Title	A fast time-domain EM-TCAD coupled simulation framework via matrix exponential
Author(s)	Chen, Q; Schoenmaker, W; Weng, SH; Cheng, CK; Chen, G; Jiang, L; Wong, N
Citation	The 30th IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2012), San Jose, CA., 5-8 November 2012. In ICCAD - IEEE / ACM International Conference on Computer-Aided Design Proceedings, 2012, p. 422-428
Issued Date	2012
URL	http://hdl.handle.net/10722/189785
Rights	ICCAD - IEEE / ACM International Conference on Computer-Aided Design. Proceedings. Copyright © IEEE Computer Society.

A Fast Time-Domain EM-TCAD Coupled Simulation Framework via Matrix Exponential

Quan Chen*, Wim Schoenmaker†, Shih-Hung Weng‡, Chung-Kuan Cheng‡, Guan-Hua Chen§, Li-Jun Jiang* and Ngai Wong*
*Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong
Email: {quanchen,ljiang,nwong}@eee.hku.hk
†Magwel NV, Leuven, Belgium
Email: wim.schoenmaker@magwel.com
‡Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA
Email: {s2weng,ckcheng}@ucsd.edu
§Department of Chemistry, University of Hong Kong, Hong Kong
Email: ghc@everest.hku.hk

***Abstract*—We present a fast time-domain multiphysics simulation framework that combines full-wave electromagnetism (EM) and carrier transport in semiconductor devices (TCAD). The proposed framework features a division of linear and nonlinear components in the EM-TCAD coupled system. The former is extracted and handled independently with high efficiency by a matrix exponential approach assisted with Krylov subspace method. The latter is treated by ordinary Newton's method yet with a much sparser Jacobian matrix that leads to substantial speedup in solving the linear system of equations. More convenient error management and adaptive control are also available through the linear and nonlinear decoupling.**

I. Introduction

In recent years, there is a growing demand toward combining stand-alone electromagnetic (EM) solvers and technology computer-aided design (TCAD) semiconductor device simulators in mixed-signal, RF and multi-domain simulation. This is because the simplification of semiconductors (to conductors with equivalent conductivity) in linear EM analysis and the neglect of magnetic effects in TCAD simulation have become insufficient to characterize the field-carrier interactions that are getting stronger as a direct consequence of the increasing operational frequency and the decreasing signal level.

The EM-TCAD coupled simulation essentially refers to a concurrent solution of the Maxwell's equations that describe the EM dynamics and the transport equations that describe the charge carrier dynamics in semiconductor. A widely tested co-simulation framework in the frequency domain was proposed in [1], [2]. The Maxwell's equations are formulated in terms of scalar potential V and vector potential $\mathbf{A}$ to obtain straightforward coupling with the drift and diffusion (DD) semiconductor model. The A-V co-simulator has been translated into a series of commercial tools [3], and verified against measurements with a number of industrial examples [4], [5]. Meanwhile, the A-V formulation has also been coupled with a quantum mechanical model to enable a multiscale simulation framework (called QM/EM method) for emerging nano-electronic devices [6], [7].

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2012, November 5-8, 2012, San Jose, California, USA

Whereas the frequency-domain A-V solver has been developed in a rather advanced stage, the transient counterpart so far has been less explored. Unlike the objective of obtaining small-signal response in the frequency domain, the need of transient EM-TCAD simulation comes from the desire to handle large-signal response. The first implementation of time-domain A-V solver was reported recently in [8]. The implicit backward Euler (BE) approach was employed for time discretization and the Newton's method was used to solve the nonlinear system arising from semiconductor dynamics. Due to the complications in physics and numerical treatment, five variables have to be used in the formulation and solved simultaneously. Therefore, the number of unknowns in each Newton iteration is commonly $5 \sim 6$ times the number of nodes in the computational grid. This rapid growth of problem size severely limits applicability of the co-simulation framework.

In this paper, we develop a new solution technique to dramatically improve the scalability of the time-domain EM-TCAD simulation. Our approach starts from separating the linear components in the coupled system, which commonly arise from the back-end structure, and the nonlinear components that are resulted from the front-end structure. The time integration of the linear part is realized with the lately developed matrix exponential (MEXP) method, where the matrix exponential is computed efficiently by Krylov subspace approximation. The MEXP solver has been shown to provide better scalability than BE [9] for large problems. The nonlinear part of the system is still handled by the conventional implicit time discretization and the Newton's method. The number of nonzeros in the Jacobian matrix in the Newton's method, however, is largely reduced, since the elements from linear sub-system do not appear in the Jacobian anymore. Therefore, the two main advantages of MEXP are that

- The maximum manageable problem size is substantially elevated for hybrid structures that include a non-negligible portion of linear components.
- More convenient error control and adaptive time-stepping are enabled for further performance enhancement.

Several efforts are exerted to enable the proposed MEXP approach. The singular matrix problem facing matrix-exponential-based schemes is circumvented by the introduction of differentiated Gauss law. The block structure of system matrices is leveraged to enable a fast computation of matrix vector product that is core to the Krylov subspace approximation of matrix exponential. The nonlinear

function is decoupled from the matrix exponential term by a special technique. Numerical experiments demonstrate that the MEXP solver can lead to over $10X$ speedup in Newton iteration compared with the BE solver.

II. Time-Domain Formulation of EM-TCAD Problem

The starting point of the time-domain EM-TCAD formulation is the common full-wave Maxwell's equations

$$\nabla \cdot \mathbf{D} = \rho, \quad \nabla \cdot \mathbf{B} = 0 \tag{1a}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \quad \nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \tag{1b}$$

where $\mathbf{D}, \mathbf{E}, \mathbf{B}, \mathbf{H}, \mathbf{J}$ and ρ are the displacement field, electric field, magnetic induction, magnetic field, free current density and charge density, respectively.

In the semiconductor region the Maxwell's equations are coupled with the current continuity of electrons and holes

$$\nabla \cdot \mathbf{J}_n - \frac{\partial}{\partial t} n - R\left(n, p\right) = 0 \tag{2a}$$

$$\nabla \cdot \mathbf{J}_p + \frac{\partial}{\partial t} p + R\left(n, p\right) = 0 \tag{2b}$$

where n and p are the electron and hole densities, and $R(n, p)$ denotes the net generation/recombination rate of carriers. The particle current densities in semiconductor are described by the DD model

$$\mathbf{J}_n = q\mu_n n\mathbf{E} + qkT\mu_n \nabla n \tag{3a}$$

$$\mathbf{J}_p = q\mu_p p\mathbf{E} - qkT\mu_p \nabla p \tag{3b}$$

where μ, k and T are the carrier mobility, Boltzmann constant and temperature, respectively.

To facilitate the coupling between the EM and TCAD solvers, the Maxwell's equations are rewritten with the potential variables V and $\mathbf{A}$ that satisfy $\mathbf{B} = \nabla \times \mathbf{A}$ and $\mathbf{E} = -\nabla V - \frac{\partial \mathbf{A}}{\partial t}$ [1], [2]. A new variable the pseudo-canonical momentum $\mathbf{\Pi} = \frac{\partial \mathbf{A}}{\partial t}$ is also introduced to avoid the second-order time derivative [8]. The complete system of equations is then laid out in (4) (under the "de Mari" scaling scheme [2])

$$\begin{cases} \nabla \cdot \left[\varepsilon \left(\nabla V + \mathbf{\Pi}\right)\right] + \rho = 0, \ \rho = p - n + N_D \\ \nabla \cdot \left[\varepsilon \left(\nabla \frac{\partial}{\partial t} V + \frac{\partial}{\partial t} \mathbf{\Pi}\right)\right] + \nabla \cdot \left[\sigma \left(\nabla V + \mathbf{\Pi}\right)\right] - \nabla \cdot \mathbf{J}_{sd} = 0 \end{cases} \tag{4a}$$

$$\nabla \cdot \mathbf{J}_n - \frac{\partial}{\partial t} n - R\left(n, p\right) = 0 \tag{4b}$$

$$\nabla \cdot \mathbf{J}_p + \frac{\partial}{\partial t} p + R\left(n, p\right) = 0 \tag{4c}$$

$$\frac{\partial}{\partial t} \mathbf{A} - \mathbf{\Pi} = 0 \tag{4d}$$

$$K\varepsilon \left(\frac{\partial}{\partial t} \mathbf{\Pi} + \nabla \frac{\partial}{\partial t} V\right) + K\nabla \left(-\varepsilon \frac{\partial}{\partial t} V\right) + \left[\nabla \times \left(\nabla \times \mathbf{A}\right) - \nabla \left(\nabla \cdot \mathbf{A}\right)\right] + K\sigma \left(\nabla V + \mathbf{\Pi}\right) - K\mathbf{J}_{sd} = 0 \tag{4e}$$

where $\mathbf{J}_{sd} = \mathbf{J}_n + \mathbf{J}_p$ is the total semiconductor current and K is a dimensionless constant in the scaling. The upper equation in (4a) is the common Gauss law, and the lower one is the current continuity for nodes attached with metals, where the Gauss law will be used to recover the charge densities that are not explicit unknowns. Eqn. (4e) represents a "modified" Maxwell-Ampere equation that includes the subtraction of the divergence of Lorentz gauge condition

$$\nabla \cdot \mathbf{A} + K\varepsilon \frac{\partial V}{\partial t} = 0 \tag{5}$$

to eliminate the intrinsic singularity of the curl-curl operator [10] through making $\nabla^2 = \nabla \times (\nabla \times) - \nabla(\nabla \cdot)$.

The system after spatial discretization can be assembled in a matrix format as shown in (6) on top of next page. I denotes identity matrix with conformal dimension. Eqn. (6) can be further condensed to a nonlinear differential equation

$$C\dot{x} = -Gx - F(x) - b \tag{7}$$

where the constant matrices C and G collect the linear dynamics in the system, F collects the nonlinear dynamics (generally the nonlinear semiconductor currents), and b the input term.

The work-horse numerical approach to solve (7) consists of first approximating the time derivative by certain polynomial expansion, and then solving the resulted algebraic nonlinear equations by the Newton's method. For instance, the BE approximation results in the following nonlinear equation for the solution of the $(n+1)$th step

$$\begin{aligned} C\frac{x_{n+1} - x_n}{h} &= -Gx_{n+1} - F(x_{n+1}) - b_{n+1} \\ F(x_{n+1}) + \left(\frac{C}{h} + G\right) x_{n+1} - \frac{C}{h} x_n + b_{n+1} &= 0 \end{aligned} \tag{8}$$

in which h is the time step size. Then the Newton's method solves (8) by building and solving the Jacobian matrix in each iteration

$$\begin{aligned} &\left(\left.\frac{\partial F}{\partial x}\right|_{x_{n+1}^{k+1}} + \frac{C}{h} + G\right) \Delta x_{n+1}^{k+1} = \\ &\qquad - \left(F(x_{n+1}^k) + \left(\frac{C}{h} + G\right) x_{n+1}^k - \frac{C}{h} x_n + b_{n+1}\right) \end{aligned} \tag{9}$$

where the Jacobian $J = \left(\left.\frac{\partial F}{\partial x}\right|_{x_{n+1}^{k+1}} + \frac{C}{h} + G\right)$. Δx_{n+1}^{k+1} denotes the update of x at the $(k+1)$th Newton iteration in the $(n+1)$th step.

III. Time Evolution with Matrix Exponential

A. Basic Matrix Exponential Method

The matrix exponential method relies on converting (7) to an ordinary differential equation (ODE) by multiplying the inverse of C on both sides

$$\dot{x} = -C^{-1}Gx - C^{-1}F(x) - C^{-1}b \tag{11}$$

Note that we assume the inverse of C is easy to obtain at this moment. The treatment for a singular C and the fast computation of C^{-1} will be presented in later subsections.

The analytical solution of (11) for x_{n+1} is given with the matrix exponential [11]

$$\begin{aligned} x_{n+1} = &\, e^{Mh} x_n + \\ &\int_0^h e^{M(h-\tau)} \left[-C^{-1}F(t_n + \tau) - C^{-1}b(t_n + \tau)\right] d\tau \end{aligned} \tag{12}$$

in which $M = -C^{-1}G$.

Successful application of the above matrix exponential formula requires answers to the following three questions:

- Is C really invertible? How to deal with the systems with a singular C?
- How to compute the matrix exponential e^{Mh} (which involves C^{-1}) efficiently for large-scale problems?
- How to handle the nonlinear function F in the integral?

The following three subsections will be devoted to addressing these three issues.

B. Regularization of C

The matrix C is generally singular. The singularity arises from the Gauss law which includes no time derivative term. As evident in (6), the nodes (collected in V_1) where Gauss law is enforced generates empty rows in C and renders the matrix singular. For the nodes on which the current continuity is enforced (generally nodes attached

$$\begin{bmatrix} \begin{bmatrix}0\\ \nabla\cdot(\varepsilon\nabla)\end{bmatrix} & & \begin{bmatrix}0\\ \nabla\cdot(\varepsilon)\end{bmatrix} \\ & -I & \\ & I & \\ & I & \\ K(\varepsilon\nabla-\nabla\cdot(\varepsilon)) & & K\varepsilon \end{bmatrix}\begin{bmatrix}\dot{V}_1\\ \dot{V}_2\\ \dot{n}\\ \dot{p}\\ \dot{A}\\ \dot{\Pi}\end{bmatrix} = -\begin{bmatrix} \begin{bmatrix}\nabla\cdot(\varepsilon\nabla)\\ \nabla\cdot(\sigma\nabla)\end{bmatrix}\begin{bmatrix}-I_n\\ 0\end{bmatrix}\begin{bmatrix}I_p\\ 0\end{bmatrix} & \begin{bmatrix}\nabla\cdot(\varepsilon)\\ \nabla\cdot(\sigma)\end{bmatrix} \\ & \\ K\sigma\nabla & \nabla^2 \quad -I \quad K\sigma \end{bmatrix}\begin{bmatrix}V_1\\ V_2\\ n\\ p\\ A\\ \Pi\end{bmatrix} - \begin{bmatrix}\begin{bmatrix}0\\ -\nabla\cdot\bar{J}_{semi}\end{bmatrix}\\ \nabla\cdot\bar{J}_n - R(n,p)\\ \nabla\cdot\bar{J}_p + R(n,p)\\ 0\\ -K\bar{J}_{semi}\end{bmatrix} - \begin{bmatrix}\begin{bmatrix}V_S+N_D\\ 0\end{bmatrix}\\ 0\\ 0\\ 0\\ 0\end{bmatrix} \tag{6}$$

$$\begin{bmatrix} \begin{bmatrix}\nabla\cdot(\varepsilon\nabla)\\ \nabla\cdot(\varepsilon\nabla)\end{bmatrix} & \begin{bmatrix}-I_n\\ 0\end{bmatrix}\begin{bmatrix}I_p\\ 0\end{bmatrix} & \begin{bmatrix}\nabla\cdot(\varepsilon)\\ \nabla\cdot(\varepsilon)\end{bmatrix} \\ & -I & \\ & I & \\ & I & \\ K\varepsilon\nabla - K\nabla\cdot(\varepsilon) & & K\varepsilon \end{bmatrix}\begin{bmatrix}\dot{V}_1\\ \dot{V}_2\\ \dot{n}\\ \dot{p}\\ \dot{A}\\ \dot{\Pi}\end{bmatrix} = -\begin{bmatrix} 0 & & \begin{bmatrix}0\\ \nabla\cdot(\sigma)\end{bmatrix} \\ \begin{bmatrix}\nabla\cdot(\sigma\nabla)\end{bmatrix} & 0 & \\ & 0 & \\ K\sigma\nabla & 0 \quad \nabla^2 & -I \quad K\sigma \end{bmatrix}\begin{bmatrix}V_1\\ V_2\\ n\\ p\\ A\\ \Pi\end{bmatrix} - \begin{bmatrix}\begin{bmatrix}0\\ -\nabla\cdot\bar{J}_{semi}\end{bmatrix}\\ \nabla\cdot\bar{J}_n - R(n,p)\\ \nabla\cdot\bar{J}_p + R(n,p)\\ 0\\ -K\bar{J}_{semi}\end{bmatrix} - \begin{bmatrix}\begin{bmatrix}\dot{V}_S\\ 0\end{bmatrix}\\ 0\\ 0\\ 0\\ 0\end{bmatrix} \tag{10}$$

with metals), the presence of displacement current (though small compared with the conduction current) prevents the vacancy in C.

To overcome the singular C problem, we differentiate the Gauss law, which gives

$$\nabla\cdot\left[\varepsilon\left(\nabla\frac{\partial V}{\partial t}+\frac{\partial \mathbf{\Pi}}{\partial t}\right)\right]+\frac{\partial \rho}{\partial t}=0 \tag{13}$$

In the matrix equation (6), this is equivalent to moving the first row of G to C and differentiating the b term accordingly, yielding a new equation system with nonsingular C (10). Note that G and M are allowed to be singular because the exponential function is analytical.

Several implications are induced from the "regularization" process above. From a physical perspective, the differentiated Gauss law represents essentially another "form" of current continuity in semiconductor and insulator regions, which equates the displacement current and the temporal change of charge. The explicitly solved current continuity equations (4b) and (4c), on the other hand, equates the particle current and the temporal change of charge. These two continuity equations together recover the "true" current continuity in semiconductor (and insulator), viz. the displacement current = the particle current. Note that the generation/recombination terms for n- and p-type carriers cancel each other in the differentiated Gauss law. From a numerical perspective, the regularization converts the original equation $f(t) = g(t)$ to an equation in the form of $\dot{f}(t) = \dot{g}(t)$. Therefore, it is important to guarantee the temporal evolution starts from a correct initial condition, i.e., ensure that $f(0) = g(0)$. The correct initial condition can be obtained by a prior static simulation.

In addition, the derivative of source excitation V_S is now required, which is trivial if the sources are defined by continuous differentiable functions. If the sources are defined in piece-wise-linear form (which is common in circuit simulation), the selection of time step must respect the turning points of the waveforms, so as to ensure that within each step all the derivatives of sources are constant.

C. Fast Krylov Subspace Approximation of Matrix Exponential

The matrix-exponential-vector-product (MEVP) $e^{Mh}x$, which is the main computation for the time marching scheme (12), can be done efficiently by projecting the pair of (M, x) onto an m-dimensional Krylov subspace defined by $\mathrm{K_m}(M, x) = span\{x, Mx, \ldots, M^{m-1}x\}$, and computing the MEVP on the Krylov subspace.

Specifically, an N-by-m orthonormal basis V_m ($N = dim(M)$) and an $(m + 1)$-by-m upper Hessenberg matrix H for the Krylov subspace $\mathrm{K_m}$ are first constructed by the Arnoldi process. Then $e^{Mh}x$ can be approximated by [12]

$$e^{Mh}x \approx \|x\|_2 V_m e^{H_m h} e_1 \tag{14}$$

where $H_m = H(1 : m, 1 : m)$. Since m is small (< 100), the exponential of H_m can be easily computed. A posteriori error bound [12] is available to estimate the approximation error of (14)

$$err = \|x\|_2 H(m+1, m)|e_m^{\mathsf{T}} e^{H_m h} e_1| \tag{15}$$

The main cost of the Krylov subspace approximation lies in the m matrix vector products Mx involved in the Arnoldi process. Recall that $M = C^{-1}G$, then each $Mx = -C^{-1}(Gx)$ involves one sparse matrix vector product and one sparse linear solve. Since, typically, C is rather sparser than $C + G$ that needs to be inverted in BE-type methods when dealing with linear structures, the whole matrix exponential method can be more efficient than the BE solver as reported in [9]. Further benefit can be gained from easy parallelization of the matrix vector multiplications.

The direct computation of Mx, though generally well-performed, involves still an N-by-N matrix. An even more efficient evaluation of Mx is available by exploiting the block structures of C and G. As shown in (16), the inverse of C can be constructed explicitly with the block matrix inverse lemma, and the computation of Mx boils down to a series of block matrix operations requiring only the inverse of a much smaller matrix S.

$$\begin{aligned}
Mx =& -C^{-1}Gx \\
=& -\begin{bmatrix} C_{VV} & C_{Vn} & C_{Vp} & 0 & C_{V\Pi} \\ & -I & & & \\ & & I & & \\ & & & I & \\ C_{\Pi V} & 0 & 0 & 0 & C_{\Pi\Pi} \end{bmatrix}^{-1}\begin{bmatrix} G_{VV} & 0 & 0 & 0 & G_{V\Pi} \\ & & 0 & & \\ & & & 0 & \\ & & & & 0 \quad -I \\ G_{\Pi V} & 0 & 0 & G_{\Pi A} & G_{\Pi\Pi} \end{bmatrix}\begin{bmatrix}x_V\\ x_n\\ x_p\\ x_A\\ x_\Pi\end{bmatrix} \\
=& -\begin{bmatrix} X & \tilde{C}_{Vn} & \tilde{C}_{Vp} & 0 & Y \\ & -I & & & \\ & & I & & \\ & & & I & \\ Z & \tilde{C}_{\Pi n} & \tilde{C}_{\Pi p} & 0 & U \end{bmatrix}\begin{bmatrix} G_{VV} & 0 & 0 & 0 & G_{V\Pi} \\ & & 0 & & \\ & & & 0 & \\ & & & & 0 \quad -I \\ G_{\Pi V} & 0 & 0 & G_{\Pi A} & G_{\Pi\Pi} \end{bmatrix}\begin{bmatrix}x_V\\ x_n\\ x_p\\ x_A\\ x_\Pi\end{bmatrix} \\
=& -\begin{bmatrix} XG_{VV}+YG_{\Pi V} & 0 & 0 & YG_{\Pi A} & XG_{V\Pi}+YG_{\Pi\Pi} \\ & & 0 & & \\ & & & 0 & \\ & & & & 0 \quad -I \\ ZG_{VV}+UG_{\Pi V} & 0 & 0 & UG_{\Pi A} & ZG_{V\Pi}+UG_{\Pi\Pi} \end{bmatrix}\begin{bmatrix}x_V\\ x_n\\ x_p\\ x_A\\ x_\Pi\end{bmatrix} \\
=& -\begin{bmatrix} (XG_{VV}+YG_{\Pi V})\,x_V + YG_{\Pi A}x_A + (XG_{V\Pi}+YG_{\Pi\Pi})\,x_\Pi \\ 0 \\ 0 \\ -x_\Pi \\ (ZG_{VV}+UG_{\Pi V})\,x_V + UG_{\Pi A}x_A + (ZG_{V\Pi}+UG_{\Pi\Pi})\,x_\Pi \end{bmatrix}
\end{aligned} \tag{16}$$

where

$$\begin{aligned}
&X = S^{-1}, \quad Y = -S^{-1}C_{V\Pi}C_{\Pi\Pi}^{-1}, \quad Z = -C_{\Pi\Pi}^{-1}C_{\Pi V}S^{-1}, \\
&U = C_{\Pi\Pi}^{-1}\left(C_{\Pi V}S^{-1}C_{V\Pi}C_{\Pi\Pi}^{-1} + I\right), \\
&S = C_{VV} - C_{V\Pi}C_{\Pi\Pi}^{-1}C_{\Pi V}
\end{aligned} \tag{17}$$

Note that the inverse of the diagonal matrix $C_{\Pi\Pi}$ is trivial to obtain. With further reuse of intermediate results (the derivation is given in Appendix), the cost of Mx can be ultimately reduced to only *one* sparse linear solution with S and several sparse (sub)-matrix vector products. This explicit construction process substantially reduces the size of linear system to be solved, since for general hybrid structures

$$dim(S) = \#\text{ of nodes} < \frac{1}{5}dim(M) \tag{18}$$

D. Solution of Nonlinear System

The nonlinear function in (12) is approximated with a second-order implicit formula $F(\tau) = (F(x_n) + F(x_{n+1}))/2$ (which renders the scheme A-stable even for the nonlinear term [13]). Assuming further a constant derivative of source in each step (i.e., b is a constant), the integral in (12) can be integrated explicitly, yielding

$$x_{n+1} = \frac{h}{2}\frac{e^{Mh}-I}{Mh}F(x_{n+1}) + e^{Mh}x_n + h\frac{e^{Mh}-I}{Mh}\left(\frac{F_n}{2} + b_n\right) \tag{19}$$

The function of matrix exponential before $F(x_{n+1})$ is highly undesirable. In each Newton iteration the Jacobian matrix will be pre-multiplied with a matrix exponential, the computation of which is generally prohibitive. Recall that we prefer computing only the product of a matrix exponential and a vector. A close analysis reveals that the first term in (19) represents a coupling between the linear dynamics in e^{Mh} and the nonlinear dynamics in $F(x_{n+1})$. Hence, we adopt a technique developed in [14], which decouples the linear and the nonlinear terms by approximating the entire integrand in (12) with Lagrange polynomial. The second-order approximation yields

$$\begin{aligned} x_{n+1} = &-\frac{h}{2}C^{-1}F(x_{n+1}) + e^{Mh}\left(x_n - \frac{h}{2}C^{-1}F_n\right) \\ &+ \frac{e^{Mh}-I}{M}\left(-C^{-1}b_{n+1}\right) \end{aligned} \tag{20}$$

Now $F(x_{n+1})$ has a constant coefficient and e^{Mh} is only multiplied with terms involving known quantities from the previous step. Further saving can be achieved by merging the last two terms in (20) into one single matrix exponential with a slightly larger matrix [15]

$$\begin{aligned} x_{n+1} &= -\frac{h}{2}C^{-1}F(x_{n+1}) \\ &\quad + [I_N, 0]\exp\left\{\begin{bmatrix} M & -C^{-1}b_{n+1} \\ 0 & 0 \end{bmatrix} h\right\}\begin{bmatrix} x_n - \frac{h}{2}C^{-1}F_n \\ 1 \end{bmatrix} \\ &= -\frac{h}{2}C^{-1}F(x_{n+1}) + u_{n+1} \end{aligned} \tag{21}$$

The vector u_{n+1}, denoted as the *linear update* term, is calculated only once at the beginning of each time step by the Krylov subspace method described in Subsection III-C.

With (21), the Newton's method solves the following linear system in each iteration

$$\left(C + \frac{h}{2}\frac{\partial F}{\partial x}\bigg|_{x_{n+1}^{k+1}}\right)\Delta x_{n+1}^{k+1} = -\left(\frac{h}{2}F_{n+1}^k + Cx_{n+1}^k - Cu_{n+1}\right) \tag{22}$$

The local truncation error (LTE) of the approximation (20) is

$$-\frac{1}{12}\left((Mh)^2F_n + (Mh)\dot{F}_n + \ddot{F}_n\right) \tag{23}$$

One major advantage that MEXP can bring is a sparser Jacobian matrix in the Newton iterations. Comparing (9) and (22), the Jacobian in BE consists of $\left(\frac{\partial F}{\partial x} + \frac{C}{h} + G\right)$, whereas in MEXP it is only $\left(C + \frac{h}{2}\frac{\partial F}{\partial x}\right)$. In large-scale computation, improvement in sparsity in a linear system can lead to significant computational savings.

The rationale behind this improved sparsity is that, the linear components of a system generally do not change during the treatments for nonlinear components. As reflected in the composition of Jacobian matrix, the C and G matrices in the BE context are both constant in the whole duration of Newton's method; only the Jacobian of F varies among iterations. *In conventional approaches, the stable linear components are mixed into the Jacobian of each Newton iteration, which do not provide new information while still consuming a substantial part of computation. In contrast, the MEXP approach separates the linear and the nonlinear parts of the system, and represents the linear sub-system in a matrix exponential form. The linear sub-system is then handled by the efficient Krylov subspace technique in one run, leaving just the nonlinear sub-system participating in the subsequent Newton iterations.*

E. Error Control and Adaptivity

Whereas constant step size is simple to implement, adaptive time-stepping commonly provides better accuracy and performance in simulation. The adaption requires certain error control to determine when to change a step size and re-evaluation scheme when the step size does change. The separation of linear and nonlinear components in MEXP, from this perspective, also enables convenient and efficient error control and adaptive scheme compared with that in traditional methods. The left flowchart in Fig. 1 shows the flow of BE-like methods, wherein the linear accuracy (i.e., the accuracy of the polynomial expansion of the time derivative) is tested only after the Newton iterations converge. Once the temporary solution cannot pass the test, the time step must reverse and a new round of Newton iterations will be restarted, incurring possibly a significant waste of computation.

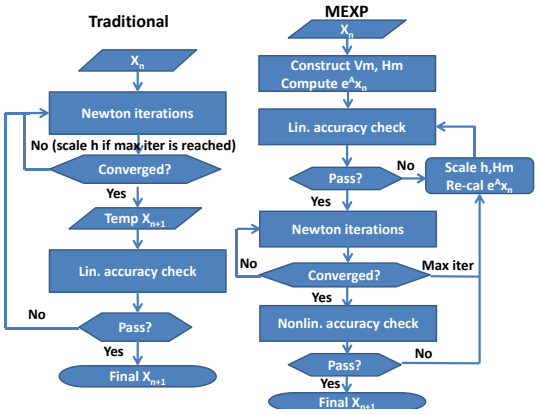


Fig. 1. Flows of traditional methods and MEXP.

On the contrary, the linear accuracy check (15) in MEXP, as shown in the right flowchart in Fig. 1, is prior to the nonlinear solution. The accuracy in the matrix exponential approximation can be checked and tuned to be satisfactory before the algorithm enters the nonlinear phase. In addition, the Krylov subspace approximation (14) has a nice scaling invariant property that $\alpha M \rightarrow \alpha H_m$, which suggests that the calculated V_m and H can be reused (with corresponding scaling) to generate a new solution without demanding a new Arnoldi process when h is changed, i.e.,

$$e^{Mh_1}x \approx \|x\|_2 V_m e^{\alpha H_m}e_1, \quad \alpha = h_1/h \tag{24}$$

Therefore, the step size can be adjusted multiple times at a negligible cost since each re-evaluation involves merely an exponential of H_m.

Although the nonlinear accuracy check (23) may also result in restart of Newton iterations in the MEXP flow, the chance is far lower

TABLE I
SPECIFICATIONS OF TEST STRUCTURES

Name	Description	# of nodes (unknowns)	Breakdown of lin./nonlin. nodes
CNT	Single carbon nanotube embedded in silicon substrate	3,825 (27,294)	63/3,753
SUB	Substrate noise isolation structure [10]	6,384 (41,368)	1,064/5,320
VCO	8-shaped inductor in voltage controlled oscillator [8]	21,312 (149,898)	4,736/16,576

than in the BE flow. This is because in general the majority of error in time-domain discretization comes from the linear components, e.g., the small parasitics in back-end structures that require small time step to capture. Hence, the separation of linear and nonlinear components can help avoid effectively the waste of Newton iterations when adaptive scheme is applied.

IV. NUMERICAL RESULTS

The time-domain EM-TCAD simulation with MEXP solver is implemented in Matlab. For comparison a BE solver is also implemented. Three test structures with the specifications detailed in Table IV were simulated to compare their performance. The linear/nonlinear nodes [1] refer to the nodes attached without/with semiconductor volumes, respectively. All tests were conducted on a 3.2GHz 16Gb-RAM Linux-based server. The iterative solver involved in the test is GMRES ($tol = 10^{-6}$), equipped with incomplete LU preconditioner ILU(10^{-3}) and column approximate minimum degree (COLAMD) permutation [16]. The combination represents the state-of-the-art in iterative solution of large linear system.

Fig. 2 shows the transient current through the left contact of the CNT example when a sinusoidal voltage of 100GHz is applied. A small step size of $0.25fs$ is used to guarantee that the BE method is accurate enough to serve as a benchmark. The dimension $m = 65$ of Krylov subspace approximation in MEXP is selected to guarantee the (estimated) error throughout the simulation smaller than 10^{-8}. A nice agreement is observed between the BE and the MEXP curves, which confirms the validity of the regularization of C as well as the accuracy of the MEXP method.

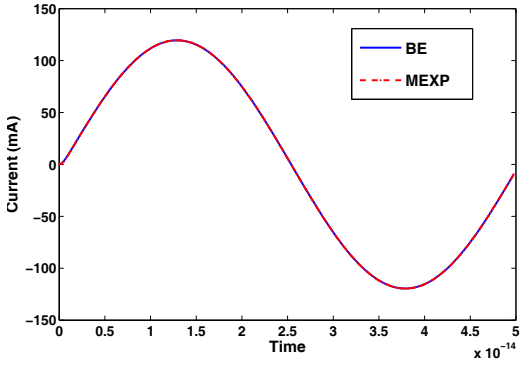
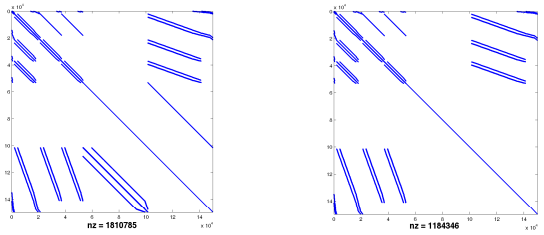


Fig. 2. Current through the CNT structure obtained by BE and MEXP.

The runtime breakdown of the two methods for one step are shown in Table II. The time taken by each matrix vector product Mx via (16) (the numbers in parenthesis are the time taken by a direct

[1]Nonlinear nodes can still generate nonzeros in the G matrix



(a) (b)

Fig. 3. Sparsity pattern of the Jacobian matrices in BE and MEXP solvers (VCO example).

computation), and the total time for the Krylov subspace calculation of matrix exponential are reported. In the Newton iteration sector, the runtime per iteration is measured for using a direct solver (backslash in Matlab) and the iterative solver (COLAMD+ILU+GMRES combo). The numbers of nonzeros in the Jacobian in Newton iteration are also shown.

It is seen that the cost of matrix exponential computation, which is sole for MEXP, is generally moderate because of the efficiency of Krylov subspace method. The fast computation of Mx in (16) enables further saving (over $2X$ for the larger VCO example) compared with the direct computation. For the runtime of Newton iteration, remarkable speedup is observed for the MEXP solver, ranging from $5X$ to $10X$ for the direct solver and $5X$ to $20X$ for the iterative solver. Note that, given the memory bottleneck and the lack of parallelizability for direct solvers, in general the iterative solvers remain the only feasible choice for large-scale problems. Therefore, the speedup from MEXP for iterative solvers is more indicative. The improved sparsity in Jacobian matrices is also evident from the reduction in the number of nonzeros. In general, the more Newton iterations in one time step, the higher gain from MEXP.

Fig. 3 shows the difference in sparsity of the Jacobian in BE and MEXP. Because of the absence of the G component, the Jacobian in MEXP is free of the ∇^2 term in the last row of (10) and the identity term for Π in the equation of A. In particular, the matrix block for the former tends to be the most dense one in the Jacobian of BE, the elimination of which in MEXP thus represents rather significant saving when the sparse system is solved.

Table III compares the performance of BE and MEXP using a simple adaptive time-stepping scheme. The CNT structure is simulated with a time span of $50fs$ and an initial h of $0.25fs$. In BE, the step size is decreased by $0.8X$ when the linear LTE (which is estimated by $h^2\ddot{x}/2$) is larger than 10^{-4}, and increased by $1.25X$ for the next step when LTE $< 10^{-6}$ and the Newton's method converges fast within ≤ 2 iterations. In MEXP, the linear accuracy is measured by (15) and tuned by the fast re-evaluation (24). The step size is enlarged with the same criteria as BE. The m is fixed to be 60.

In the adaptive BE, more then $1/5$ of the Newton iterations are wasted due to the withdrawing of temporary solutions that cannot pass the linear accuracy check. For more stringent linear accuracy requirement, more Newton iterations may be discarded. On the other hand, no Newton iteration is wasted in the adaptive MEXP, since the linear accuracy has been guaranteed before the nonlinear solution and the time-domain error from the linear components is more dominant. This adds additional runtime saving on top of the speedup in individual Newton iteration reported in Table II.

Some remarks are in order:

- Solving large linear system in Newton-like methods has long been a bottleneck in time-domain scientific computing involving nonlinearity. Workarounds have been investigated actively from

TABLE II
RUNTIME BREAKDOWN OF BE AND MEXP (TIME UNIT: SECONDS)

Cases	Method	Krylov approx. of matrix exp.			Newton's method (per iteration)		
		m	time per Mx	total time	direct	GMRES	nnz(Jacob)
CNT	BE	-	-	-	5.7	10.3	0.4M
	MEXP	65	0.086(0.12)	5.76	0.9	1.9	0.2M
SUB	BE	-	-	-	4.6	12.9	0.5M
	MEXP	40	0.11(0.19)	4.58	0.9	2.3	0.3M
VCO	BE	-	-	-	57.5	380.3	1.8M
	MEXP	60	0.71(1.51)	47.7	5.6	27.1	1.2M

TABLE III
PERFORMANCE OF BE AND MEXP WITH ADAPTIVE TIME STEPPING (TIME UNIT: SECONDS)

Method	# of steps	# of NT (total)	# of NT (wasted)	T_{total}
BE	101	299	59	1993.0
MEXP	92	201	0	258.4

various perspectives, including divide-and-conquer (e.g., domain decomposition) to reduce the problem size of individual solution, matrix-free nonlinear solvers (fixed point method and matrix-free Newton's method, etc.) to avoid solution of linear systems, and fast linear solvers (exploiting advanced preconditioners and parallelization).

- The MEXP framework proposed in this work pursues yet another direction. Since it is still desirable to keep the Newton's method in the game for better convergence and stability, we aim to minimize the portion of system that will participate in the nonlinearity treatment and handle the remaining (linear) portion with techniques that have better scalability than direct solving of linear systems, e.g., matrix exponential formula + Krylov subspace method. This way, the overall simulation capacity can be substantially expanded until the solution to the nonlinear fraction itself hits again the ceiling of available computation power. The division of linear and nonlinear components also facilitates error control and adaptive time-stepping.
- The real benefit of dividing linear and nonlinear components depends on their proportion in the system. The more linear components, the higher gain one can expect, and vice versa. In the extreme case that the structure contains no linear component, the MEXP scheme will perform similarly with the BE method. Hence, the MEXP method is specially devised for the EM-TCAD context where front-end and back-end structures are commonly coexistent in the same simulation domain.

V. CONCLUSION

We have presented a fast solution technique for time-domain EM-TCAD co-simulation. The essence lies in the division of linear components and nonlinear components, and handling the two with different strategies. The sparsity of the Jacobian matrix in the nonlinear treatment is largely improved, since the linear components will no longer participate in the treatment. Several techniques have been developed to guarantee the scalability of the solution of linear part. It has been demonstrated that the new MEXP method can improve substantially the performance of the coupled simulator and provides convenient control of error and adaptive stepping.

VI. ACKNOWLEDGMENTS

The authors would like to acknowledge the support from Hong Kong University Grant Council (AoE/P-04/08), Hong Kong General Research Fund (GRF) project 718711E and NSF CCF-1017864.

REFERENCES

[1] P. Meuris, W. Schoenmaker, and W. Magnus, "Strategy for electromagnetic interconnect modeling," *IEEE Trans. Comput.-Aided Design*, vol. 20, no. 6, pp. 753–762, Jun. 2001.
[2] W. Schoenmaker and P. Meuris, "Electromagnetic interconnects and passives modeling: software implementation issues," *IEEE Trans. Comput.-Aided Design*, vol. 21, no. 5, pp. 534–543, May 2002.
[3] Magwel. [Online]. Available: http://www.magwel.com/
[4] CODESTAR - compact modeling of on-chip passive structures at high frequencies. [Online]. Available: http://www.magwel.com/codestar/
[5] ICESTARS - integrated circuit/EM simulation and design technologies for advanced radio systems-on-chip. [Online]. Available: http://www.icestars.eu
[6] C. Yam, L. Meng, G. Chen, Q. Chen, and N. Wong, "Multiscale quantum mechanics/electromagnetics simulation for electronic devices," *Phys. Chem. Chem. Phys. (PCCP)*, vol. 13, pp. 14 365–14 369, 2011.
[7] L. Meng, C. Yam, S. Koo, Q. Chen, N. Wong, and G. Chen, "Dynamic multiscale quantum mechanics / electromagnetics simulation method," *J. Chem. Theory and Comput. (JCTC)*, vol. 8, no. 4, pp. 1190–1199, Feb 2012.
[8] W. Schoenmaker, M. Matthes, B. D. Smedt, S. Baumanns, C. Tischendorf, and R. Janssen, "Large signal simulation of integrated inductors on semiconducting substrates," in *Proc. Design, Automation and Test in Europe (DATE)*, Mar 2012, pp. 1221–1226.
[9] S.-H. Weng, Q. Chen, and C.-K. Cheng, "Circuit simulation using matrix exponential method," in *ASIC (ASICON), 2011 IEEE 9th International Conference on*, Oct 2011, pp. 369 –372.
[10] Q. Chen, W. Schoenmaker, P. Meuris, and N. Wong, "An effective formulation of coupled electromagnetic-TCAD simulation for extremely high frequency onwards," *IEEE Trans. Comput.-Aided Design*, vol. 30, no. 6, pp. 866–876, Jun 2011.
[11] L. O. Chua and P. M. Lin, *Computer-Aided Analysis of Electronic Circuits*. New Jersey: Prentice-Hall, 1975, ch. 8.
[12] Y. Saad, "Analysis of some Krylov subspace approximations to the matrix exponential operator," *SIAM Journal on Numerical Analysis*, 1992.
[13] G. Beylkin, J. M. Keiser, and L. Vozovoi, "A new class of time discretization schemes for the solution of nonlinear PDEs," *J. Comput. Phys.*, vol. 147, pp. 362–387, Dec 1998.
[14] Q. Nie, Y.-T. Zhang, and R. Zhao, "Efficient semi-implicit schemes for stiff systems," *Journal of Computational Physics*, vol. 214, p. 521C537, 2006.
[15] A. H. Al-Mohy and N. J. Higham, "Computing the action of the matrix exponential, with an application to exponential integrators," *SIAM Journal on Scientific Computing*, vol. 33, no. 2, pp. 488–511, 2011.
[16] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng, "A column approximate minimum degree ordering algorithm," *ACM Trans. Math. Softw.*, vol. 30, no. 3, pp. 353–376, 2004.

APPENDIX

Following (16),

$$Mx = -\begin{bmatrix} (XG_{VV} + YG_{\Pi V})\, x_V + YG_{\Pi A} x_A + (XG_{V\Pi} + YG_{\Pi\Pi})\, x_\Pi \\ 0 \\ 0 \\ -x_\Pi \\ (ZG_{VV} + UG_{\Pi V})\, x_V + UG_{\Pi A} x_A + (ZG_{V\Pi} + UG_{\Pi\Pi})\, x_\Pi \end{bmatrix} \tag{25}$$

with

$$X = S^{-1}, Y = -S^{-1} C_{V\Pi} C_{\Pi\Pi}^{-1}, Z = -C_{\Pi\Pi}^{-1} C_{\Pi V} S^{-1}, U = C_{\Pi\Pi}^{-1} \left(C_{\Pi V} S^{-1} C_{V\Pi} C_{\Pi\Pi}^{-1} + I \right), S = C_{VV} - C_{V\Pi} C_{\Pi\Pi}^{-1} C_{\Pi V} \tag{26}$$

The top and bottom elements in (25) are respectively

$$\begin{aligned} &(XG_{VV} + YG_{\Pi V})\, x_V + YG_{\Pi A} x_A + (XG_{V\Pi} + YG_{\Pi\Pi})\, x_\Pi \\ &= M^{-1} G_{VV} x_V - M^{-1} C_{V\Pi} C_{\Pi\Pi}^{-1} G_{\Pi V} x_V - M^{-1} C_{V\Pi} C_{\Pi\Pi}^{-1} G_{\Pi A} x_A + M^{-1} G_{V\Pi} x_\Pi - M^{-1} C_{V\Pi} C_{\Pi\Pi}^{-1} G_{\Pi\Pi} x_\Pi \\ &= M^{-1} \left[(G_{VV} x_V + G_{V\Pi} x_\Pi) - C_{V\Pi} C_{\Pi\Pi}^{-1} \left(G_{\Pi V} x_V + G_{\Pi A} x_A + G_{\Pi\Pi} x_\Pi \right) \right] \end{aligned} \tag{27}$$

and

$$\begin{aligned} &(ZG_{VV} + UG_{\Pi V})\, x_V + UG_{\Pi A} x_A + (ZG_{V\Pi} + UG_{\Pi\Pi})\, x_\Pi \\ &= -C_{\Pi\Pi}^{-1} C_{\Pi V} M^{-1} G_{VV} x_V + \left(C_{\Pi\Pi}^{-1} C_{\Pi V} M^{-1} C_{V\Pi} C_{\Pi\Pi}^{-1} + C_{\Pi\Pi}^{-1} \right) G_{\Pi V} x_V + \left(C_{\Pi\Pi}^{-1} C_{\Pi V} M^{-1} C_{V\Pi} C_{\Pi\Pi}^{-1} + C_{\Pi\Pi}^{-1} \right) G_{\Pi A} x_A \\ &\quad - C_{\Pi\Pi}^{-1} C_{\Pi V} M^{-1} G_{V\Pi} x_\Pi + \left(C_{\Pi\Pi}^{-1} C_{\Pi V} M^{-1} C_{V\Pi} C_{\Pi\Pi}^{-1} + C_{\Pi\Pi}^{-1} \right) G_{\Pi\Pi} x_\Pi \\ &= -C_{\Pi\Pi}^{-1} C_{\Pi V} M^{-1} \left[(G_{VV} x_V + G_{V\Pi} x_\Pi) - C_{V\Pi} C_{\Pi\Pi}^{-1} \left(G_{\Pi V} x_V + G_{\Pi A} x_A + G_{\Pi\Pi} x_\Pi \right) \right] + C_{\Pi\Pi}^{-1} \left(G_{\Pi V} x_V + G_{\Pi A} x_A + G_{\Pi\Pi} x_\Pi \right) \end{aligned} \tag{28}$$

Defining

$$v_1 = G_{VV} x_V + G_{V\Pi} x_\Pi,\ v_2 = C_{\Pi\Pi}^{-1} \left(G_{\Pi V} x_V + G_{\Pi A} x_A + G_{\Pi\Pi} x_\Pi \right),\ v_3 = S^{-1} \left(v_1 - C_{V\Pi} v_2 \right) \tag{29}$$

one can obtain

$$Mx = -\begin{bmatrix} v_3 \\ 0 \\ 0 \\ -x_\Pi \\ -C_{\Pi\Pi}^{-1} C_{\Pi V} v_3 + v_2 \end{bmatrix} \tag{30}$$

which involves only one sparse linear solution and several sparse matrix vector multiplications.