An Algorithmic Framework for Efficient Large-Scale Circuit Simulation Using Exponential Integrators

Hao Zhuang[1], Wenjian Yu[2], Ilgweon Kang[1], Xinan Wang[1], and Chung-Kuan Cheng[1]
[1]Department of Computer Science & Engineering, University of California, San Diego, CA, USA
[2]Department of Computer Science & Technology, Tsinghua University, Beijing, China
hao.zhuang@cs.ucsd.edu, yu-wj@tsinghua.edu.cn, {igkang, xinan, ckcheng}@ucsd.edu

***Abstract*—We propose an efficient algorithmic framework for time-domain circuit simulation using exponential integrators. This work addresses several critical issues exposed by previous matrix exponential based circuit simulation research, and makes it capable of simulating stiff nonlinear circuit system at a large scale. In this framework, the system's nonlinearity is treated with exponential Rosenbrock-Euler formulation. The matrix exponential and vector product is computed using invert Krylov subspace method. Our proposed method has several distinguished advantages over conventional formulations (e.g., the well-known backward Euler with Newton-Raphson method). The matrix factorization is performed only for the conductance/resistance matrix G, without being performed for the combinations of the capacitance/inductance matrix C and matrix G, which are used in traditional implicit formulations. Furthermore, due to the explicit nature of our formulation, we do not need to repeat LU decompositions when adjusting the length of time steps for error controls. Our algorithm is better suited to solving tightly coupled post-layout circuits in the pursuit for full-chip simulation. Our experimental results validate the advantages of our framework.**

Categories and Subject Descriptors
B.7.2 [**Integrated Circuits**]: Design Aids - *simulation*
General Terms
Algorithms, Design, Performance
Keywords
Circuit Simulation, Transient Simulation, Exponential Integrators

I. Introduction

SPICE-like transistor-level circuit simulation [1]–[3] of integrated circuits is indispensable during the cycle of very large scale integration (VLSI) designs. It is crucial to cost-efficient production of VLSI chips. Smaller process geometries, larger designs as well as tighter design margins translate to the need for more accurate large-scale circuit simulation, e.g., post-layout simulations with more detailed parasitic couplings [4]–[6]. With advancing technologies, three dimensional IC structures, and increasing complexities of system designs, the numbers of chip components are easily more than millions [7], [8], which make simulation tasks very time-consuming. Finding effective algorithms still remains challenging and has always been an important research topic for several decades [1]–[3].

Time-domain circuit simulation algorithm relies on numerical integration to solve differential algebraic equations (DAE). To capture circuit's transient behaviors, numerical integration is computed step by step till the end of whole simulation time span. Conventional numerical integration formulations are forward Euler (FE), backward Euler (BE), Trapezoidal (TR), Gear's and multi-step methods [1]–[3]. Despite explicit formulation, such as FE, avoids solving linear system, the time steps are usually restricted for ensuring numerical stability [2], [3], [9], [10]. In general, VLSI circuits form stiff and nonlinear dynamical systems. Therefore, the implicit formulations, e.g., BE, are much more stable, robust and widely deployed in VLSI circuit simulators. Based on the implicit formulation of such nonlinear system, Newton-Raphson (NR) iterations are typically adopted to obtain solutions. Each NR iteration needs to linearize system and then solves the linear system. During this process, direct solvers [11], [12] have been favored and applied because of their robustness and ease of use. However, it is known that direct solvers, e.g., LU decomposition, have super-linear computational complexities and very expensive to simulate large-scale and strongly coupled circuit systems (the cost of LU decomposition can approach the worst case $O(n^3)$ [4], [5]). It is crucial to devise efficient algorithms to reduce the numbers of expensive LU operations and NR iterations in the circuit simulation. For example, leveraging the explicit formulations is a research direction [9], [10].

Above formulations are all categorized into the low order approximation of DAE, which is with the order typically lower than ten [3]. The error of those formulations are proportional to the step size due to local truncation error (LTE). Beyond those conventional low order schemes, a high order matrix exponential based time integration kernel [3] has been recently triggering academic researchers' interests because of the progress of efficient matrix function computation using Krylov subspace methods [13]–[16]. In VLSI CAD and EDA research, this exponential integration kernel has been applied in time domain electromagnetic and technology semiconductor device simulation [17], power distribution network analysis [18], [19] as well as general circuit simulation [20], [21]. Those frameworks provide analytical solution for transient simulation of linear system, and better properties in numerical accuracy, stability and time step controlling, etc [3], [16], [18], [19]. However, one drawback of previous works for matrix exponential based circuit simulation [20] is the slow convergence rate for matrix exponential and vector product (MEVP) by using standard Krylov subspace [17], [20]. In addition, the nonlinear formulation still requires NR iterations [17], [20], which does not fully utilize the explicit nature from matrix exponential formulation [14].

In this work, two major processes are devised to address above challenges. First, we propose a new matrix exponential based framework using exponential Rosenbrock-Euler integration formulation [15], [16], which well suits nonlinear dynamical differential equation problems and preserves explicit features. Then, we also design error control scheme for adaptive time stepping. Second, we deploy invert Krylov subspace method [19] to compute MEVP in an efficient manner. The major contributions are listed as follows.

- By invert Krylov subspace strategy, we improve the convergence rate for the MEVP computation, the key operation within the matrix exponential based method. Besides, this approach also removes the regularization process [20], [22], which would be impractical for large designs with singular matrix C. Therefore, this approach enables the application for the simulation of large stiff VLSI designs.
- Proposed invert Krylov subspace strategy removes C from

DAC '15, June 07 - 11, 2015, San Francisco, CA, USA

http://dx.doi.org/10.1145/2744769.2744793

matrix factorization processes. Building invert Krylov subspace bases only needs to factorize G, which is much sparser and simpler than C in strongly coupled post-layout circuits. In contrast, conventional implicit methods require LU decomposition of the combinations of G and C. Therefore, our method is better suited to handle those strongly coupled post-layout circuits in the pursuit for full-chip simulation.

- We devise exponential Rosenbrock-Euler formulation for stable explicit DAE solver, as well as correction term on top of that to further improve the accuracy. The stability of the explicit formulation is preserved by the high-order approximation of exponential operator [13], [15], [16]. Thus, it takes only one LU decomposition per time step, while conventional methods, e.g., BE with NR (BENR), require at least two times of LU to verify the convergence.
- Moreover, our method does not need to repeat LU when we adjust the length of time steps for error controls. It is because our matrix exponential based explicit formulation and (time-step) scaling-invariant property of Krylov subspace [13], [20]. On the contrary, the low order approximation schemes force time step embedded in the linear matrix and conduct matrix factorization. Once the time step is adjusted, LU is unavoidable in order to solve the new linear system.
- We test our framework with baseline BENR against our large-scale circuits. For some challenging test cases, our framework can achieve speedups by magnitude, and even manage to finish them when conventional counterpart fails.

The remainder of this paper is organized as follows. Sec. II introduces the background of nonlinear circuit simulation. Sec. III presents our new circuit simulation framework that utilizes exponential Rosenbrock-Euler method and related formulations. Sec. IV illustrates the MEVP computation uses invert Krylov subspace method. Sec. V shows experimental results and validates our framework. Sec. VI concludes this paper.

II. Background

Given a circuit netlist and device models, time-domain simulation of general nonlinear electronic circuit is formulated as

$$\frac{d\vec{q}(\vec{x}(t))}{dt} + \vec{f}(\vec{x}(t)) = B\vec{u}(t), \tag{1}$$

where $\vec{x}(t) \in \mathbb{R}^{n\times 1}$ denotes nodal voltages and branch currents and n is the length of vector. $\vec{q} \in \mathbb{R}^{n\times 1}$ and $\vec{f} \in \mathbb{R}^{n\times 1}$ represent the charge/flux and current/voltage terms, respectively. $\vec{u}(t)$ is a vector representing all the external excitations at time t; B is an incident matrix that inserts those signals to the system.

A. Circuit Simulation Using Low Order Integration Schemes

The conventional numerical approaches discretized Eq. (1) with time step size $h_k = t_{k+1} - t_k$. To compute the solution $\vec{x}_{k+1}$ at t_{k+1}, we start from an initial solution $\vec{x}_k$. The well-known BE, which is a stable implicit integration scheme, leads to

$$\begin{aligned} C(\vec{x}_{k+1})\frac{\vec{x}_{k+1} - \vec{x}_k}{h_k} + G(\vec{x}_{k+1})\vec{x}_{k+1} \\ = B\vec{u}(t_{k+1}) + \vec{F}(\vec{x}_{k+1}), \end{aligned} \tag{2}$$

where $C(\vec{x}_k) \in \mathbb{R}^{n\times n}$ is a matrix of capacitances and inductances linearized at $\vec{x}_k$; $G(\vec{x}_k) \in \mathbb{R}^{n\times n}$ represents the linearized resistances and conductances as well as the incidence between voltages and currents. F collects the nonlinear dynamics from transistor models, such as BSIM3. Usually NR is a widely adopted method of solving this implicit formulation Eq. (2). Each NR iteration, direct solver (e.g., LU decomposition) is applied to solve

$$\frac{\partial \vec{T}(\vec{x}^i)}{\partial \vec{x}}(\vec{x}^{i+1} - \vec{x}^i) = -T(\vec{x}^i) \tag{3}$$

until it is convergent, which means the difference of the solution from i-th iteration $\vec{x}^i$ and $\vec{x}^{i+1}$ is "small enough". $\vec{T}(\vec{x}) = C(\vec{x})\frac{\vec{x}-\vec{x}_k}{h_k} + G(\vec{x})\vec{x} - B\vec{u}(t_k + h_k) - \vec{F}(\vec{x})$ corresponds to Eq. (2), and $\frac{\partial \vec{T}(\vec{x}^i)}{\partial \vec{x}} \in \mathbb{R}^{n\times n}$ is the Jacobian matrix of $\vec{T}$. This approach usually serves a *de facto* method in industrial tools and academic research, which is called as *BENR* in this paper. To solve Eq. (1), we can also deploy other low order implicit discretized forms, such as TR and Gear's, with NR.

Due to the low order implicit integration scheme, Eq. (3) embeds time step h_k in $\vec{T}$ and $\frac{\partial \vec{T}(\vec{x}^i)}{\partial \vec{x}}$ (following the practice of SPICE). If the estimated LTE violates numerical error budget, h_k will be reduced. Then new NR iterations for $\vec{x}_{k+1}$ are re-launched with the updated h_k. Besides, C always exists in Jacobian matrix $\frac{\partial \vec{T}(\vec{x}^i)}{\partial \vec{x}}$. Large volume of non-zeros of C are introduced from the post-layout parasitics extraction [23]–[25], resulting to huge computational challenges for the capability of numerical integration algorithms [6] and model order reductions [26]. In addition, the non-zero fill-ins of off-diagonal terms in C and G are usually mutually exclusive in VLSI circuits. For example, the matrices in Fig. 1 are from post-extraction of a design **FreeCPU** [23]. We notice that the distribution of non-zeros in C (Fig. 1(a)) is much more complicated than G (Fig. 1(b)). After LU decompositions of those matrices, the factorized matrices Fig. 1(c) and Fig. 1(d) of C, Fig. 1(g) and Fig. 1(h) of $\left(\frac{C}{h} + G\right)$ also contain much larger nnz (number of non-zeros) than Fig. 1(e) and Fig. 1(f) of G.

To address above challenges, we use exponential integrators [14] and architect a framework with stable explicit formulation. The next sub-section briefs previous progresses of circuit simulation using related exponential integration methods and the major problems, which keep them from the large-scale circuit simulation.

B. Circuit Simulation Using Exponential Integrators

Recently, many works are proposed to solve time-domain simulation problem using exponential integrators [17]–[21]. Chen *et al.* [17] and Weng *et al.* [20] adopted a technique developed in [27], which decouples the linear and nonlinear terms by approximating the integrand in Eq. (4) with Lagrange polynomial.

$$\begin{aligned} \vec{x}_{k+1} \;=\; & e^{h_k J(\vec{x}_k)}\vec{x}_k + \\ & \int_0^{h_k} e^{(h_k-\tau)J(\vec{x}_k)}C^{-1}(\vec{x}_k)\left(\vec{F}(\vec{x}_k+\tau) + B\vec{u}(t_k+\tau)\right)d\tau \end{aligned} \tag{4}$$

where $J(\vec{x}) = -C^{-1}(x)G(x)$. The second-order approximation yields $\vec{x}_{k+1}$ as the solution of a nonlinear equation, which again can be solved by NR using Eq. (3). The Jacobian matrix in Eq. (3) is $C(\vec{x}^i) + \frac{h}{2}\frac{\partial \vec{F}(\vec{x}^i)}{\partial \vec{x}}$. More details can be found in [17], [21]. Therefore, it is similar to conventional method in Sec. II-A. During NR iterations, the matrix factorization involves C in the Jacobian matrix. Then it will suffer from the runtime degradation when C is relatively large and complicated.

Another problem comes from the key operation of $e^J\vec{x}$, the matrix exponential-and-vector product (MEVP), where J is a $n \times n$ matrix. MEVP is computed by *standard Krylov subspace* [13], [17], [20], [21]. Such Krylov subspace is constructed as

$$\mathrm{K_m}(J, \vec{x}) := \mathrm{span}\{\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_m\} = \mathrm{span}\{\vec{x}, J\vec{x}, \cdots, J^{m-1}\vec{x}\}. \tag{5}$$

An $n \times m$ orthonormal basis and an $(m+1) \times m$ upper Hessenberg matrix H for the Krylov subspace $\mathrm{K_m}$ are first constructed by Arnoldi process [17], [20], [21]. Then MEVP is computed via

$$e^{hJ}v \approx \|v\|V_m e^{hH_m}e_1 \tag{6}$$

where $H_m = H(1:m, 1:m)$, $e_1 \in \mathbb{R}^{m\times 1}$ and e_1 is the first unit vector. When A is mildly stiff, H_m is usually much smaller

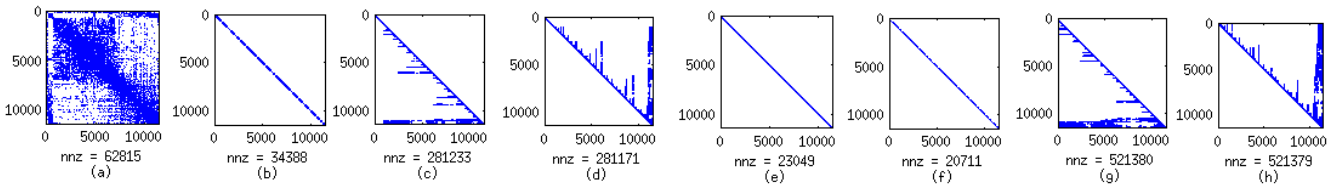

Fig. 1: Visualization of post-extraction matrices' non-zero elements distributions from a design **FreeCPU** [23], the sizes of matrix are 11417×11417, which are obtained from SPEF extracted by industrial tool *Synopsys Star-RCXT*. nnz is the number of non-zeros in the matrix. (a) Extracted capacitance matrix C (non-zero entries distribute widely in the matrix). (b) Extracted conductance matrix G (there are many off-diagonal non-zeros in the matrix, but the bandwidth is much smaller than C). (c) Lower triangular matrix L_C and (d) Upper triangular matrix U_C of $LU\_decompose(C)$; (e) Lower triangular matrix L_G and (f) Upper triangular matrix U_G of $LU\_decompose(G)$; (g) Lower triangular matrix $L_{\frac{C}{h}+G}$ and (h) Upper triangular matrix $U_{\frac{C}{h}+G}$ of $LU\_decompose(\frac{C}{h}+G)$. The function of $LU\_decompose$ uses MATLAB2013a UMFPACK. L_G and U_G contain much smaller nnz than L_C, U_C, $L_{\frac{C}{h}+G}$ and $U_{\frac{C}{h}+G}$.

than original A [20]. However, when the values in C vary in magnitudes caused by stiff circuit system, standard Krylov subspace demands large dimension of subspace to approximate MEVP, then degrades performance of matrix exponential-based circuit simulation. Such phenomenon has been observed in power distribution network simulation using matrix exponential framework [18], [19]. Besides, C should not be singular during the process of standard Krylov subspace strategy. Otherwise, related regularization process [20], [22] is required, which is time-consuming for large-scale designs.

III. Circuit Simulation Using Exponential Integrators Based on Rosenbrock-Euler Formulation

In contrast to standard low order and previous matrix exponential integration schemes, our framework adopts exponential Rosenbrock-Euler method, and leads to a stable explicit method, which does not require NR iterations [15]. To simplify the derivations from exponential Rosenbrock-Euler to SPICE-like formulation, we consider a non-autonomous ordinary differential equation (ODE) system,

$$\frac{d\vec{x}(t)}{dt} = \vec{g}(\vec{x}, \vec{u}, t). \tag{7}$$

Exponential Rosenbrock-Euler method is derived to compute $\vec{x}_{k+1}$ with step size h_k as follows,

$$\vec{x}_{k+1} = \vec{x}_k + h_k\phi_1(h_kJ_k)\vec{g}(\vec{x}_k, \vec{u}, t_k) + h_k^2\phi_2(h_kJ_k)\vec{b}_k \tag{8}$$

where J_k denotes the Jacobian matrix of $\vec{g}$, and $\vec{b}_k = \frac{\partial \vec{g}}{\partial t}(\vec{x}_k, \vec{u}, t_k)$. [14]–[16]. $\phi_i(z)$ describes ODE's time evolution, where $\phi_i(z) = \int_0^1 e^{z(1-s)}\frac{s^{i-1}}{(i-1)!}ds \quad (i \geq 1)$. We have $\phi_0(h_kJ_k) = e^{h_kJ_k}$ and $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix, then

$$\phi_1(h_kJ_k) = \frac{e^{h_kJ_k} - I_n}{h_kJ_k}, \quad \phi_2(h_kJ_k) = \frac{e^{h_kJ_k} - I_n}{h_k^2J_k^2} - \frac{I_n}{h_kJ_k}. \tag{9}$$

More details can be found in the works of Hochbruck and Ostermann [14], [15]. The advantage of this formulation is the explicit nature and the superior stable region (in the entire complex plane) than the class of low order integrations schemes. Therefore ϕ_i functions permit a large value for the step size h_k with guaranteed stability [14]–[16].

To utilize above formulation in SPICE-like circuit simulation, we apply the chain rule to Eq. (1),

$$\frac{d\vec{q}(\vec{x}(t))}{d\vec{x}} \cdot \frac{d\vec{x}(t)}{dt} = B\vec{u}(t) - f(\vec{x}). \tag{10}$$

Eq. (10) is linearized at the state $\vec{x}_k$, when $\frac{d\vec{q}(\vec{x}_k)}{d\vec{x}} = C(\vec{x}_k) = C_k$. We obtain the equivalent format for the non-autonomous dynamical system

$$\frac{d\vec{x}(t)}{dt} = \vec{g}(\vec{x}, \vec{u}, t) = J_k\vec{x}(t) + C_k^{-1}\left(\vec{F}(\vec{x}(t)) + B\vec{u}(t)\right) \tag{11}$$

where $J_k = -C_k^{-1}G_k$ and $G_k = G(\vec{x}_k)$. The circuit system response is computed by high order function $h_k\phi_1(h_kJ_k)\vec{g}_k$, where

$$\vec{g}_k = J_k\vec{x}_k + C_k^{-1}\left(\vec{F}_k + B\vec{u}(t_k)\right), \tag{12}$$

and $\vec{F}_k = F(\vec{x}_k)$. Furthermore, we assume $\vec{u}(t)$ is a piecewise-linear function for $t \in [t_k, t_{k+1}]$ in VLSI designs, so the contribution from external excitations can also be captured by $h_k^2\phi_2(h_kJ_k)\vec{b}_k$ [14]–[16], where

$$\vec{b}_k = C_k^{-1}B\frac{\vec{u}(t_{k+1}) - \vec{u}(t_k)}{h_k} \tag{13}$$

The next time step solution $\vec{x}_{k+1}$ with h_k is

$$\vec{x}_{k+1}(h_k) = \vec{x}_k + h_k\phi_1(h_kJ_k)\vec{g}_k + h_k^2\phi_2(h_kJ_k)\vec{b}_k. \tag{14}$$

Note the denominator of ϕ_i in Eq. (9) always cancels out C_k^{-1} in Eq. (12) and Eq. (13). Hence, we avoid the matrix factorization of C_k.

A. Local Nonlinear Error Estimator

The error estimator is an important ingredient of adaptive time-stepping for the nonlinear systems Based on the notion proposed by Caliari and Ostermann [16], we devise a formula to estimate local nonlinear truncation error to control the step size,

$$e_{rr}(\vec{x}_{k+1}, \vec{x}_k) = h_k\phi_1(h_kJ_k)C_k^{-1}\Delta\vec{F}_k = -\left(e^{h_kJ_k} - I_n\right)G_k^{-1}\Delta\vec{F}_k, \tag{15}$$

where $\Delta\vec{F}_k = \vec{F}_{k+1} - \vec{F}_k$. Intuitively, Eq. (15) represents the response changes inside the nonlinear system along time evolutions. When the strong nonlinear phenomenon is detected, the absolute value of e_{rr} becomes large and shrinks the step size to obtain an accurate enough solution.

B. Modified Correction Term for Exponential Rosenbrock-Euler Method

Furthermore, we can reuse the intermediate results $\Delta\vec{F}_k$ (after device evaluations at $\vec{x}_{k+1}$) of Eq. (15) to improve the accuracy of solution [16] The correction strategy is designed by utilizing ϕ_2 [16],

$$\vec{D}_k = \gamma h_k\phi_2(h_kJ_k)C_k^{-1}\Delta\vec{F}_k, \tag{16}$$

where γ is constant and has several choices [14], [16]. Within the time step, by adding this correction term, the corrected solution $\vec{x}_{k+1,c}$ is

$$\vec{x}_{k+1,c} = \vec{x}_{k+1} - \vec{D}_k. \tag{17}$$

In order to obtain such more accurate $\vec{x}_{k+1,c}$ by reusing $\Delta\vec{F}_k$, we need extra computations, including Krylov subspace generations.

IV. MEVP Using Invert Krylov Subspace

Krylov subspace method enables the time-domain circuit simulation algorithm to use exponential integrators. The key of efficient Krylov subspace based matrix function evaluation is keeping the size of the Krylov basis m small so that calculation of Eq. (6) is cheap. However, using standard Krylov subspace for MEVP $e^J\vec{v}$ with stiff $J = -C^{-1}G$ is not efficient enough due to the demand of large dimensional subspace. Our interpretation of the inefficiency is standard Krylov subspace method tends to oversample eigenvalues with large magnitudes in the spectrum of J, which are not crucial players to define dynamical behaviors [18], [19].

Another drawback of standard Krylov subspace comes from matrix factorization of C for subspace generations. To generate basis v_{i+1}, we need to solve $C\vec{v}_i = \vec{b}$, where $\vec{b} = -G\vec{v}_{i-1}$, $1 \le i \le m$, in Eq. (5). C may be singular, relatively complicated or denser than G (Fig. 1) [23]–[25]. If C is singular, we should apply regularization process to make a non-singular C. It is time-consuming and impractical for large designs [20], [22].

A. Invert Krylov Subspace Method

We adopt a technique called as *invert Krylov subspace* [19] to avoid the matrix factorization of C. It also helps capture important eigenvalues for exponential matrix function and accelerate MEVP evaluation. In [19], invert Krylov subspace method is the second on convergent rate and the length of step-size h after rational Krylov subspace method. However, its basis generation can be cheaper for general nonlinear circuit simulation problems. Besides, the properties fit well with nonlinear dynamical systems where the step-size is limited by the nonlinearity of the devices. Invert Krylov subspace is constructed as follows,

$$\begin{aligned} \mathrm{K_m}(J^{-1}, \vec{v}) &= \mathrm{span}\{v, J^{-1}\vec{v}, \cdots, J^{-(m-1)}\vec{v}\} \quad (18)\\ &= \mathrm{span}\{\vec{v}, -G^{-1}C\vec{v}, \cdots, (-G^{-1}C)^{(m-1)}\vec{v}\}. \end{aligned}$$

During the process, only G is required for matrix factorization. Therefore, we are able to gain computational advantages when C contains large number of non-zeros to describe many parasitics and strongly coupling effects.

$$J^{-1}V_m = V_m H_m + h_{m+1,m}\vec{v}_{m+1}\vec{e}_m^{\mathbf{T}} \quad (19)$$

MEVP of $x(t) = e^{tJ}v = e^{-tC^{-1}G}\vec{v}$ is computed as

$$x_m(t) = \|\vec{v}\| V e^{tH_m^{-1}}\vec{e}_1. \quad (20)$$

Error estimation of MEVP is derived to determine dimension size of Krylov subspace for MEVP. We have the derivative of $\vec{x}_m(t)$

$$\frac{d\vec{x}_m(t)}{dt} = \|\vec{v}\| V_m H_m^{-1} e^{tH_m^{-1}}\vec{e}_1. \quad (21)$$

The residual is the difference between $\frac{d\vec{x}_m(t)}{dt}$ and $J\vec{x}_m(t)$. Then, we use the following residual to check Kirchhoff's current and voltage laws (KCL/KVL).

$$\begin{aligned} r_m(t) &= C\left(\frac{d\vec{x}_m(t)}{dt} - J\vec{x}_m(t)\right)\\ &= \|\vec{v}\| C\left(V_m H_m^{-1} e^{tH_m^{-1}} e_1 - JV_m e^{tH_m^{-1}}\vec{e}_1\right)\\ &= -\|\vec{v}\| h_{m+1,m} G\vec{v}_{m+1} e^{\mathbf{T}} H_m^{-1} e^{tH_m^{-1}}\vec{e}_1 \quad (22) \end{aligned}$$

We design *MEVP_IKS* in **Algorithm 1** and embed error estimator above as a convergence criteria to terminate Arnoldi process. Then, given the initial state vector $\vec{v}$, time step size h_k and error budget ϵ, $\phi_1\vec{v}$ and $\phi_2\vec{v}$ are computed, respectively, as $\phi_1^{(e)}$ and $\phi_2^{(e)}$, where

$$\phi_1^{(e)}(C_k, G_k, \vec{v}, \epsilon, h_k) = \frac{1}{h_k J_k}\vec{m}_{evp} - \frac{1}{h_k J_k}\vec{v}$$

Algorithm 1: *MEVP_IKS*: Arnoldi Algorithm of Invert Krylov Subspace Method for MEVP $e^{-hC^{-1}G}\vec{v}$

Input: $C, G, \vec{v}, \epsilon, h$
Output: $\vec{m}_{evp}, V_m, H_m, m$

```
1  v_1 = v / ||v||;
2  for j = 1 : m do
3      Solve -Gw = Cv_j and obtain w; for i = 1 : j do
4          h_{i,j} = w^T v_i;
5          w = w - h_{i,j} v_i;
6      end
7      h_{j+1,j} = ||w||;
8      v_{j+1} = w / h_{j+1,j};
9      if ||r_m(h)|| < ε using Eq. (22) then
10         m = j;
11         break;
12     end
13 end
14 m_evp = ||v|| V_m e^{h H_m^{-1}} e_1 ;
```

$$\phi_2^{(e)}(C_k, G_k, \vec{v}, \epsilon, h_k) = \frac{1}{h_k^2 J_k^2}\vec{m}_{evp} - \frac{1}{h_k^2 J_k^2}\vec{v} - \frac{1}{h_k J_k}\vec{v} \quad (23)$$

where $\vec{m}_{vep}$ is obtained from *MEVP_IKS*$(C_k, G_k, \vec{v}, \epsilon, h_k)$. Note the denominator J_k always helps cancel out C^{-1} when forming the vector $\vec{v}$ during the whole simulation, and leaves only G^{-1}, which is factorized by LU decomposition once per step and reused. The estimated nonlinear truncation error is

$$e_{rr}^{(e)}(\vec{x}_{k+1}, \vec{x}_k) = h_k\phi_1^{(e)}(C_k, G_k, C_k^{-1}\Delta\vec{F}_k, \epsilon, h_k). \quad (24)$$

This requires another MEVP via Krylov subspace. When $e_{rr}^{(e)}$ is larger than given error budget, we reduce the time step $h_k = \alpha h_k$ ($\alpha < 1$). In Sec III-B, we add a correction term using $\phi_2^{(e)}$ and computed $\Delta\vec{F}_k$, then the corrected solution is

$$\vec{x}_{k+1,c} = \vec{x}_{k+1} - \gamma h_k\phi_2^{(e)}\left(C_k, G_k, C_k^{-1}\Delta\vec{F}_k, \epsilon, h_k\right). \quad (25)$$

B. Overall Circuit Simulation Framework

Algorithm 2 shows our circuit simulation framework. The proposed method takes only one LU decomposition per time step h (line 5), while BENR makes at least two LU decompositions (one for integration and NR's convergence check). When the solutions are not converged, BENR updates the time step and repeats iterations including LU operations. Moreover, based on the explicit formulation, our method does not need to repeat LU decomposition when we adjust the length h of time steps for error controls. When the error is beyond the error budget E_{rr}, our framework easily adjusts time step without extra LU operations to reduce the nonlinear error (from line 8 to 20). Note that there is no need to perform LU decomposition of G. There is no matrix factorization with C. In addition, invert Krylov subspace method deals with a much simpler matrix G, while BENR method uses the format of $\left(G + \frac{C}{h}\right)$. If C contains more detailed parasitics, BENR is affected more than our method (more non-zeros in C indeed increase the operations of sparse matrix-and-vector multiplication when we construct Krylov subspace in our framework). If the option of correction term O_{pt_c} is enabled, line 12 to 14 will perform extra computations to provide more accurate solution, leading to **ER-C**, the method of exponential Rosenbrock-Euler with correction term (i.e., Eq. (17)). Without this option, the method is plain **ER** (i.e., Eq. (14)). Line 23 shows that fast convergence rate triggers step-size increase.

V. Results

The algorithms are implemented in MATLAB and C/C++, where all devices are evaluated by BSIM3. The interactions between C/C++

Algorithm 2: **ER** and **ER-C** Methods: Circuit Simulation Using Exponential Rosenbrock-Euler Integrators

Input: Circuit netlist; E_{rr} is the error budget for circuit simulator, ϵ criteria for the convergence within MEVP_IKS; Corrected option O_{pt_c} is enabled for **ER-C**, otherwise it is **ER.**

Output: Voltage/current solution for time period $[0, T]$

1 Initialization phase: (a) load the netlist; (b) build linear matrices. Set $t = 0, k = 0, h_k = h_{init}$;
2 $x(0) = x_k =$ DC_solution;
3 **while** $t \leq T$ **do**
4 Derive $C_k, G_k, \vec{F}_k, \vec{g}_k, \vec{b}_k$ from device models at $\vec{x}_k$;
5 $LU\_decompose\,(G_k)$;
6 Compute $\phi_1^{(e)}$, $\phi_2^{(e)}$ using *MEVP_IKS* (**Algorithm 1**) with above matrices and factorized ones. Obtain $(\vec{m}_{evp}, H_m, V_m, \vec{v}, m)$ for $\phi_1^{(e)}$ and $\phi_2^{(e)}$, respectively;
7 Set $i = 0$; // the iteration index
8 **while** *true* **do**
9 Obtain $\vec{x}_{k+1}$ by Eq. (14) with the given/updated step size h_k and previous sets of $(\vec{m}_{evp}, H_m, V_m, \vec{v}, m)$ from line 6;
10 Derive $\vec{F}_{k+1}$ from device models at $\vec{x}_{k+1}$ and obtain $\Delta\vec{F}_k = \vec{F}_{k+1} - \vec{F}_k$;
11 Compute $e_{rr}^{(e)}(\vec{x}_{k+1}, \vec{x}_k) = h_k\phi_1^{(e)}(C_k, G_k, C_k^{-1}\Delta\vec{F}_{k+1}, \epsilon, h_k)$ by *MEVP_IKS*;
12 **if** O_{pt_c} *is enabled* **then**
13 $\vec{D} = \gamma h_k \phi_2^{(e)}(C_k, G_k, C_k^{-1}\Delta\vec{F}_k, \epsilon, h_k)$ by *MEVP_IKS*;
14 $\vec{x}_{k+1} = \vec{x}_{k+1} - \vec{D}$; // ER-C, *e.g.*, $\gamma = 0.1$
15 **end**
16 **if** $\|e_{rr}^{(e)}(\vec{x}_{k+1}, \vec{x}_k)\|_\infty \leq E_{rr}$ **then**
17 $\vec{x}(t + h) = \vec{x}_{k+1}$;
18 break;
19 **end**
20 $i = i + 1$, $h = \alpha h$; // update h_k, i.e., $\alpha = 1/2$
21 **end**
22 $t = t + h_k$, $k = k + 1$;
23 **if** i *is small enough* **then**
  /* e.g., update h_k when $i < 5$ */
24 $h_k = \beta h_k$; // e.g., $\beta = 2$
25 **end**
26 **end**

and MATLAB2013a are through MATLAB Executable (MEX) external interface with GCC 4.7.3. We test our algorithms in a Linux workstation (Intel CPU i7 3.4GHZ and 32GB memory). All of algorithms and procedures utilize single thread mode. All of test cases in our experiment are stiff designs with singular C matrices. Hence, we do not use previous matrix exponential method in [20], [21], since it requires extra time-consuming regularization processes at each time step just to make C to be non-singular. We compare our framework to the baseline circuit simulator using the standard backward Euler with Newton-Raphson method (BENR).

A stiff nonlinear circuit containing a inverter chain is used to demonstrate the favorable characteristics of our proposed method. We extract waveforms from one observed node of that circuit to compare the accuracy of BENR, exponential Rosenbrock-Euler method (ER) and ER with correction term (ER-C). In Fig. 2, compared to the reference solution (REF) obtained by BENR with smaller step size ($10^{-14}s$), our ER and ER-C are more accurate than BENR.

In Table I, we list the relatively large test cases with their specifications. We compare the runtime performance and capability of those large test cases among BENR and our proposed algorithms ER, ER-C. We set $\epsilon = 10^{-7}$ in **Algorithm** 1 for MEVP convergence condition. We try test cases $ckt1$ to $ckt3$, which have capacitances matrices C with extremely sparse non-zeros distributions. Our framework achieves speedups, that is 1.8× by ER and 1.4× by ER-C. The

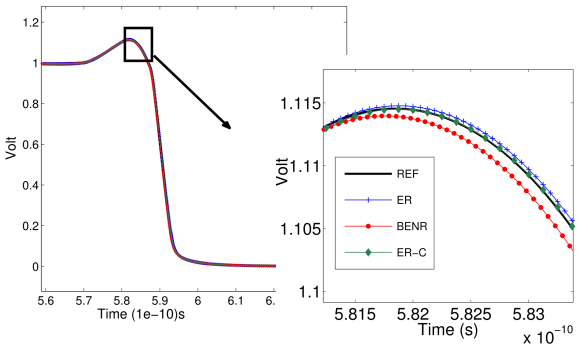

Fig. 2: Accuracy comparisons of transient simulation solutions obtained by BENR, ER and ER-C. REF is the reference solution obtained from BENR with step size $10^{-14}s$. BENR and ER both use step size $10^{-13}s$. ER-C uses 2X step size as BENR and ER and still maintain better accuracy.

small speedups are because of their relatively simple C matrices. LU decomposition of $\left(\frac{C}{h} + G\right)$ is dominated by the part of G. ER and ER-C need LU decomposition of G for invert Krylov subspace constructions as well. Therefore, our framework is not fully beneficial here. However, we notice the reduced numbers of LU operations and time steps improve the runtime performance and compensate the portion of inverted Krylov subspace generations.

Cases from $ckt4$ to $ckt8$ are more challenging since they have more complicated capacitance matrices. For the available speedup numbers, we achieve speedups by magnitude, **23.2**× by ER and **20.7**× by ER-C. For the speedup numbers not available, BENR cannot complete the simulation tasks under our limited computing resources in a reasonable time range. However, our framework processes those tasks efficiently. Note the design of $ckt5$ contains interconnect structure of FreeCPU (Fig. 1), and there are corresponding 40 drivers similar to $ckt3$. In such simulation with stronger parasitic couplings, BENR needs 54K seconds to finish the whole simulation. In contrast, the runtime performance of our methods is more stable and achieves over **35**× speedups. The increasing runtime compared to our methods on $ckt3$ is partially due to the matrix and vector multiplication, e.g., $C\vec{v}$ (line 3 in **Algorithm** 1) during the invert Krylov subspace generations since C has larger nnz. $ckt4$ has more nonlinear MOSFET devices but simpler coupling capacitances than $ckt5$. The over **4**× speedups are still observed from our methods. Cases $ckt6$ to $ckt8$ are even more challenging ones with many parasitics in certain parts of C. LU decompositions in BENR exhaust our machine's memory (32GB). Thus, we mark them as "Out of Memory" in Table I. **NA** (no available) denotes the speedups number are not available, which represents our methods' capability of handling those challenging cases while standard BENR cannot. Our methods only need to factorize G. In these three cases, the peak memory consumption of our framework is smaller than 12GB reported by Linux *top* command for corresponding process of MATLAB instance (BENR costs at least 32GB). Our methods can run through all the simulation tasks and the solutions converge to the version of ER-C with smaller step-sizes. The benchmark performance shows our algorithmic framework's advantages on memory usage and runtime performance.

VI. Conclusions

We propose a new and efficient algorithmic framework for time-domain large-scale circuit simulation using exponential integrators. We also devise a correction term on top of this formulation and further improve the accuracy. By virtue of the stable explicit nature of our formulation, we remove Newton-Raphson iterations and reduce the number of LU decomposition operations. In this framework, matrix exponential and vector product is computed by efficient invert

TABLE I: Simulation Results.

#N: the number of unknowns; **#Dev.**: the number of nonlinear devices. **nnz**$_C$ and **nnz**$_G$: the number of non-zero elements in linear capacitance/inductance matrix C and conductance/resistance matrix G. **#step**: the number of steps for transient simulation; **#NR**$_a$: the average number of Newton-Raphson iterations for each time step. **#m**$_a$: the average dimension number of invert Krylov subspace for each time step. **RT(s)**: the runtime of transient simulation. **SP**: the runtime speedup of proposed method over BENR (backward Euler formulation with Newton-Raphson iterations).

Designs & Specifications					BENR			ER				ER-C			
Case	#N	#Dev.	nnz$_C$	nnz$_G$	#step	#NR$_a$	RT(s)	#step	#m$_a$	RT(s)	SP	#step	#m$_a$	RT(s)	SP
$ckt1$	84K	40K	19K	188K	1950	2.8	1983.6	1398	30.2	1314.6	1.5×	1404	31.6	1461.6	1.4×
$ckt2$	3.3M	0.1M	1.6M	9.8M	2375	3.1	122926.2	1995	28.4	78042.2	1.6×	2005	32.7	96406.3	1.3×
$ckt3$	54K	40	19K	181K	1928	2.8	1497.4	1380	27.4	659.0	2.3×	1418	31.1	888.9	1.7×
$ckt4$	84K	40K	37K	188K	1950	2.8	6153.0	1372	27.7	1138.6	5.4×	1422	30.3	1411.4	4.4×
$ckt5$	54K	40	82K	181K	1917	2.8	53873.5	1353	27.8	1298.2	41×	1365	31.5	1447.4	37×
$ckt6$	84K	40K	157K	188K	Out of Memory			1282	27.0	1447.0	**NA**	1299	31.2	1917.7	**NA**
$ckt7$	0.2M	0.1M	0.2M	0.6M	Out of Memory			1395	26.7	8016.6	**NA**	1399	30.4	9373.8	**NA**
$ckt8$	3.3M	0.1M	1.7M	9.8M	Out of Memory			2310	29.3	86386.7	**NA**	2360	34.2	98681.7	**NA**

Krylov subspace. This approach can keep capacitance/inductance matrix from matrix factorization and avoid the time-consuming regularization process when there are singular capacitance/inductance matrices during the simulation. Moreover, this method does not need to repeat LU decompositions when we adjust the length of time steps for error controls. Compared to conventional methods, our new framework has several distinguished features. The proposed method can handle many parasitics, strongly coupled or post-layout circuits, even when conventional methods are not applicable. We test our proposed framework against standard backward Euler method with Newton-Raphson iterations, called as BENR. Our framework does not only accelerate the simulation, but also manages to finish the challenging test cases when BENR processes extremely slow or even fails. Many aspects are worthwhile to investigate within this new circuit simulation framework. For instance, matrix and vector multiplication plays an important role in our framework. Parallel processing this kind of operation by multi-core/many-core architectures could be beneficial to enhance the runtime performance.

ACKNOWLEDGMENTS

We acknowledge the support from NSF CCF-1017864. Hao Zhuang is supported by the UCSD Powell Fellowship and Qualcomm FMA Fellowship. Xinan Wang is supported by the UCSD Jacobs Fellowship. This work is also partially supported by NSFC (grant No. 61422402). We thanks the discussions with Prof. Mike Botchev, Dr. Quan Chen, Mr. Ryan Coutts, Prof. Marlis Hochbruck, Mr. Chia-Hung Liu, Dr. John Loffeld, Dr. Jingwei Lu, Prof. Alexander Ostermann, Prof. Mayya Tokman, Dr. Lining Zhang and Mr. Xiang Zhang.

REFERENCES

[1] L. Nagel, *SPICE2: A computer program to simulate semiconductor circuits*. Ph.D. dissertation, 1975.
[2] F. N. Najm, *Circuit simulation*. Wiley, 2010.
[3] L. O. Chua and P.-M. Lin, *Computer Aided Analysis of Electric Circuits: Algorithms and Computational Techniques*. Prentice-Hall, 1975.
[4] Z. Li and C.-J. Shi, "SILCA: SPICE-accurate iterative linear-centric analysis for efficient time-domain simulation of vlsi circuits with strong parasitic couplings," *IEEE TCAD*, vol. 25, no. 6, pp. 1087–1103, 2006.
[5] J. R. Phillips and L. M. Silveira, "Simulation approaches for strongly coupled interconnect systems," in *ICCAD*, pp. 430–437, 2001.
[6] Z. Zhu, H. Peng, C. K. Cheng, K. Rouz, M. Borah, and E. S. Kuh, "Two-stage Newton-Raphson method for transistor-level simulation," *IEEE TCAD*, vol. 26, no. 5, pp. 881–895, 2007.
[7] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng, and C. K. Cheng, "ePlace-MS: Electrostatics based placement for mixed-size circuits," *IEEE TCAD*, 2015.
[8] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. Huang, C.-C. Teng, and C.-K. Cheng, "ePlace: Electrostatics based placement using Nesterov's method," in *DAC*, 2014.
[9] Q. He, H. Gan, and D. Jiao, "Explicit time-domain finite-element method stabilized for an arbitrarily large time step," *IEEE TAP*, vol. 60, no. 11, pp. 5240–5250, 2012.
[10] W. Dong and P. Li, "Parallelizable stable explicit numerical integration for efficient circuit simulation," in *DAC*, 2009.
[11] T. A. Davis, *Direct Method for Sparse Linear Systems*. SIAM, 2006.
[12] X. Chen, Y. Wang, and H. Yang, "NICSLU: An adaptive sparse matrix solver for parallel circuit simulation," *IEEE TCAD*, vol. 32, no. 2, pp. 261–274, 2013.
[13] Y. Saad, "Analysis of some krylov subspace approximations to the matrix exponential operator," *SIAM J. Numer. Anal.*, vol. 29, no. 1, pp. 209–228, 1992.
[14] M. Hochbruck and A. Ostermann, "Exponential integrators," *Acta Numerica*, vol. 19, pp. 209–286, 2010.
[15] M. Hochbruck, A. Ostermann, and J. Schweitzer, "Exponential Rosenbrock-type methods," *SIAM J. Numer. Anal.*, vol. 47, no. 1, pp. 786–803, 2009.
[16] M. Caliari and A. Ostermann, "Implementation of exponential rosenbrock-type integrators," *Appl. Numer. Math.*, vol. 59, no. 3, pp. 568–581, 2009.
[17] Q. Chen, W. Schoenmaker, S.-H. Weng, C. K. Cheng, G.-H. Chen, L.-J. Jiang, and N. Wong, "A fast time-domain em-tcad coupled simulation framework via matrix exponential," in *ICCAD*, pp. 422–428, 2012.
[18] H. Zhuang, S.-H. Weng, and C. K. Cheng, "Power grid simulation using matrix exponential method with rational krylov subspaces," in *ASICON*, 2013.
[19] H. Zhuang, S.-H. Weng, J.-H. Lin, and C. K. Cheng, "MATEX: A distributed framework of transient simulation of power distribution networks," in *DAC*, 2014.
[20] S.-H. Weng, Q. Chen, and C. K. Cheng, "Time-domain analysis of large-scale circuits by matrix exponential method with adaptive control," *IEEE TCAD*, vol. 31, no. 8, pp. 1180–1193, 2012.
[21] S.-H. Weng, Q. Chen, N. Wong, and C. K. Cheng, "Circuit simulation via matrix exponential method for stiffness handling and parallel processing," in *ICCAD*, pp. 407–414, 2012.
[22] Q. Chen, S.-H. Weng, and C. K. Cheng, "A practical regularization technique for modified nodal analysis in large-scale time-domain circuit simulation," *IEEE TCAD*, vol. 31, no. 7, pp. 1031–1040, 2012.
[23] C. Zhang and W. Yu, "Efficient space management techniques for large-scale interconnect capacitance extraction with floating random walks," *IEEE TCAD*, vol. 32, no. 10, pp. 1633–1637, 2013.
[24] H. Zhuang, W. Yu, G. Hu, Z. Liu, and Z. Ye, "Fast floating random walk algorithm for multi-dielectric capacitance extraction with numerical characterization of Green's functions," in *ASP-DAC*, pp. 377–382, 2012.
[25] W. Yu, H. Zhuang, C. Zhang, G. Hu, and Z. Liu, "RWCap: A floating random walk solver for 3-D capacitance extraction of very-large-scale integration interconnects," *IEEE TCAD*, vol. 32, no. 3, pp. 353–366, 2013.
[26] R. Ionutiu, J. Rommes, and W. H. Schilders, "SparseRC: Sparsity preserving model reduction for RC circuits with many terminals," *IEEE TCAD*, vol. 30, no. 12, pp. 1828–1841, 2011.
[27] Q. Nie, Y. Zhang, and R. Zhao, "Efficient semi-implicit schemes for stiff systems," *J. Comput. Phys.*, vol. 214, no. 2, pp. 521–537, 2006.