = \cdots = Q^\nu q_0$, $\nu \geq 1$, let

$$\underline{\lambda}_\nu = \min_{1 \leq i \leq n} \frac{q_{\nu+1}^{(i)}}{q_\nu^{(i)}} \quad \text{and} \quad \overline{\lambda}_\nu = \max_{1 \leq i \leq n} \frac{q_{\nu+1}^{(i)}}{q_\nu^{(i)}} \tag{49}$$

where the superscript i represents the ith component of a vector. Then, the spectral radius of Q satisfies

$$\underline{\lambda}_0 \leq \underline{\lambda}_1 \leq \underline{\lambda}_2 \leq \cdots \leq \rho(Q) \leq \cdots \overline{\lambda}_2 \leq \overline{\lambda}_1 \leq \overline{\lambda}_0. \tag{50}$$

One can use Theorem 4 to iteratively generate bounds on the spectral radius of $|F|$ until the difference between the upper bound and the lower bound becomes less than some tolerance δ. To guarantee convergence, $|F|$ must be irreducible and primitive [31]. A non-negative square matrix X is said to be primitive if $X^k > 0$ for some $k \in \mathbb{N}$. A sufficient condition for a matrix to be primitive is for the matrix to be non-negative, irreducible, and have one strictly positive element on the main diagonal. Unfortunately, $|F|$ might not be irreducible.

To overcome this problem, we let

$$P = |F| + \begin{bmatrix} 0 & \epsilon & 0\ldots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & & \ddots & \epsilon \\ \epsilon & 0 & \ldots & 0 \end{bmatrix} \in \mathbb{R}^{n\times n}$$

for some small $\epsilon > 0$. This ensures that P is irreducible [31]. Being irreducible, nonnegative, and having at least one strictly positive diagonal entry (say from the diagonal of $D^{-1}B$), P is a primitive matrix [29]. Therefore, one can use the variant of the power method presented above to find $\rho(P)$, which is very close to $\rho(|F|)$ for practical purposes, if ϵ is small enough [31].

VII. Implementation

In this section, we present some details regarding the implementation of Algorithms 1 and 2. We provide these details because it may not be clear from Algorithms 1 and 2 how certain steps are to be done in a real implementation of our method.

A. Computing the Matrix RH

Computing the matrix RH requires finding $D^{-1}H$. We do this using an LU factorization of D followed by a sequence of forward/backward solves against the columns of H. Every time a column of $D^{-1}H$ is generated, all the very small entries are dropped in order to reduce the memory consumption. The threshold that determines which entries to drop is found using a separate engine which takes, as input, a user-defined threshold (in mV) on the voltage fluctuations at every node. We skip the details of this engine due to lack of space. Once $D^{-1}H$ is computed, finding the rest of RH is a simple task because the matrix $-E^{-1}M^T$ is an extremely sparse matrix.

B. Matrix–Vector Multiplications in Theorem 4

The procedure of Theorem 4 requires successive matrix–vector multiplications of the form $Px = (|F| + \mathcal{E})x$ where $x \in \mathbb{R}^n_+$. Let $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ where $x_1 \in \mathbb{R}^{n_v}_+$ and $x_2 \in \mathbb{R}^{n_l}_+$.

Let $Y \triangleq D^{-1}M$. The matrix Y can be found relatively easily because n_l (which is the number of columns of M) is generally much smaller than n_v, and hence one can find the jth column y_j of Y by solving the system $Dy_j = m_j$ for y_j, where m_j is the jth column of M. Once Y is found explicitly, and because $D^{-1}B \geq 0$, one can use (18) to write

$$Px = \begin{bmatrix} D^{-1}Bx_1 + |Y|x_2 \\ |E^{-1}Y^TB|x_1 + |I_{n_l} - E^{-1}M^TY|x_2 \end{bmatrix} + \mathcal{E}x. \quad (51)$$

To find $z = D^{-1}Bx_1$, one can quickly find Bx_1 (because B is a diagonal matrix), and then solve the system of equations $Dz = Bx_1$ for z.

C. Conjugate Gradient

The last step of Algorithm 1 requires solving the linear system $(I_{2n} - \tilde{F})\bar{x} = z^*$ for $\bar{x}$. We do this using the CG method [26]. But, because CG requires the system matrix to be positive definite, we will transform our system into one that can be solved using CG, and which exhibits the same solution

$$\left(I_{2n} - \tilde{F}\right)^T\left(I_{2n} - \tilde{F}\right)\bar{x} = \left(I_{2n} - \tilde{F}\right)^T z^*. \quad (52)$$

The matrix $(I_{2n} - \tilde{F})^T(I_{2n} - \tilde{F})$ is clearly symmetric, and is in fact positive definite. To see why this is true, notice that for any vector $y \neq 0$, $y^T(I_{2n} - \tilde{F})^T(I_{2n} - \tilde{F})y = ((I_{2n} - \tilde{F})y)^T(I_{2n} - \tilde{F})y = \|(I_{2n} - \tilde{F})y\|_2$ which is always non-negative. But $(I_{2n} - \tilde{F})$ has full rank because it is nonsingular when Δt is chosen as explained in Section VI. Hence, its null space contains only the zero vector, so that $(I_{2n} - \tilde{F})y \neq 0$ because $y \neq 0$. Consequently, $\|(I_{2n} - \tilde{F})y\|_2$ is in fact strictly positive. Let $T = (I_{2n} - \tilde{F})^T(I_{2n} - \tilde{F})$, and $e = (I_{2n} - \tilde{F})^T z^*$. The CG algorithm is shown in Algorithm 3. The algorithm requires, as input, a tolerance ϵ which specifies when CG converges, and a nonzero initial solution $x_{b,0}$, which can be set to a vector of ones. The vector e and the matrix T are also required in Algorithm 3. Unfortunately, $\tilde{F}$ is not available explicitly because it depends on D^{-1} which we do not compute. Therefore, finding e cannot be done directly, and T is not available explicitly. Luckily, this problem can be easily solved because T is only needed to perform matrix–vector multiplications of the form Ty for some $y \in \mathbb{R}^{2n}$. One can observe that such multiplications, as well as finding e, require computing products of the form $\tilde{F}y$ or $\tilde{F}^Ty$ for some $y \in \mathbb{R}^{2n}$. These products can be easily found because $\tilde{F}$ can be expressed in terms of F and $|F|$. We already know how to compute products of the form $|F|x$ from Section VII-B. Products of the form $|F|^Tx$, Fx, and F^Tx can be done in a similar fashion.

Algorithm 3 CONJUGATE_GRADIENT

Input: T, initial solution $x_{b,0}$, ϵ
Output: $\bar{x}$

```
Set r_0 ← T x_{b,0} − e
Set p_0 ← −r_0, k ← 0
while ‖r_k‖_∞ > ε do
    α_k = r_k^T r_k / (p_k^T T p_k)
    x_{b,k+1} ← x_{b,k} + α_k p_k
    r_{k+1} ← r_k + α_k T p_k
    β_{k+1} ← r_{k+1}^T r_{k+1} / r_k^T r_k
    p_{k+1} ← −r_{k+1} + β_{k+1} p_k
    k ← k + 1
end while
return x_k
```

D. Discussion

In this section, we compare the complexity of our approach with the complexity of the approach of [19] to show why our approach is superior. There are four steps in the algorithm of [19] that dominate the total runtime: 1) computing an approximate full matrix inverse; 2) a sequence of dense matrix–matrix multiplications; 3) solving a sequence of LPs; and 4) solving a dense linear system using LU. Our approach presents a significant improvement over [19] in terms of speed

because it refines each of the four steps above. First, our approach does not require the full inverse. Instead, it only computes m columns of the inverse (Recall that m is the number of current sources in the grid). If m is a small fraction of n_v, one could save a lot of resources by computing only a portion of the inverse of D. Typically, m is a small fraction of n_v due to the fact that current sources are usually attached only to the bottommost metal layer in the grid. Second, our approach does not require any dense matrix–matrix multiplications because the parameter r from [19] is always 1 in our case due to our choice of Δt. Third, our approach requires solving only $2(n_v + n_l)$ LPs instead to $2rn_v$ LPs as in the case of [19]. Because n_l is much smaller than n_v, and because r can be as large as 10, we have $2(n_v + n_l) \ll 2rn_v$. Finally, instead of performing an LU factorization of a dense matrix as was done in the last step of [19], we use the CG method while taking advantage of the structure of the matrix $\tilde{F}$ as explained in Section VII-C. One additional step that our algorithm performs is finding the proper Δt. As will see in the experimental results section, this step is quite efficient.

VIII. Experimental Results

Algorithms 1–3, have been implemented in C++ making use of the Mosek optimization package [32] to solve the required LPs. We carried out several experiments on a set of power grids, using a 3.4 GHz Linux machine with 32 GB of memory. Some of the grids were generated based on user specifications, including grid dimensions, metal layers, pitch and width per layer, and C4 and current source distribution. These C4s and current sources were randomly placed on the grid. Moreover, the circuits generated include user-defined RLC models for the package-grid interconnections and all the experiments were performed on grids with a 1.1 V supply voltage. The rest of the grids tested were obtained by extracting the VDD subgrids from the IBM power grid benchmarks [23].

To asses the quality of our results, we computed the maximum and minimum worst-case voltage fluctuations on the grids using the exact summation (26) and using the proposed bounds. Because the exact solution is an infinite summation, it is impossible to compute it exactly. Thus, we use the following approximation:

$$x^* \approx \tilde{x}^* = \sum_{q=0}^{N} \operatorname*{eopt}_{i \in \mathcal{F}} \left[F^q RHi \right] \tag{53}$$

for some large enough integer N. Notice that in this expression, and because this refers to the exact solution, we do not use the value of Δt generated using Algorithm 2. Instead, we use a much smaller Δt, namely $\Delta t = 100$ fs, in order to guarantee that the BE discretization used is as accurate as possible. Choosing such a small Δt still allows (26) to converge because, unlike the derived bounds, the convergence of (26) depends on the spectral radius of F, which is always less than 1 as explained in Section IV-C.

Table I shows the time-steps found and the maximum absolute error between the upper and lower bound vectors obtained by our technique (using the Δt that was generated to guarantee $\rho(\tilde{F}) < 1$) and the vectors resulting from computing $\tilde{x}^*$ (using a much smaller Δt). Because $\tilde{x}^*$ is very expensive to compute, we were only able to test the accuracy of the bounds for grids having at most 4075 nodes. For the grids tested, we can clearly see that the error is below 1mV, which shows that our approach is very accurate for all practical purposes. In fact, our upper and lower bound errors are almost ten times smaller than the errors reported in [19]. Fig. 6 shows an example of a voltage drop waveform with peaks that are quite close to the bounds generated by our algorithm. The figure corresponds to node 4000 in the 4075-nodes grid and the loading currents are chosen such that they satisfy the local and global constraints. This particular simulation was done using HSPICE.

To further demonstrate the accuracy of our approach, Fig. 7 shows a scatter plot of the relative error, in percent, versus the maximum worst-case voltage fluctuations on the 4075-node grid. The figure also shows the curve that corresponds to an absolute error of 1 mV. Every point below the curve corresponds to a node where the bound is less than 1 mV away from the exact worst-case voltage fluctuation. From the scatter plot, it is clear that the absolute errors arising from the bounds are very small.

In terms of speed, Table II shows the speed-up of our approach over the approach of [19] for grids PG1-PG4. The data is obtained by running both engines on the same Linux machine. We can see that our approach can achieve up to 32.33× speed-up Also, to show its scalability, we run our engine on larger power grids (PG5-PG7) generated using the same user specification, as well as on the IBM power grid benchmarks. We report the resulting runtime in Table III.

TABLE I
Accuracy of the Proposed Bound

Power Grid	Time Step	Comparison to the Exact Approach	
Nodes		Upper Bound Error (μV)	Lower Bound Error (μV)
507	128 ps	529	380
1085	188 ps	769	709
2083	175 ps	712	434
4075	188 ps	991	331

TABLE II
Proposed Approach Versus Approach of [19]

Power Grid				Time Step	Speed-up
Name	Nodes	Sources	Inductors		
PG1	42,950	3,306	311	209 ps	**32.33X**
PG2	76,877	5,852	543	210 ps	**25.22X**
PG3	119,509	9,120	759	260 ps	**24.02X**
PG4	170,220	12,656	455	752 ps	**19.6 X**

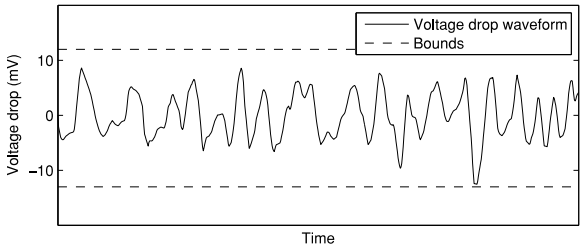

Fig. 6. Voltage drop waveform and bounds at node 4000 in the 4075-nodes grid.

TABLE III
RUNTIME OF THE PROPOSED APPROACH

Power Grid				Time Step	Runtime Breakdown				Total
Name	Nodes	Sources	Inductors		Finding the Time Step	Computing RH	Optimization	System Solve (CG)	
ibmpg1t	11,572	5,387	100	9.04 ns	0.1 sec	3.77 sec	35.03 sec	0.79 sec	**39.69 sec**
ibmpg2t	80,216	18,416	120	9.89 ns	4.76 sec	7.39 min	17.32 min	1.29 min	**26.08 min**
PG5	320,915	23,562	831	762 ps	2.18 min	54.34 min	54.16 min	7.83 min	**1.98 h**
PG6	503,489	38,220	3,119	258 ps	13.55 min	2.37 h	2.01 h	54.70 min	**5.52 h**
ibmpg3t	525,407	100,527	320	75.88 ns	4.02 sec	13.27 min	7.89 h	1.23 sec	**8.11 h**
ibmpg4t	607,752	132,972	312	3.65 ns	3.01 min	8.33 h	14..20 h	51.19 min	**23.43 h**
ibmpg5t	639,893	101,400	50	4.43 ns	2.11 sec	16.96 min	9.44 h	2.88 sec	**9.72 h**
ibmpg6t	785,004	380,742	132	9.33 ns	1.28 min	34.29 h	52.24 h	22.00 min	**3.62 d**
PG7	1,026,901	77,562	6,383	216 ps	59.88 min	10.51 h	8.00 h	3.95 h	**23.46 h**

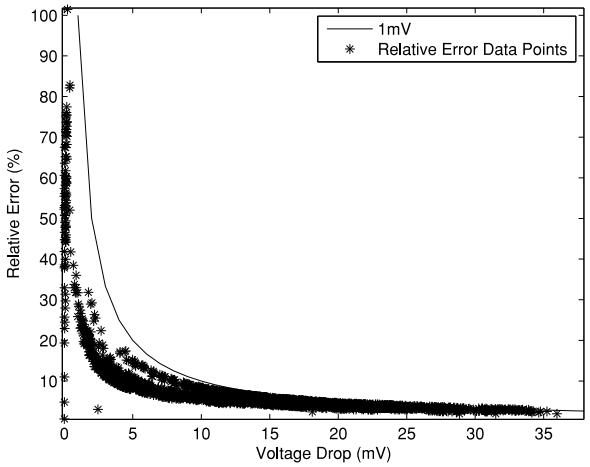

Fig. 7. Accuracy of the proposed approach (4075-node grid).

We can see that our approach computes the bounds in less than 1 day for largest power grid generated (PG7 with around one million nodes), and in around 3.62 days for the largest IBM grid tested.

By examining the runtime breakdown in Table III, one can notice that the slowest steps in the method (for most of the grids) are computing the matrix RH and performing the eopt operation. Luckily, these two steps are highly parallelizable. Computing RH requires finding columns of $D^{-1}H$, where each column can be computed independently as described in Section VII-A. Similarly, the eopt operation is an element-wise optimization where each element can be computed separately. By observing that these two steps combined take up to 80% of the total runtime, and given a cluster of multicore machines, the total runtime of our algorithm can be reduced by a large factor once properly parallelized.

IX. CONCLUSION

We described an early power grid verification approach under an RLC model using the constraint-based framework. RLC vectorless verification has always been a challenging problem because it generally requires solving a large number of optimization problems. We proposed a technique that carefully selects a time step Δt, and then computes estimates of the worst-case voltage fluctuations using that time step. Our choice of the time step is made in a way to ensure the parameter r from [19] always equal to 1. This has significant implications on the amount of work required to compute the solution. As compared to [19], our approach requires a much smaller number of LPs, and eliminates the need for any dense matrix–matrix multiplications and full matrix inverse computation. Experimentally, our approach is very accurate when compared to the exact solution, as the error is consistently under 1 mV. Moreover, we showed a speed-up of up to 32× over previous work. The technique was also shown to be much more scalable as we were able to verify grids having up to 1 million nodes.

APPENDIX A
PROOF OF LEMMA 11

We will prove that $\lim_{\Delta t\to\infty}|F(\Delta t)| = \mathbf{0}_{n\times n}$. The proof of the second identity is quite similar and is omitted. We will use the following, easy to derive, facts:

$$\lim_{\Delta t\to\infty} B = \mathbf{0}_{n_v\times n_v},\quad \lim_{\Delta t\to\infty} A^{-1} = G^{-1},\quad \lim_{\Delta t\to\infty} E = \mathbf{0}_{n_l\times n_l}.$$

Using (54) from Lemma 13, and because $M^TG^{-1}M$ is invertible by Lemma 12 (see Appendix B), we have

$$\begin{aligned}&\lim_{\Delta t\to\infty} D^{-1}B\\ &= \lim_{\Delta t\to\infty} A^{-1}\Big(I_{n_v} - M\Big(E + M^TA^{-1}M\Big)^{-1}M^TA^{-1}\Big)B\\ &= G^{-1}\Big(I_{n_v} - M\Big(M^TG^{-1}M\Big)^{-1}M^TG^{-1}\Big)\mathbf{0}_{n_v\times n_v} = \mathbf{0}_{n_v\times n_v}.\end{aligned}$$

Also, by (54), we can write

$$\begin{aligned}&\lim_{\Delta t\to\infty} -E^{-1}M^TD^{-1}B = \lim_{\Delta t\to\infty} -\Delta t L^{-1}M^TD^{-1}\frac{C}{\Delta t}\\ &= \lim_{\Delta t\to\infty} L^{-1}M^TA^{-1}\Big(I_{n_v} - M\Big(E + M^TA^{-1}M\Big)^{-1}M^TA^{-1}\Big)C\\ &= L^{-1}M^TG^{-1}\Big(I_{n_v} - M\Big(M^TG^{-1}M\Big)^{-1}M^TG^{-1}\Big)C\\ &= L^{-1}\Big(M^TG^{-1} - M^TG^{-1}\Big)C = \mathbf{0}_{n_l\times n_v}.\end{aligned}$$

Moreover, using (55) from Lemma 13

$$\begin{aligned}\lim_{\Delta t\to\infty} D^{-1}M &= \lim_{\Delta t\to\infty} A^{-1}M\Big(E + M^TA^{-1}M\Big)^{-1}E\\ &= G^{-1}M\Big(M^TG^{-1}M\Big)^{-1}\times\mathbf{0}_{n_l\times n_l} = \mathbf{0}_{n_v\times n_l}.\end{aligned}$$

And finally, $\lim_{\Delta t\to\infty}(I_{n_l} - E^{-1}M^TD^{-1}M)$

$$= \lim_{\Delta t\to\infty}\left(I_{n_l} - \Delta t L^{-1}M^TA^{-1}M\left(E + M^TA^{-1}M\right)^{-1}\frac{L}{\Delta t}\right)$$
$$= I_{n_l} - L^{-1}M^TG^{-1}M\left(M^TG^{-1}M\right)^{-1}L = I_{n_l} - I_{n_l} = \mathbf{0}_{n_l\times n_l}.$$

Therefore, considering the structure of F in (18), we have $\lim_{\Delta t\to\infty} F(\Delta t) = \mathbf{0}_{n\times n}$.

By continuity of the function $x \mapsto |x|$, we will show that $\lim_{\Delta t\to\infty}|F(\Delta t)| = \mathbf{0}_{n\times n}$ as well. Because $\lim_{\Delta t\to\infty} F(\Delta t) = \mathbf{0}_{n\times n}$, then for every $\epsilon > 0$, there exists $K > 0$ such that if $\Delta t > K$, $|F(\Delta t)| < \epsilon Q$ where Q is a matrix that has 1 at every entry. Accordingly, we have $||F(\Delta t)| - \mathbf{0}_{n\times n}| < \epsilon Q$. This being true for every ϵ, we conclude that $\lim_{\Delta t\to\infty}|F(\Delta t)| = \mathbf{0}_{n\times n}$.

By continuity of the eigenvalues of $|F(\Delta t)|$, we know that for every ϵ, there exists a $K > 0$ such that if $\Delta t > K$ then $|\lambda_i| < \epsilon$ for every eigenvalue λ_i of $|F(\Delta t)|$. This is due to that fact that all eigenvalues of $\mathbf{0}_{n\times n}$ are 0. This implies that $\rho(|F(\Delta t)|) < \epsilon$. This, being true for every ϵ, leads to $\lim_{\Delta t\to\infty}\rho(|F(\Delta t)|) = 0$.

APPENDIX B
LEMMAS 12 AND 13

Here we present and prove Lemmas 12 and 13 used in the proof of Lemma 11.

Lemma 12: The matrix $M^TG^{-1}M$ is invertible.

Proof: Recall that M is the incidence matrix that specifies the locations of the inductors in the grid.

Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a directed graph whose vertex set $\mathcal{V}$ contains all the nodes in the grid including the ground node, and whose edge set $\mathcal{E}$ is constructed as follows: for every inductor in the grid, an edge is added between the two nodes that are connected to the inductor; the direction of the edge is the same as the reference the direction of current through the inductor. Because we assume that no purely inductive loop exists in the grid, we can deduce that $\mathcal{G}$ is acyclic. Accordingly, $\mathcal{G}$ is a directed acyclic graph. Let $M_{\mathcal{G}}$ be the incidence matrix of $\mathcal{G}$, and $M'_{\mathcal{G}}$ be the matrix obtained by deleting the row corresponding to the ground node in the matrix $M_{\mathcal{G}}$. It is known that the columns of $M'_{\mathcal{G}}$ are linearly independent [33]. One can verify that M and $M'_{\mathcal{G}}$ have the same set of columns (possibly in a different order), so that the columns of M are linearly independent, meaning M has a full rank. Now, consider the matrix

$$S = \begin{bmatrix} G & M \\ M^T & 0 \end{bmatrix}.$$

Because M has full rank, and because G is invertible, we conclude, from [34], that S is invertible. But S is invertible if and only if its Schur complement $-M^TG^{-1}M$ is invertible [26]. Knowing that S is invertible, it directly follows that $M^TG^{-1}M$ is invertible. ■

Lemma 13: We have

$$D^{-1} = A^{-1}\left(I_{n_v} - M\left(E + M^TA^{-1}M\right)^{-1}M^TA^{-1}\right) \quad (54)$$

$$D^{-1}M = A^{-1}M\left(E + M^TA^{-1}M\right)^{-1}E. \quad (55)$$

Proof: Notice that $D = A + ME^{-1}M^T$ so that D^{-1} is the inverse of a sum of matrices. In [35], the authors showed that such a sum can be expressed as follows:

$$D^{-1} = A^{-1} - A^{-1}M\left(E + M^TA^{-1}M\right)^{-1}M^TA^{-1}$$

which leads directly to (54). Moreover, notice that

$$\begin{aligned}
D^{-1}M &= \left(A + ME^{-1}M^T\right)^{-1}M \\
&= \left[\left(I + ME^{-1}M^TA^{-1}\right)A\right]^{-1}M \\
&= A^{-1}\left(I + ME^{-1}M^TA^{-1}\right)^{-1}M \\
&= A^{-1}\left(I + ME^{-1}M^TA^{-1}\right)^{-1}M \\
&\quad\times\left(I + E^{-1}M^TA^{-1}M\right)\left(I + E^{-1}M^TA^{-1}M\right)^{-1} \\
&= A^{-1}\left(I + ME^{-1}M^TA^{-1}\right)^{-1} \\
&\quad\times\left(M + ME^{-1}M^TA^{-1}M\right)\left(I + E^{-1}M^TA^{-1}M\right)^{-1} \\
&= A^{-1}\left(I + ME^{-1}M^TA^{-1}\right)^{-1} \\
&\quad\times\left(I + ME^{-1}M^TA^{-1}\right)M\left(I + E^{-1}M^TA^{-1}M\right)^{-1} \\
&= A^{-1}M\left(I + E^{-1}M^TA^{-1}M\right)^{-1}E^{-1}E \\
&= A^{-1}M\left(E\left(I + E^{-1}M^TA^{-1}M\right)\right)^{-1}E \\
&= A^{-1}M\left(E + M^TA^{-1}M\right)^{-1}E
\end{aligned}$$

as required in (55). ■

REFERENCES

[1] Z. Zeng, T. Xu, Z. Feng, and P. Li, "Fast static analysis of power grids: Algorithms and implementations," in *Proc. ACM/IEEE Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2011, pp. 488–493.

[2] C.-H. Chou *et al.*, "On the preconditioner of conjugate gradient method—A power grid simulation perspective," in *Proc. ACM/IEEE Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2011, pp. 494–497.

[3] H. Qian, S. R. Nassif, and S. S. Sapatnekar, "Power grid analysis using random walks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 8, pp. 1204–1224, Aug. 2005.

[4] B. Boghrati and S. S. Sapatnekar, "Incremental analysis of power grids using backward random walks," *ACM Trans. Design Autom. Electron. Syst.*, vol. 19, no. 3, Jun. 2014, Art. no. 31.

[5] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "A multigrid-like technique for power grid analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 10, pp. 1148–1160, Oct. 2002.

[6] M. Zhao, R. V. Panda, S. S. Sapatnekar, and D. Blaauw, "Hierarchical analysis of power distribution networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 2, pp. 159–168, Feb. 2002.

[7] J. Yang, Z. Li, Y. Cai, and Q. Zhou, "PowerRush: Efficient transient simulation for power grid analysis," in *Proc. ACM/IEEE Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2012, pp. 653–659.

[8] T. Yu and M. D. F. Wong, "PGT_SOLVER: An efficient solver for power grid transient analysis," in *Proc. ACM/IEEE Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2012, pp. 647–652.

[9] X. Xiong and J. Wang, "Parallel forward and back substitution for efficient power grid simulation," in *Proc. ACM/IEEE Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2012, pp. 660–663.

[10] H. Zhuang, S.-H. Weng, J.-H. Lin, and C.-K. Cheng, "MATEX: A distributed framework for transient simulation of power distribution networks," in *Proc. IEEE/ACM Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2014, pp. 1–6.
[11] D. Kouroussis and F. N. Najm, "A static pattern-independent technique for power grid voltage integrity verification," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, Anaheim, CA, USA, Jun. 2003, pp. 99–104.
[12] N. H. A. Ghani and F. N. Najm, "Fast vectorless power grid verification using an approximate inverse technique," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2009, pp. 184–189.
[13] X. Xiong and J. Wang, "A hierarchical matrix inversion algorithm for vectorless power grid verification," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2010, pp. 543–550.
[14] X. Xiong and J. Wang, "Dual algorithms for vectorless power grid verification under linear current constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 10, pp. 1469–1482, Oct. 2011.
[15] N. H. A. Ghani and F. N. Najm, "Power grid verification using node and branch dominance," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, New York, NY, USA, 2011, pp. 682–687.
[16] H. Yu, Y. Shi, and L. He, "Fast analysis of structured power grid by triangularization based structure preserving model order reduction," in *Proc. IEEE/ACM Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2006, pp. 205–210.
[17] Abhishek and F. N. Najm, "Incremental power grid verification," in *Proc. IEEE/ACM Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2012, pp. 151–156.
[18] N. H. A. Ghani and F. N. Najm, "Handling inductance in early power grid verification," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2006, pp. 127–134.
[19] N. H. A. Ghani and F. N. Najm, "Fast vectorless power grid verification under an RLC model," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 5, pp. 691–703, May 2011.
[20] X. Xiong and J. Wang, "Verifying RLC power grids with transient current constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 7, pp. 1059–1071, Jul. 2013.
[21] X. Xiong and J. Wang, "Vectorless transient power grid verification: A case study with IBM benchmarks," in *Proc. IEEE Symp. Electromagn. Compat. Signal Int. (EMC&SI)*, Santa Clara, CA, USA, Mar. 2015, pp. 271–276.
[22] S. Pant and E. Chiprout, "Power grid physics and implications for CAD," in *Proc. IEEE/ACM Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2006, pp. 199–204.
[23] S. R. Nassif. *IBM Power Grid Benchmarks*. Accessed on Mar. 2016. [Online]. Available: http://dropzone.tamu.edu/∼pli/PGBench
[24] *HSPICE User Guide: Basic Simulation and Analysis*, Synopsys, San Jose, CA, USA, Sep. 2014.
[25] F. N. Najm, *Circuit Simulation*. Hoboken, NJ, USA: Wiley, 2010.
[26] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA, USA: SIAM, 2003.
[27] T.-H. Chen, C. Luk, and C. C.-P. Chen, "INDUCTWISE: Inductance-wise interconnect simulator and extractor," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 7, pp. 884–894, Jul. 2003.
[28] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. Chichester, U.K.: Wiley, 1991.
[29] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 1st ed. Cambridge, U.K.: Cambridge Univ. Press, 1990.
[30] M. T. Heath, *Scientific Computing: An Introductory Survey*, 2nd ed. Boston, MA, USA: McGraw-Hill, 2002.
[31] R. J. Wood and M. J. O'Neill, "An always convergent method for finding the spectral radius of an irreducible non-negative matrix," *ANZIAM J.*, vol. 45, pp. C474–C485, Jan. 2004.
[32] *The MOSEK Optimization Software*. Accessed on May 12, 2016. [Online]. Available: http://www.mosek.com
[33] R. B. Bapat, *Graphs and Matrices*, 1st ed. New York, NY, USA: Springer, 2010.
[34] M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," *Acta Numerica*, vol. 14, pp. 1–137, May 2005.
[35] H. V. Henderson and S. R. Searle, "On deriving the inverse of a sum of matrices," *SIAM Rev.*, vol. 23, no. 1, pp. 53–60, Jan. 1981.

Mohammad Fawaz (S'08) received the B.E. degree in computer and communications engineering (with high distinction) from the American University of Beirut, Beirut, Lebanon, in 2011, and the M.A.Sc. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2013, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering.

His current research interests include computer-aided design for integrated circuits with a focus on the verification, reliability, and analysis of power grids.

Farid N. Najm (S'85–M'89–SM'96–F'03) received the B.E. degree in electrical engineering from the American University of Beirut, Beirut, Lebanon, in 1983, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign (UIUC), Champaign, IL, USA, in 1989.

He is a Professor with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering (ECE), University of Toronto, Toronto, ON, Canada. From 1989 to 1992, he was with Texas Instruments, Dallas, TX, USA. He then joined the ECE Department, UIUC, as an Assistant Professor and became an Associate Professor, in 1997. In 1999, he joined the ECE Department, University of Toronto, where he is currently a Professor and the Chair. In 2010, he has authored the book entitled *Circuit Simulation* (Wiley, New York). His current research interests include CAD for very large scale integration, with an emphasis on circuit level issues related to power, timing, variability, and reliability.

Dr. Najm was a recipient of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS Best Paper Award, the National Science Foundation (NSF) Research Initiation Award, the NSF CAREER Award, and the Design Automation Conference (DAC) Prolific Author Award. He was an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, from 2001 to 2009 and THE IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, from 1997 to 2002. He served on the executive committee of the International Symposium on Low-Power Electronics and Design (ISLPED), from 1999 to 2013, and served as the TPC Chair and the General Chair for ISLPED. He has also served on the technical committees of various conferences, including International Conference on Computer-Aided Design, DAC, Custom Integrated Circuits Conference, International Symposium on Quality Electronic Design, and ISLPED. He is a fellow of the Canadian Academy of Engineering.