A Multilevel Newton Algorithm with Macromodeling and Latency for the Analysis of Large-Scale Nonlinear Circuits in the Time Domain

N. B. GUY RABBAT, SENIOR MEMBER, IEEE, ALBERTO L. SANGIOVANNI-VINCENTELLI, MEMBER, IEEE AND HSUEH Y. HSIEH, MEMBER, IEEE

***Abstract*—Analysis techniques which take advantage of the structural properties of large-scale electrical networks are discussed. Exact macromodels of a subnetwork are defined and a sufficient condition on the subnetwork equations for the existence of a macromodel is given. A multilevel Newton algorithm based on macromodels is presented. The algorithm is shown to have local quadratic convergence provided that suitable conditions on the continuity and nonsingularity of the Jacobian of the network equations are satisfied. The concept of latency for the analysis of large-scale networks in the time domain is discussed. The relationship between latency and numerical integration methods is investigated.**

I. Introduction

THE time-domain analysis of electronic circuits requires the solution of nonlinear algebraic-differential equations. Implicit integration formulas (for example [1]), modifications of the Newton–Raphson's algorithm (for example [2]) and sparse matrix techniques (for example [3]) made possible the accurate analysis of circuits containing up to hundreds of active devices within reasonable computation time. Computer programs such as SPICE [4] and ASTAP [5] have been developed by applying the previously mentioned numerical techniques.

Recent advances in large-scale integrated circuits have posed the challenge of analyzing circuits containing thousands of active devices. In this framework, the use of circuit simulation programs such as ASTAP or SPICE is not economically feasible. Recently, timing simulation [6], [7] has been proposed as a viable alternative to circuit simulation when only an approximate analysis of the digital circuit is required. Hybrid simulation [8], [9] has now been introduced to analyze circuits where various parts of the same integrated circuit must be analyzed with different accuracy. Hybrid simulation programs such as DIANA [8] and SPLICE [9] perform concurrent circuit, timing, and logic analysis of various parts of the same circuit.

In this paper, we will discuss techniques required to exploit the characteristic properties of many electronic circuits for a more efficient analysis. In particular, we will explore the fact that many of these i) consist of identical repetitive subnetworks and ii) contain subcircuits which are "inactive," i.e., their electrical variables are almost constant, for most of the simulation time. Characteristic i) can be exploited by using tearing algorithms [23]–[32] and by macromodeling [11], [17], [18]. The basic idea is to decompose the circuit into identical subcircuits and to analyze them separately.

Characteristic ii) has been used to speed up the analysis in logic simulation called "event driven" when only the active part of the circuit is analyzed. The idea of taking advantage of the latency of electronic circuits in a circuit simulation has been introduced in [11], [12]. Basically, when a subnetwork is found to be latent at a certain instant of time t_{n+1}, the corresponding elements in the Jacobian of the circuit equations are not evaluated at t_{n+1}, and the value of the subcircuit variables is set equal to the one taken at time t_n.

In this paper, we define rigorously an exact macromodel of a given nonlinear network and we give a sufficient condition on the nonlinear network equations for the existence of a macromodel. Then we propose a new multilevel Newton algorithm with local quadratic convergence properties. The algorithm is based on macromodels and effectively decomposes the network into smaller subsystems which can be analyzed separately. Then we introduce the concept of latency. The relationship between latency and numerical integration methods is investigated. The multilevel Newton algorithm with macromodels and latency has been implemented in the IBM[1] program MACRO (macromodular analysis of circuit response and operation).

Manuscript received January 22, 1979; revised May 10, 1979. This research was supported by the Joint Services Electronics Program under Contract F44620-76-C-0100.

N. B. Guy Rabbat and H. Y. Hsieh are with the Advanced Analysis Group, IBM Data Systems Division Laboratory, Hopewell Junction, NY 12533.

A. L. Sangiovanni-Vincentelli is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

[1]MACRO is described in [36].

II. Theoretical Background

A nonlinear lumped circuit can be analyzed in the time domain by solving a set of differential-algebraic nonlinear equations of the form

$$f(z(t),\dot{z}(t),t)=\boldsymbol{\theta}, \qquad T\geqslant t\geqslant 0 \tag{2.1}$$

where $z(t)\in \boldsymbol{R}^p$ is, in general, the vector of node voltages, branch voltages, branch currents, capacitor charges and inductor fluxes, $\boldsymbol{\theta}$ is the origin in $\boldsymbol{R}^p$, f: $\boldsymbol{R}^p\times\boldsymbol{R}^p\times\boldsymbol{R}^1\rightarrow\boldsymbol{R}^p$ is a continuously differentiable function with respect to $z(t)$ and $\dot{z}(t)$, and a piecewise continuous function of t. Since (2.1) is, in general, a stiff system, implicit and stiffly stable [1] integration formulas are used to solve (2.1). In particular, we will concentrate on the backward differentiation formulas introduced in [13]. According to [13], we "discretize" the operator d/dt and use a backward differentiation formula of order k to obtain

$$-h\dot{z}_{n+1}=\sum_{i=0}^{k}\alpha_i z_{n+1-i} \tag{2.2}$$

where $\dot{z}_{n+1}$ is the computed value of $\dot{z}(t_{n+1})$, z_{n+1-i} is the computed value of $z(t_{n+1-i})$, $i=0,\cdots,k$, $h\triangleq t_{n+1}-t_n$, and the α_i's are determined by the requirements that (2.2) be exact for polynomials of degree $\leqslant k$. By using (2.2), (2.1) becomes

$$\hat{f}(z_{n+1},z_n,\cdots,z_{n+1-k},t_{n+1})=\boldsymbol{\theta}, \qquad n=k-1,\cdots,q \tag{2.3}$$

where $t_{q+1}=T$, and $z_1,\cdots,z_{k-1}$ are computed by a first-order backward differentiation formula or a Runge–Kutta method (both self-starting). Since z_{n+1-i}, $i=1,\cdots,k$, have been already computed at time t_{n+1}, (2.3) can be considered as a function of z_{n+1} only. Then (2.3) can be written as

$$\boldsymbol{F}_{n+1}(z_{n+1})=\boldsymbol{\theta}, \qquad n=k-1,\cdots,q \tag{2.4}$$

where $\boldsymbol{F}_{n+1}$: $\boldsymbol{R}^p\rightarrow\boldsymbol{R}^p$ is continuously differentiable and the index $n+1$ indicates that the function is different at different instants of time. Equation (2.4) must be solved for z_{n+1}. The most commonly used methods for solving (2.4) are based on the Newton–Raphson method which consists of the following iterative schemes:

$$z_{n+1}^{j+1}=z_{n+1}^j-D\boldsymbol{F}_{n+1}(z_{n+1}^j)^{-1}\boldsymbol{F}_{n+1}(z_{n+1}^j) \tag{2.5}$$

where $D\boldsymbol{F}_{n+1}(z_{n+1}^j)^{-1}$ is the inverse of the Jacobian of $\boldsymbol{F}_{n+1}$ computed at z_{n+1}^j. It is well known that the iterations defined by (2.5) converge to z_{n+1} if the initial guess z_{n+1}^0 is sufficiently close to z_{n+1}, and that the rate of convergence is quadratic (e.g., see [14]). In order to improve convergence, it is often worth starting with an initial guess z_{n+1}^0 which is predicted by fitting a polynomial of degree k through $z_n,\cdots,z_{n-k}$. Therefore, we have

$$z_{n+1}^0=z_{n+1}^{pr}\triangleq\sum_{i=1}^{k+1}\gamma_i z_{n+1-i}. \tag{2.6}$$

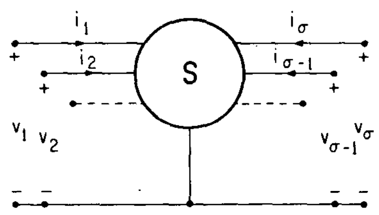


Fig. 1. The σ-port.

The truncation error of the backward differentiation formula of order k for a component $z_{r,n+1}$ of z_{n+1}, $r=1,\cdots,p$, has been proven (e.g., see [13]) to be

$$\mathcal{E}_k^r\triangleq h(\dot{z}_r(t_{n+1})-\dot{z}_{r,n+1})=E_k^r+\mathcal{O}(h^{k+2}), \qquad r=1,\cdots,p \tag{2.7}$$

where

$$E_k^r=\frac{h}{t_{n+1}-t_{n-k}}(z_{r,n+1}-z_{r,n+1}^{pr})+\mathcal{O}(h^{k+2}), \qquad r=1,\cdots,p \tag{2.8}$$

and $(z_{r,n-i}-z_r(t_{n-i}))$ is assumed to be $\mathcal{O}(h^{k+1})$.

III. Macromodels

A. Definition of Macromodels

Let $\mathfrak{N}$ be the large-scale network to be analyzed. Let $\mathfrak{N}$ consist of interconnected (possibly repetitive) subnetworks S_i, $i=1,\cdots,\rho$. In general, each of the ρ subnetworks interacts with the rest of the network only at a small number of nodes. A macromodel of a network consists of a set of nonlinear and/or time varying elements or of a set of nonlinear algebraic-differential equations [18] simulating the external behavior of the subnetwork, We make use of macromodels at the Newton–Raphson iteration level, i.e., when (2.4) is solved.

Let S be a subnetwork to be represented by a macromodel. Let $S(t_n)$ be the nonlinear companion network associated with the integration formula used at time t_n [23]. Let N, $|N|^2=\sigma+1$, be the subset of nodes of S which are connected to the rest of the network. We now pick up a node in N as a reference node and we consider S as a σ-port (see Fig. 1). Let $\boldsymbol{v}\in\boldsymbol{R}^\sigma$ and $\boldsymbol{i}\in\boldsymbol{R}^\sigma$ be the port voltages and the port currents of S. Let $\boldsymbol{y}\in\boldsymbol{R}^\sigma$, $\boldsymbol{y}\triangleq[v_1,\cdots,v_r,i_{r+1},\cdots,i_\sigma]^T$ be the vector of the *output* or *responses* of S and $\boldsymbol{u}\in\boldsymbol{R}^\sigma$, $\boldsymbol{u}\triangleq[i_1,\cdots,i_r,v_{r+1},\cdots,v_\sigma]^T$ be the vector of the *inputs* or *stimuli* of S. We assume that the interactions of S with the rest of the network take place only at the nodes in N, i.e., there is no coupling between elements of S and elements of the rest of $\mathfrak{N}$. Let

$$\boldsymbol{H}_n(\boldsymbol{u}_n,\boldsymbol{x}_n,\boldsymbol{y}_n)=\boldsymbol{\theta} \tag{3.1}$$

be the set of the nonlinear algebraic equations describing the behavior of the companion network $S(t_n)$, where $\boldsymbol{x}_n\in\boldsymbol{R}^\pi$ is the vector of "internal" variables of $S(t_n)$ and $\boldsymbol{H}_n$:

[2] $|\cdot|$ denotes the cardinality of a set.

$\boldsymbol{R}^\sigma \times \boldsymbol{R}^\sigma \times \boldsymbol{R}^\pi \to \boldsymbol{R}^{\sigma+\pi}$. Since now on all the discussion will be concerning $S(t_n)$, for the sake of notational simplicity, we drop all the subscripts n from vectors and functions. Let $\Omega \subset \boldsymbol{R}^{2\sigma+\pi}$ be the set of all the *admissible variables* for $S(t_n)$, i.e.,

$$\Omega \triangleq \{(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y}) | \boldsymbol{H}(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{\theta}\}. \tag{3.2}$$

We define an exact macromodel for $S(t_n)$ as follows.

Definition 3.1: Let $S(t_n)$ be described by (3.1). An *exact macromodel* of $S(t_n)$ is an input-output map of the form

$$\boldsymbol{y} = \boldsymbol{G}_y(\boldsymbol{u}) \tag{3.3}$$

where $\boldsymbol{G}_y\colon \boldsymbol{R}^\sigma \to \boldsymbol{R}^\sigma$, such that for all $(\hat{\boldsymbol{u}},\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) \in \Omega$, $\hat{\boldsymbol{y}} = \boldsymbol{G}_y(\hat{\boldsymbol{u}})$. □

Remark 3.2: Some papers (e.g., [17], [18]) introduce macromodels represented by circuit elements or equations which *approximate* the "external" behavior of $S(t_n)$ or of S. Our definition of macromodel is such that the external behavior of the circuit is *exactly* represented by the macromodel. □

B. Existence of Macromodels and Their Differentiability Properties

A basic question to answer is under which conditions on (3.1) a macromodel of $S(t_n)$ exists.

Assumption 3.3: Let $\boldsymbol{DH}$ denote the Jacobian of $\boldsymbol{H}$ and $\boldsymbol{DH}(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y})$ denote the Jacobian of $\boldsymbol{H}$ evaluated at the point $(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y})$. $\boldsymbol{H}$ in (3.1) is Lipschitz continuously differentiable and $\boldsymbol{DH}$ is uniformly bounded on Ω, i.e.,

a) there exists $L > 0$ such that for all pairs $(\boldsymbol{u}^1,\boldsymbol{x}^1,\boldsymbol{y}^1)$, $(\boldsymbol{u}^2,\boldsymbol{x}^2,\boldsymbol{y}^2) \in \Omega$,

$$\|\boldsymbol{DH}(\boldsymbol{u}^1,\boldsymbol{x}^1,\boldsymbol{y}^1) - \boldsymbol{DH}(\boldsymbol{u}^2,\boldsymbol{x}^2,\boldsymbol{y}^2)\| \leqslant L\|(\boldsymbol{u}^1,\boldsymbol{x}^1,\boldsymbol{y}^1) - (\boldsymbol{u}^2,\boldsymbol{x}^2,\boldsymbol{y}^2)\| \tag{3.4}$$

b) there exists $\alpha > 0$ such that for all $(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y}) \in \Omega$,

$$\|\boldsymbol{DH}(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y})\| \leqslant \alpha. \tag{3.5}$$

□

Assumption 3.4: Let $D_x\boldsymbol{H}$, $D_y\boldsymbol{H}$, $D_u\boldsymbol{H}$, $D_{x,y}\boldsymbol{H}$ denote, respectively, the Jacobians of $\boldsymbol{H}$ with respect to $\boldsymbol{x}$, to $\boldsymbol{y}$, to $\boldsymbol{u}$, and to $\boldsymbol{x}$ and $\boldsymbol{y}$. The inverse of the $(\sigma+\pi)\times(\sigma+\pi)$ matrix $D_{x,y}\boldsymbol{H}(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y})$, $D_{x,y}\boldsymbol{H}(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y})^{-1}$ exists for all $(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y}) \in \Omega$ and is uniformly bounded on Ω. □

Proposition 3.5: Suppose that Assumptions 3.3 and 3.4 hold. Then a macromodel of $S(t_n)$ exists and is Lipschitz continuously differentiable.

Proof: By the implicit function theorem (see [14, p. 128]) for all $(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y}) \in \Omega$, there exists a unique continuously differentiable function $\boldsymbol{G}\colon \boldsymbol{R}^\sigma \to \boldsymbol{R}^{\sigma+\pi}$, $\boldsymbol{G}(\boldsymbol{u}) \triangleq \begin{pmatrix} \boldsymbol{G}_x(\boldsymbol{u}) \\ \boldsymbol{G}_y(\boldsymbol{u}) \end{pmatrix}$, $\boldsymbol{G}_x\colon \boldsymbol{R}^\sigma \to \boldsymbol{R}^\pi$, $\boldsymbol{G}_y\colon \boldsymbol{R}^\sigma \to \boldsymbol{R}^\sigma$, such that

$$\boldsymbol{H}(\boldsymbol{u},\boldsymbol{G}_x(\boldsymbol{u}),\boldsymbol{G}_y(\boldsymbol{u})) = \boldsymbol{\theta} \tag{3.6}$$

and

$$\boldsymbol{DG}(\boldsymbol{u}) = \begin{pmatrix} \boldsymbol{DG}_x(\boldsymbol{u}) \\ \boldsymbol{DG}_y(\boldsymbol{u}) \end{pmatrix} = -D_{x,y}\boldsymbol{H}(\boldsymbol{u},\boldsymbol{G}_x(\boldsymbol{u}),\boldsymbol{G}_y(\boldsymbol{u}))^{-1} D_u\boldsymbol{H}(\boldsymbol{u},\boldsymbol{G}_x(\boldsymbol{u}),\boldsymbol{G}_y(\boldsymbol{u})) \tag{3.7}$$

where $\boldsymbol{DG}(\boldsymbol{u})$ is the Jacobian of $\boldsymbol{G}$ evaluated at $\boldsymbol{u}$. Then, according to Definition 3.1, $\boldsymbol{G}_y$ is a macromodel of $S(t_n)$, and $\boldsymbol{G}_y$ is continuously differentiable. Next, define the set $U \triangleq \{\boldsymbol{u} | \exists \boldsymbol{x},\boldsymbol{y} \text{ such that } (\boldsymbol{u},\boldsymbol{x},\boldsymbol{y}) \in \Omega\}$. To prove that $\boldsymbol{DG}_y$ is Lipschitz on U, it is sufficient to prove that $D_{x,y}\boldsymbol{H}(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y})^{-1}$ is Lipschitz on Ω, since the product and the composition of two Lipschitz functions is Lipschitz. By uniform boundedness of $D_{x,y}\boldsymbol{H}(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y})^{-1}$ and by Lipschitz continuity of $D_{x,y}\boldsymbol{H}$, there exists $\gamma > 0$ and $L > 0$, such that for all $(\boldsymbol{u}^1,\boldsymbol{x}^1,\boldsymbol{y}^1)$, $(\boldsymbol{u}^2,\boldsymbol{x}^2,\boldsymbol{y}^2) \in \Omega$,

$$\begin{aligned}
&\|D_{x,y}\boldsymbol{H}(\boldsymbol{u}^1,\boldsymbol{x}^1,\boldsymbol{y}^1)^{-1} - D_{x,y}\boldsymbol{H}(\boldsymbol{u}^2,\boldsymbol{x}^2,\boldsymbol{y}^2)^{-1}\| \\
&= \|D_{x,y}\boldsymbol{H}(\boldsymbol{u}^1,\boldsymbol{x}^1,\boldsymbol{y}^1)^{-1}(D_{x,y}\boldsymbol{H}(\boldsymbol{u}^1,\boldsymbol{x}^1,\boldsymbol{y}^1) \\
&\quad - D_{x,y}\boldsymbol{H}(\boldsymbol{u}^2,\boldsymbol{x}^2,\boldsymbol{y}^2))D_{x,y}\boldsymbol{H}(\boldsymbol{u}^2,\boldsymbol{x}^2,\boldsymbol{y}^2)^{-1}\| \\
&\leqslant \|D_{x,y}\boldsymbol{H}(\boldsymbol{u}^1,\boldsymbol{x}^1,\boldsymbol{y}^1)^{-1}\| \, \|D_{x,y}\boldsymbol{H}(\boldsymbol{u}^1,\boldsymbol{x}^1,\boldsymbol{y}^1) \\
&\quad - D_{x,y}\boldsymbol{H}(\boldsymbol{u}^2,\boldsymbol{x}^2,\boldsymbol{y}^2)\| \, \|D_{x,y}\boldsymbol{H}(\boldsymbol{u}^2,\boldsymbol{x}^2,\boldsymbol{y}^2)^{-1}\| \\
&\leqslant \gamma^2 L\|(\boldsymbol{u}^1,\boldsymbol{x}^1,\boldsymbol{y}^1) - (\boldsymbol{u}^2,\boldsymbol{x}^2,\boldsymbol{y}^2)\|
\end{aligned} \tag{3.8}$$

and the proof is completed. □

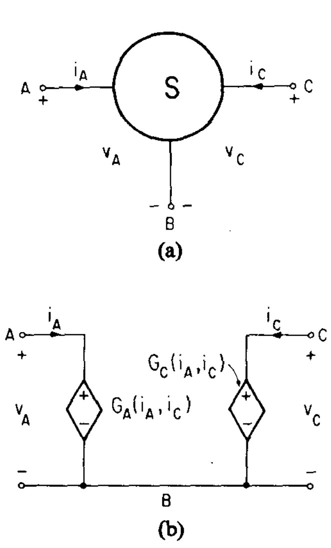


Fig. 2. An example of macromodel.

C. An Example

Consider a subnetwork S (Fig. 2(a) connected to the rest of the network at nodes A, B, and C. Node B is chosen as reference node. The output vector consists of v_A and v_C, the inputs are i_A and i_C. A macromodel is shown in Fig. 2(b), where the algebraic equations are represented by circuit elements, i.e., by two controlled sources.

IV. A Multilevel Newton Algorithm for Macromodular Networks

A. The Algorithm

For the sake of simplicity, assume that there is only one subnetwork S in $\mathfrak{N}$ which is described by its macromodel. Let the equations describing the behavior of $\mathfrak{N}$ at time t_n be written as

$$F(u, G_y(u), w) = \theta \tag{4.1}$$

where $w \in R^p$ is the vector of network variables in $\mathfrak{N}$ not interacting with S, F: $R^\sigma \times R^\sigma \times R^p \to R^{\sigma+p}$ and G_y: $R^\sigma \to R^\sigma$ is the macromodel of $S(t_n)$. Newton's algorithm applied to (4.1) consists of the following scheme:

$$\begin{aligned}\big(D_uF(u, G_y(u), w) &+ D_GF(u, G_y(u), w)\\ &\cdot DG_y(u), D_wF(u, G_y(u), w)\big)\begin{pmatrix}\Delta u\\ \Delta w\end{pmatrix}\\ &+ F(u, G_y(u), w) = \theta.\end{aligned} \tag{4.2}$$

Thus to apply Newton's method we need to evaluate $G_y(u)$ and $DG_y(u)$. According to Definition 3.1, the macromodel $G_y(u)$ is implicitly determined by the nonlinear system

$$H(u,x,y) = \theta. \tag{4.3}$$

To evaluate $G_y(u)$, we can use a second Newton process on (4.3) which yields

$$D_{x,y}H(u,x,y)\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} + H(u,x,y) = \theta. \tag{4.4}$$

This second Newton process is at a lower level since u is determined from (4.2) and held fixed in (4.4). Now, if (4.3) is solved precisely, then the error in the evaluation of the macromodel and its derivative is zero and when these are used in (4.2), we have a true Newton iteration with local quadratic convergence. However, if the macromodel and its derivative are not determined precisely, then the question of quadratic convergence is open. The idea we propose to retain local quadratic convergence in the presence of error is as follows. It would seem to make no sense to solve (4.3) to a higher precision than the current iteration for (4.1) and it would only seem necessary to tighten the convergence control for (4.4) at the same rate (4.2) is converging. Hence, iteration (4.4) is stopped whenever

$$\|(\Delta x, \Delta y)\| \leqslant \|(\Delta u, \Delta w)\|^2. \tag{4.5}$$

To distinguish this from the true Newton process (4.2) when (4.3) is solved precisely, we call our process a multilevel Newton iteration. In the algorithm, G_y^{app} and $D^{\text{app}}G_y$ denote, respectively, the computed approximations of G_y and of DG_y.

MultiLevel Newton Algorithm (MLNA) Algorithm 4.1:

Parameter: $\tau^0 \in R_+$.

Data: $u^0 \in R^\sigma$, $w^0 \in R^p$, $x^{0,0} \in R^\pi$, $y^{0,0} \in R^\sigma$.

Step 0: Initialization. Set $i = 0$.

Step 1: Initialization of the lower level Newton algorithm. Set $k = 0$.

Step 2: Compute $(x^{i,k+1}, y^{i,k+1}) = (x^{i,k}, y^{i,k}) + (\Delta x, \Delta y)$ by solving

$$D_{x,y}H(u^i, x^{i,k}, y^{i,k})\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} + H(u^i, x^{i,k}, y^{i,k}) = \theta.$$

Step 3: If $\|(\Delta x, \Delta y)\| \geqslant \tau^i$, set $k = k+1$, compute $H(u^i, x^{i,k}, y^{i,k})$, $D_{x,y}H(u^i, x^{i,k}, y^{i,k})$ and go to Step 2. Else continue.

Step 4: Exit from inner loop. Set $(x^{i+1,0}, y^{i+1,0}) = (x^{i,k+1}, y^{i,k+1})$, $G_y^{\text{app}}(u^i) = y^{i+1,0}$ and compute $D^{\text{app}}G_y(u^i)$ from

$$D^{\text{app}}G(u^i) = -D_{x,y}H(u^i, x^{i+1,0}, y^{i+1,0})^{-1} \cdot D_uH(u^i, x^{i+1,0}, y^{i+1,0}).$$

Step 5: Set $(u^{i+1}, w^{i+1}) = (u^i, w^i) + (\Delta u, \Delta w)$ by solving

$$\begin{aligned}&\big(D_uF(u^i, G_y^{\text{app}}(u^i), w^i) + D_GF(u^i, G_y^{\text{app}}(u^i), w^i)\\ &\cdot D^{\text{app}}G_y(u^i), D_wF(u^i, G_y^{\text{app}}(u^i), w^i)\big)\\ &\cdot\begin{pmatrix}\Delta u\\ \Delta w\end{pmatrix} + F(u^i, G_y^{\text{app}}(u^i), w^i) = \theta.\end{aligned}$$

Step 6: Set $\tau^{i+1} = \min\{\tau^0, \|(\Delta u, \Delta w)\|^2\}$, $i = i+1$ and go to Step 1. □

Remark 4.2: The initial data u^0, w^0, $x^{0,0}$, $y^{0,0}$ are obtained by formula of the form (2.6). □

Remark 4.3: When used to integrate (2.1), MLNA stops when $\|(\Delta u, \Delta w)\| \leqslant \eta$, where η is a positive real number. This number must be chosen so that the error made is not larger than the local truncation error of the integration rule used. □

Remark 4.4: The equations describing the network $S(t_n)$, $H(u,x,y) = \theta$ can be formulated by any analysis method (e.g., nodal analysis, modified nodal analysis [33], tableau analysis [3]) irrespective of the method chosen to formulate the equations for the rest of the network. In principle, we may also use an integration formula for the subnetwork S which is different from the one used for the rest of the network. In particular a zeroth-order integration method could be used if this satisfies the error criteria. The condition under which a zeroth-order method is accepted is called latency and is the subject of the next section. □

Remark 4.5: In the practical implementation of MLNA, we perform the LU decomposition of $D_{x,y}H(u,x,y)$ and the back substitution for x and y in symbolic form. This step needs to be performed only once per analysis. It can speed up the computation in the inner loop of MLNA, since now, Step 2 involves only function evaluations to obtain the entries of the LU factors and the coefficients for the backward substitution. □

Remark 4.6: The evaluation of $D^{\text{app}}G_y(u^i)$ can be easily performed by using the following algorithm.

Evaluation of $D^{\text{app}}G_y(u^i)$ Algorithm 4.2

Step 1: Solve the following σ systems of linear algebraic equations

$$D_{x,y}H(u^i,x^{i+1,0},y^{i+1,0})^T M = \begin{pmatrix} \mathbf{0}_\pi \\ I_\sigma \end{pmatrix}$$

for the $(\sigma+\pi)\times\sigma$ matrix M, where I_σ is the $\sigma\times\sigma$ identity matrix and $\mathbf{0}_\pi$ is the $\pi\times\sigma$ null matrix.

Step 2: Compute

$$D^{\text{app}}G_y(u^i) = -M^T D_u H(u^i,x^{i+1,0},y^{i+1,0}). \qquad \square$$

Note that Step 1 requires only a partial forward elimination and a complete backward substitution for each of the σ systems, since the LU factorization of $D_{x,y}H(u^i, x^{i+1,0},y^{i+1,0})$ is available from Step 2 of MLNA. □

Remark 4.7: It is quite straightforward to derive MLNA for any number of macromodels. The only difficulties are the rather complicated notation and bookkeeping involved. □

Remark 4.8: If the network contains several identical copies of a subnetwork, then the use of a macromodel to describe this subnetwork is particularly convenient. In this case, the symbolic LU decomposition of $D_{x,y}H(u,x,y)$ and the symbolic back substitution for x and y can be performed only once independently on how many copies of the subnetwork are present in $\mathfrak{N}$. □

Remark 4.9: The application of MLNA does not require any particular structure of the Jacobian of the circuit equations. Tearing methods [23]–[32] do require a bordered block diagonal form or a bordered block triangular form of the Jacobian of the circuit equations. □

B. Convergence Properties of MLNA

As previously pointed out, MLNA is a Newton–Raphson algorithm with errors in the computation of F and of DF. In the Appendix we prove the following local convergence result.

Theorem 4.10: Let $(\hat{u},\hat{w})$ be such that $F(\hat{u},G_y(\hat{u}),\hat{w})=\theta$. Assume that

1) F is Lipschitz continuously differentiable;

2) $J(\hat{u},\hat{w})^{-1}$ exists, where

$$J(u,w) \triangleq \big(D_u F(u,G_y(u),w) + D_G F(u,G_y(u),w)DG_y(u), D_w F(u,G_y(u),w)\big);$$

3) Assumptions 3.3 and 3.4 hold;

4) for all i, $u^i \in U$;

5) for all i, $(x^{i,0},y^{i,0})$ are such that the inner Newton loop converges.

Then, there exists $\delta>0$ such that for all $(u^0,w^0)\in B((\hat{u},\hat{w}),\delta)$, $u^0\in U$; for all $\tau^0\in[0,\delta]$, MLNA converges to $(\hat{u},\hat{w})$ with root convergence order greater than or equal to two. □

Remark 4.11: If we choose to drive τ of MLNA to zero as fast as $\|(\Delta u,\Delta w)\|$ or even independently of $\|(\Delta u,\Delta w)\|$, we can still prove convergence of MLNA but not quadratic convergence. To achieve quadratic convergence, it is crucial to drive τ to zero as fast as $\|(\Delta u,\Delta w)\|^2$. The strategy followed in driving the error to zero is similar to the one presented in [19], [20] for optimization algorithms. □

Remark 4.12: MLNA can be seen as a relaxation method. In fact, when we enter the inner Newton loop for each macromodel, we hold fixed all the external variables interacting with that macromodel. However, MLNA achieves quadratic convergence, while usual relaxation methods achieve *only linear convergence.* □

Remark 4.13: It may happen that for some u^i, $x^{i,0}$,$y^{i,0}$ the inner loop does not converge during the analysis at time t_{n+1}. To improve convergence of the inner loop, we may halve the step size, compute x and y at the intermediate time point so obtained and use these as initial guesses for t_{n+1}. When the step size is halved, u^i at the intermediate time point is computed by interpolating a polynomial of kth degree through $u_n, u_{n-1},\cdots,u_{n-k}$. Therefore, the analysis at the intermediate time point is performed only for the macromodel which does not converge at time t_{n+1} and not for the entire network. □

V. Latency

Suppose that the network to analyze consists of many repetitive subnetworks, possibly described by macromodels. In many cases, such as in the digital network analysis case, at any one time most of the subnetworks are inactive or latent, i.e., the value of their electrical variables remain constant. Moreover, each subnetwork is latent for most of the time. These considerations have led to the development of efficient logic simulators (e.g., see [15]). We now show how it is possible to exploit latency in the solution of (2.1). If at any time t_n the value of the variables of a subnetwork is found to be constant, then obviously no function or Jacobian evaluations are needed to find the value of the subnetwork variables at all the subsequent time steps until a change in the input variables of the subnetwork occurs. It has been reported [16] that up to 80 percent of computer time for the circuit simulation program SPICE is spent in the evaluation of F_{n+1} in (2.4) and of its Jacobian. We believe that this situation is typical for the most sophisticated circuit analysis programs. Therefore, the use of latency can achieve significant savings in computer time. In order to apply latency, we need a test to detect if a subnetwork S is latent at time t_n. Note that the integration of (2.1) for a latent subnetwork is done simply by setting

$$z_{n+1}=z_n. \tag{5.1}$$

Now (5.1) can be considered as a particular integration method, a zeroth-order method! In fact, (5.1) is exact for a zero degree polynomial. The local truncation error of this method cannot be computed by means of (2.8) since the zeroth-order method is an explicit method. However, it can be easily obtained. By (5.1), and by definition of local truncation error, we have

$$\begin{aligned} z_r(t_{n+1})-z_{r,n+1} &= z_r(t_{n+1})-z_{r,n} \\ &= z_r(t_{n+1})-z_r(t_n). \end{aligned} \tag{5.2}$$

By the mean value theorem [14, p. 68], we have

$$|z_r(t_{n+1})-z_r(t_n)| = |\dot{z}_r(t^*)|h = E_0^r + \mathcal{O}(h^2) \tag{5.3}$$

where $t^* \in [t_{n+1}, t_n]$ and $E_0^r \triangleq |\dot{z}_r(t_n)|h$. Since (2.1) is not in canonical state equation form, $\dot{z}_r(t)$ can be estimated by finite differences as

$$|\dot{z}_r(t_n)| \approx |z_{r,n} - z_{r,n-1}|/h_1. \tag{5.4}$$

Therefore, a zeroth-order method should be used whenever E_0^r computed by means of the approximation in (5.4) is less than or equal to the local truncation error prescribed by the user on z_r. However, since a zeroth-order method is an explicit method which involves no evaluation of $\dot{z}_r(t)$ once it has been applied, there is no feedback to correct an error which may grow larger than expected. Hence, it is safe to be conservative in the evaluation of the local truncation error for a zeroth-order method. Two possible ways of being conservative are expressed by the following formulas for approximating $|\dot{z}_r(t_n)|$.

Conservative approximation 5.1

$$|\dot{z}_r(t_n)| \approx \max_{j=1,\cdots,\hat{j}} |z_{r,n-j+1} - z_{r,n-j}|/h_j.$$

Conservative approximation 5.2

$$|\dot{z}_r(t_n)| \approx \max_{j=1,\cdots,\hat{j}} \left(|z_{r,n} - z_{r,n-j}| \Big/ \sum_{i=1}^{j} h_i \right).$$

In both formulas $\hat{j}$ is an integer larger than or equal to 1. When $\hat{j} = 1$, both formulas coincide with (5.4). Other conservative approximations for E_0^r can be invented by introducing approximations to higher order derivatives of $z_r(t)$.

As previously pointed out, once the zeroth-order method is applied we have no feedback and we may continue to use it even when the value of the variables in the network to be analyzed starts changing due to a variation in the input. Therefore, we set as another condition for the application of the zeroth-order method that the inputs remain almost constant. The conditions under which a zeroth-order method is accepted are called *latency* and the network which satisfies these conditions is said to be *latent*.

Definition 5.3: Let S be a time invariant nonlinear subnetwork of a network $\mathfrak{N}$. Let $\boldsymbol{u}(t) \in \boldsymbol{R}^{\sigma}$ be the vector of inputs, $\boldsymbol{x}(t) \in \boldsymbol{R}^{\pi}$, the vector of internal variables, and $\boldsymbol{y}(t) \in \boldsymbol{R}^{\sigma}$, the vector of outputs of S at time t. S is said to be *latent* at time t_{n+1} if

1) $y_{n+1-j}, x_{n+1-j}, j = 1, \cdots, \hat{j}$, obtained by the integration methods used at previous $\hat{j}$ steps, are such that the truncation error computed according to an appropriate conservative estimate (e.g., by means of 5.1 or 5.2) E_0^r for all components of $\boldsymbol{x}$ and $\boldsymbol{y}$ is less than or equal to the local truncation error specified by the user,

2) $\max_{j=0,\cdots,j^*} |u_s(t_{n+1}) - u_s(t_{n-j})| \leq \epsilon_s, \quad s = 1, \cdots, \sigma,$ where j^* is a positive integer and $\boldsymbol{\epsilon} \in \boldsymbol{R}_+^{\sigma}$ is an error vector supplied by the user. □

Remark 5.4: Latency can be used very effectively in conjunction with MLNA. In particular if a subnetwork satisfies the tests of Definition 5.3 at t_{n+1}, then no inner Newton loop is needed, and $\boldsymbol{G}_{y_n}(\boldsymbol{u}_n)$ is set equal to $\boldsymbol{G}_{y_{n-1}}(\boldsymbol{u}_{n-1})$. Moreover, $D\boldsymbol{G}_{y_n}(\boldsymbol{u}_n)$ is set equal to $D\boldsymbol{G}_{y_{n-1}}(\boldsymbol{u}_{n-1})$. □

Remark 5.5: The investigation of MLNA with the use of latency has shown that the solution by means of first-order backward differentiation formulas with the conservative approximation 5.2 should be as accurate as the error controls used. We analyzed a simple inverter circuit with one bipolar device model. A macromodel is created for the bipolar device model by means of the internal Newton loop of MLNA. We also analyzed the network by means of a standard Newton algorithm. The results obtained showed savings between the analysis performed using MLNA and the one performed using the standard Newton algorithm.

VI. Concluding Remarks

Macromodels by themselves offer an improvement over the usual tableau and nodal methods in circuit with multiple copies of similar subcircuits. Combined with a multilevel Newton process, macromodels effectively decompose the network into smaller systems which can be analyzed separately. We have shown how this can be done without affecting the usual quadratic rate of convergence of Newton's method.

With such a decomposition, it is then possible to treat each subcircuit as an entity in itself. Thus one can use different time steps, different methods of numerical integration, etc., on each subcircuit. Of particular interest is the possibility of using a zeroth-order numerical method when it can be determined not to affect the accuracy. Such a subcircuit is said to be latent. Numerical computations are saved in processing a latent subcircuit since the zeroth-order method keeps all variables constant. Thus the same solution $\boldsymbol{y}_n = \boldsymbol{G}_{y_n}(\boldsymbol{u}_n)$ and Jacobian $D\boldsymbol{G}_{y_n}(\boldsymbol{u}_n)$ can be reused in the upper level Newton iteration without recomputation.

Appendix

The proof of Theorem 4.10 is based on the following main result.

Newton Perturbation Theorem: Consider the sequence $\{\boldsymbol{v}^i\}$ generated by the Newton perturbed process

$$\boldsymbol{v}^{i+1} = \boldsymbol{v}^i - \hat{J}(\boldsymbol{v}^i, \boldsymbol{h}^i)^{-1} \hat{\boldsymbol{F}}(\boldsymbol{v}^i, \boldsymbol{h}^i) \tag{A.1}$$

where $\boldsymbol{v} \in \boldsymbol{R}^n$, $\boldsymbol{h} \in \boldsymbol{R}^m$, $\hat{F}$: $\boldsymbol{R}^n \times \boldsymbol{R}^m \rightarrow \boldsymbol{R}^n$, and $\hat{J}(\boldsymbol{v}, \boldsymbol{h}) \in \boldsymbol{R}^{n \times n}$. Assume that

1) $\hat{\boldsymbol{F}}(\boldsymbol{v}^*, \boldsymbol{\theta}) = \boldsymbol{\theta}$;

2) $\hat{\boldsymbol{J}}(\boldsymbol{v}, \boldsymbol{\theta}) = D\hat{\boldsymbol{F}}(\boldsymbol{v}, \boldsymbol{\theta})$;

3) $\hat{\boldsymbol{J}}(\boldsymbol{v}^*, \boldsymbol{\theta})^{-1}$ exists;

4) for all i, $\|\boldsymbol{h}^i\| \leq \eta \|\boldsymbol{v}^i - \boldsymbol{v}^{i-1}\|^4$, $\eta > 0$;

5) there exists $\delta > 0$ such that for all $\boldsymbol{v} \in B(\boldsymbol{v}^*, \delta)$, $\hat{\boldsymbol{J}}(\boldsymbol{v}, \cdot)^{-1} \hat{\boldsymbol{F}}(\boldsymbol{v}, \cdot)$ is well defined and Lipschitz on $B_h(\boldsymbol{\theta}, \delta)$, where $B_h(\boldsymbol{\theta}, \delta) \triangleq \{\boldsymbol{h} | \|\boldsymbol{h}\| \leq \delta\}$.

Then there exists $\hat{\delta}>0$ such that for all $v^0 \in B(v^*,\hat{\delta})$, $h^0 \in B_h(\theta,\hat{\delta})$, $\{v^i\}$ converges to v^* with root convergence order greater than or equal to two.

Proof: By (A.1) and assumption 5, for all $v^i \in B(v^*,\delta)$, $h^i \in B_h(\theta,\delta)$,

$$\begin{aligned}\|v^*-v^{i+1}\| &= \|v^*-v^i+\hat{J}(v^i,h^i)^{-1}\hat{F}(v^i,h^i)\| \\ &\leqslant \|v^*-v^i+\hat{J}(v^i,\theta)^{-1}\hat{F}(v^i,\theta)\| \\ &\quad + \|\hat{J}(v^i,h^i)^{-1}\hat{F}(v^i,h^i)-\hat{J}(v^i,\theta)^{-1}\hat{F}(v^i,\theta)\|. \end{aligned} \tag{A.2}$$

By Assumptions 1, 2, 3, and Theorem 10.2.2, [14, p. 312], we have that there exists $\delta'>0$, $\delta'\leqslant\delta$, and $\alpha>0$ such that for all $v^i \in B(v^*,\delta')$,

$$\|v^*-v^i+\hat{J}(v^i,\theta)\hat{F}(v^i,\theta)\| \leqslant \alpha\|v^*-v^i\|^2. \tag{A.3}$$

Therefore, by Assumptions 4 and 5, we have that for all $v^i \in B(v^*,\delta')$,

$$\begin{aligned}\|v^*-v^{i+1}\| &\leqslant \alpha\|v^*-v^i\|^2+\eta\|v^i-v^{i-1}\|^4 \\ &\leqslant \alpha\|v^*-v^i\|^2+8\eta\left(\|v^*-v^i\|^4+\|v^*-v^{i-1}\|^4\right).\end{aligned} \tag{A.4}$$

Let $\|v^*-v^i\|$, $\|v^*-v^{i-1}\|<\delta''\leqslant\delta'$. Then there exists $\gamma\geqslant 1/2$ such that

$$\|v^*-v^{i+1}\| \leqslant \gamma\left(\|v^*-v^i\|^2+\|v^*-v^{i-1}\|^4\right). \tag{A.5}$$

It is easily demonstrated by induction that

$$\|v^*-v^{i+n}\| \leqslant \frac{1}{2\gamma}(2\gamma\delta'')^{2^n} \leqslant (2\gamma\delta'')^{2^n}. \tag{A.6}$$

Hence,

$$\limsup_{n\to\infty} \|v^*-v^{i+n}\|^{2^{-n}} \leqslant 2\gamma\delta''. \tag{A.7}$$

Thus if δ'' is chosen so that $2\gamma\delta''<1$, then according to [14, pp. 287–293], the sequence $\{v^i\}$ converges to v^* with root convergence order greater than or equal to two. Therefore, the sequence will converge to v^* if v^0 is chosen so that $\|v^*-v^1\|$, $\|v^*-v^0\|<\delta''$, i.e., if v^0, $v^1 \in \mathring{B}(v^*\delta'')$ is the interior of $B(v^*,\delta'')$. Since

$$\|v^1-v^0\| = \|\hat{J}(v^0,h^0)^{-1}\hat{F}(v^0,h^0)\| \tag{A.8}$$

by Assumptions 1 and 5, there exists $\hat{\delta}<\delta''/2$ such that for all $v^0 \in \mathring{B}(v^*,\hat{\delta})$, $h^0 \in B_h(\theta,\hat{\delta})$

$$\|v^1-v^0\| = \|\hat{J}(v^0,h^0)^{-1}\hat{F}(v^0,h^0)\| < \delta''/2. \tag{A.9}$$

Therefore, for all $v^0 \in \mathring{B}(v^*,\hat{\delta})$, $h^0 \in B_h(\theta,\hat{\delta})$

$$\|v^1-v^*\| \leqslant \|v^1-v^0\|+\|v^0-v^*\| < \delta''/2+\hat{\delta}<\delta'' \tag{A.10}$$

and the proof is completed. □

Remark A.1: Several Newton perturbation results have been obtained in the literature (e.g., see [14], [21], [22]). Most of these results are related to Newton processes where the Jacobian is computed approximately. To the best of our knowledge, there is no result available when not only the Jacobian but also the function F is computed with approximations. □

In order to apply the theorem above, we set

$$v^i \triangleq (u^i,w^i), \quad h^i \triangleq G^{\text{app}}(u^i)-G(u^i), \quad h^i=\begin{bmatrix} h_x^i \\ h_y^i \end{bmatrix}$$

$$h_x^i \triangleq G_x^{\text{app}}(u^i)-G_x(u^i), \quad h_y^i \triangleq G_y^{\text{app}}(u^i)-G_y(u^i)$$

$$\hat{F}(v^i,h^i) \triangleq F\left(u^i,G_y^{\text{app}}(u^i),w^i\right) = F\left(u^i,G_y(u^i)+h_y^i,w^i\right)$$

and

$$\begin{aligned}\hat{J}(v^i,h^i) \triangleq \big(&D_uF\left(u^i,G_y(u^i)+h_y^i,w^i\right) \\ &+D_GF\left(u^i,G_y(u^i)+h_y^i,w^i\right) \\ &\cdot A(u^i,h^i), D_wF\left(u^i,G_y(u^i)+h_y^i,w^i\right)\big)\end{aligned}$$

where $A(u^i,h^i) \triangleq D^{\text{app}}G_y(u^i)$ is the result of the computation of Algorithm 4.2. By these definitions, it is clear that assumptions 1 and 2 of the Newton perturbation theorem are satisfied. Assumption 3 is equivalent to Assumption 2 of Theorem 4.10. Hence, we only need to show that the assumptions of Theorem 4.10 imply Assumptions 4 and 5 of the Newton perturbation theorem.

Proposition A.2: Under the assumptions of Theorem 4.10, there exist $\delta^*>0$ and $\eta>0$ such that for all $\tau^0 \in [0,\delta^*]$,

$$\|h^i\| \triangleq \|G^{\text{app}}(u^i)-G(u^i)\| \leqslant \eta\|v^i-v^{i-1}\|^4.$$

Proof: By Step 4 of MLNA, since the inner loop converges by hypothesis,

$$\|\left(x^{i,\bar{k}},y^{i,\bar{k}}\right)-\left(x^{i,\bar{k}-1},y^{i,\bar{k}-1}\right)\| < \min\{\tau^0,\|v^i-v^{i-1}\|^2\} \tag{A.11}$$

where $\bar{k}$ is the index of the inner iteration when the lower Newton loop stops. By the Newton–Mysovskii theorem [14, p. 412], there exist δ^*, $\eta>0$ such that for all $\tau^0 \in [0,\delta^*]$,

$$\begin{aligned}\|h^i\| &= \|G^{\text{app}}(u^i)-G(u^i)\| \\ &\leqslant \eta\|\left(x^{i,\bar{k}},y^{i,\bar{k}}\right)-\left(x^{i,\bar{k}-1},y^{i,\bar{k}-1}\right)\|^2 \\ &\leqslant \eta\|v^i-v^{i-1}\|^4.\end{aligned} \tag{A.12}$$

□

The next step consists of proving that there exists $\delta^*\geqslant\hat{\delta}>0$ such that for all $v \in B(v^*,\hat{\delta})$, $\hat{J}(v,\cdot)^{-1}\hat{F}(v,\cdot)$ is well defined and Lipschitz on $B_h(\theta,\hat{\delta})$. To prove this, we only need to prove that $\hat{F}(v,\cdot)$ is Lipschitz in h and that $\hat{J}(v,\cdot)$ is well defined and Lipschitz on $B_h(\theta,\hat{\delta})$. The first fact follows immediately from the assumption that F is Lipschitz continuously differentiable. To prove that $J(v,\cdot)^{-1}$ is well defined and Lipschitz we need the following lemma.

Lemma A.3: Under the assumptions of Theorem 4.10, there exists $\delta' > 0$, such that $\hat{J}(v, \cdot)$ is well defined and Lipschitz in h on $B_h(\theta, \delta')$.

Proof: By definition of $\hat{J}(v,h)$ and by Assumption 1 of Theorem 4.10, we only need to show that there exists $\delta' > 0$ such that $A(u,h)$ is well defined and Lipschitz in h on $B_h(\theta, \delta')$. By Assumptions 3.3 and 3.4, by a result in [14, 2.3.3, p. 46], following the second part of the proof of Proposition 3.5, this result can be easily established. □

Proposition A.4: Under the assumptions of Theorem 4.10, there exists $\delta' \geq \delta^* > 0$ such that for all $v \in B(v^*, \delta^*)$, for all $h \in B_h(\theta, \delta^*)$, $J(v,h)^{-1}$ is well defined and Lipschitz in h.

Proof: Since by Assumption 2 of Theorem 4.10, $\hat{J}(v^*, \theta)^{-1}$ exists, the proof follows from a result in [14, 2.3.3, p. 46]. □

It is now trivial to apply the Newton perturbation theorem and prove Theorem 4.10.

Acknowledgment

The authors are particularly grateful to Dr. R. K. Brayton for his careful review of the manuscript, for the many fruitful discussions and suggestions which helped in improving the presentation of the results of this paper.

The authors wish also to thank Drs. A. J. Jimenez, A. E. Ruehli, R. C. Joy, G. D. Hachtel, and F. H. Branin for the technical discussions and moral support in the course of this research, which culminated in the development of MACRO.

A. Sangiovanni-Vincentelli gratefully acknowledges the inspiring discussion on convergence properties of root-finding and optimization algorithms, and the continuous encouragement and help of Prof. E. Polak.

References

[1] C. W. Gear, "The automatic integration of ordinary differential equations," *Information Processing 68*, A. F. H. Morrel, Ed. Amsterdam, North-Holland, The Netherlands: 1968, pp. 187–193.

[2] D. A. Calahan, *Computer Aided Network Design*, revised ed. New York:McGraw-Hill, 1972.

[3] G. D. Hachtel, R. K. Brayton, and F. G. Gustavson, "The sparse tableau approach to network analysis and design," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 101–112, Jan. 1971.

[4] L. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Electronics Research Laboratory Rep. No. ERL-M520, University of California, Berkeley, May 1975.

[5] W. T. Weeks, A. J. Jimenez, G. W. Mahoney, D. Mehta, H. Qassemzadeh, and T. R. Scott, "Algorithms for ASTAP—A network analysis program," *IEEE Trans. Circuit Theory*, vol. CT-20, pp. 628–634, Nov. 1973.

[6] B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS—An MOS timing simulator," *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 301–310, Dec. 1975.

[7] J. D. Crawford *et. al.*, "MOTIS-C users guide," Electronics Research Laboratory, University of California, Berkeley, June 1978.

[8] G. Arnout and H. J. DeMan, "The use of threshold functions and Boolean-controlled network elements for macromodelling of LSI circuits," *IEEE J. Solid-State Circuits*, vol. SC-13, pp. 326–332, June 1978.

[9] A. R. Newton, "The simulation of large scale integrated circuits," Ph.D. dissertation, University of California, Berkeley, Electronics Research Laboratory Rep. No. UCB/ERL M-78/52, July 1978.

[10] "LOGCAP II user's guide," National Semiconductor, Norwalk, Conn.

[11] N. B. Rabbat and H. Y. Hsieh, "A latent macromodular approach to large-scale sparse networks," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 745–752, Dec. 1976.

[12] N. B. Rabbat and H. Y. Hsieh, "Concepts of latency in the time-domain solution of nonlinear differential equations," *Proc. IEEE Symp. Circuits and Systems*, pp. 813–826 (New York, May 1978).

[13] R. K. Brayton, F. G. Gustavson, and G. D. Hachtel, "A new efficient algorithm for solving differential-algebraic systems using implicit backward differential formulas," *Proc. IEEE*, vol. 60, pp. 98–108, Jan. 1972.

[14] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic Press, 1970.

[15] P. Wilcox and A. Rombeck, "F/LOGIC, an interactive fault and logic simulator for digital circuits," *Proc. 13th ACM Design Automation Workshop*, pp. 68–73, 1976.

[16] A. R. Newton and D. O. Pederson, "Analysis time, accuracy and memory requirement tradeoffs in SPICE 2," *Proc. 11th Annu. Asilomar Conf. on Circuits, Systems, and Computers* (Asilomar, California, Nov. 1977).

[17] M. Y. Hsueh, A. R. Newton, and D. O. Pederson, "The development of macromodels for MOS timing simulators," *Proc. IEEE Symp. on Circuits and Systems*, pp. 1–4, (New York, May 1978).

[18] M. H. Heydemann, "Functional macromodeling of electrical circuits," *ibid.*, pp. 532–535.

[19] R. Klessig and E. Polak, "An adaptive precision gradient method for optimal control," *SIAM J. Contr.*, vol. 11, no. 1, pp. 80–93, Feb. 1977.

[20] H. Mukai and E. Polak, "On the use of approximation in algorithms for optimization problems with equality and inequality constraints," *SIAM J. Numer. Anal.*, vol. 15, no. 4, pp. 674–693, 1978.

[21] E. Polak, "On the global stabilization of locally convergent algorithms for optimization and root finding," *Proc. 6th IFAC World Congress on Computational Methods in Control* (Boston, MA, Aug. 1975).

[22] E. Polak and I. Teodoru, "Newton derived methods for nonlinear equations and inequalities," *Proc. Mathematical Programming Symp.*, (University of Wisconsin, Madison, Apr 1974).

[23] G. Kron, *Diakoptics—Piecewise Solution of Large-Scale Systems*. London, England: MacDonald, 1963.

[24] F. H. Branin, Jr., "A sparse matrix modification of Kron's method of piecewise analysis," *Proc. of IEEE Int. Symp. on Circuits and Systems*, pp. 383–386, (Boston, MA).

[25] A. J. Jimenez, "Experiences in the modular analysis of circuits," *Int. Symp. Simulation and Software*, (Virginia Polytechnic Institute, Blacksburg, VA, Mar. 1977).

[26] L. O. Chua and L. K. Chen, "Diakoptic and generalized hybrid analysis," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 694–706.

[27] F. F. Wu, "Solution of large-scale networks by tearing," *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 706–713, Dec. 1976.

[28] A. Sangiovanni-Vincentelli, L. K. Chen and L. O. Chua, "A new tearing approach—Node tearing nodal analysis," *Proc. IEEE Symp. Circuits and Systems*, pp. 143–147 (Phoenix, AZ, Apr. 1977).

[29] A. George, "On block elimination for sparse linear systems," *SIAM J. Numer. Anal.*, vol. 11, no. 3, pp. 585–603, June 1974.

[30] G. Guardabassi and A. Sangiovanni-Vincentelli, "A two levels algorithm for tearing," *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 783–791, Dec. 1976.

[31] A. Sangiovanni-Vincentelli, L. K. Chen and L. O. Chua, "Three decomposition-based solution methods for solving a large system of linear equations," *Proc. IEEE Symp. Circuits and Systems* (New York), pp. 582–586.

[32] I. Hajj, "Sparsity considerations in network solution by tearing," *ibid.*, pp. 170–174.

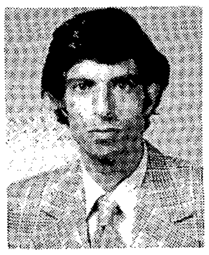
[33] C. W. Ho, A. E. Ruehli, and P. A. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 504–509, June 1975.

[34] N. B. Rabbat and H. Y. Hsieh, "Macromodularity, latency and latent graph theory," IBM Tech. Rep. No. TR 22.2067, Oct. 1976.

[35] N. B. Rabbat and H. Y. Hsieh, "Latency and latent graph theory as applied to the solution of nonlinear differential equations," IBM Tech. Rep. No. TR 22.2173, East Fishkill, Hopewell Junction, New York, Jan. 5, 1978.

[36] H. Y. Hsieh and N. B. Rabbat, "Macroanalysis of large digital networks by latent techniques," to be published.

[37] N. B. Rabbat, H. Y. Hsieh, and A. E. Ruehli, "Macromodeling for the analysis of large-scale networks," *IEEE Int. Conv. Rec.*, Electro 76, pp. 21-4-1 to 21-4-8, *Computer-Aided Design I*, Boston, MA, May 11–14, 1976.



N. B. Guy Rabbat (M'68–SM'78) was born on January 30, 1943. He received the French Baccalaureate with highest honors in 1961, the B.S. degree from Cairo University, Cairo, Egypt, in 1967, the M.S. and Ph.D degrees from Queen's University, Belfast, Northern Ireland, U.K., in 1969 and 1971, all in electrical engineering.

From 1963–1968, he worked part-time in West Germany, especially at Siemens A.G. From 1968–1972, he was a Research Assistant and Assistant Lecturer in the Department of Electrical and Electronic Engineering, Queens University. From 1973–1974, he was an Assistant Professor in the Department of Electrical Engineering, the University of Alabama, Birmingham. From 1972–1975, he was a full-time Research Associate (1972–1973) and part-time Lecturer and Affiliate Professor at Washington University, St. Louis, MO. Since May 1974, he has been with the Computer-Aided Circuit Design Group of the IBM Data Systems Division Laboratory, where he is now an Advisory Engineer. His research interests include work in macromodeling, latency and latent graph theory, symbolic analysis, graph theory, lumped-distributed networks, transmission lines and coupled lines, integrated circuit design, computer hardware system design, microprocessors and fault-tolerant computing. He is the author of two circuit analysis programs.

Dr. Rabbat is a member of Sigma Xi. He is the author or co-author of over 60 technical papers and has 18 technical disclosures and patents. In 1978, he was awarded an IBM First Invention Achievement Award for his work in macromodeling. Since 1977, he has been a Member of the Large-Scale Systems Technical Committee of the IEEE Circuits and Systems Society, where he is now Chairman and a Member of the Membership Development and Services Committee. Since 1975, he has been a Member of the Executive Committee of the IEEE Mid-Hudson Section, where he is now Chairman. He was also the organizer and first Chairman of the IEEE Mid-Hudson Chapter on Circuits and Systems in (1977–1978). He is listed in *Who's Who in Computer Research*.

Alberto L. Sangiovanni-Vincentelli (M'74), for a photograph and biography please see page 272 of the April 1979 issue of this TRANSACTIONS.

Hsueh Y. Hsieh (M'62) received the B.S. degreee from National Taiwan University, Taiwan, China, in 1958, the M.S. degree from the Polytechnic Institute of Brooklyn, Brooklyn, NY., in 1963, and the Ph.D. degree from New York University, New York, N.Y., in 1972, all in electrical engineering.

After receiving the B.S. degree, he joined the faculty of the Electrical Engineering Department of National Taiwan University as a Teaching Assistant. From 1959 to 1961 he worked for the Raytheon Company, where he was involved in electronic circuit design. From 1961 to 1963 he worked for the Kaman Aircraft Corporation, where he was involved in the avionic system for the U.S. Navy. From 1963 to 1966 he worked for the Itek Corporation, where he was a Project Engineer responsible for developing the optical ROM (2×10^8 bits) writing system with the He Ne C. W. laser as a light source. Since 1966 he has been with the IBM System Products Division, where he worked on computer control and material, device, and circuit characterization for FET and bipolar devices. Since 1970 he has been working in the area of computation methods for circuit and system design. His publications include work on computer-aided circuit design methods, statistical techniques, and material and circuit characterization.

Dr. Hsieh is a member of Eta Kappa Nu. He was a 1973 recipient of New York University Founders Day Award.

Techniques for the Simulation of Large-Scale Integrated Circuits

A. RICHARD NEWTON, MEMBER, IEEE

Invited Paper

Abstract—**New techniques for the efficient simulation of large-scale integrated MOS circuits are described. These techniques have been implemented in the computer program SPLICE which combines circuit, timing, and logic analyzes in a single package. The use of *SOR*-Newton methods permits all three forms of analysis to be performed simultaneously, while event-control is used to enhance execution speed. The performance of SPLICE for the simulation of large NMOS circuits is also described.**

Manuscript received February 11, 1979; revised May 4, 1979. This work was supported in part by Hewlett-Packard, Corporate Engineering, Palo Alto, CA.

The author is with the Department of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California, Berkeley, CA 94720.

I. INTRODUCTION

A NUMBER of simulation techniques are available for the analysis of electronic circuits. For small circuits where analog voltage levels are critical to circuit performance, or where tightly coupled feedback loops exist, a circuit simulator such as SPICE2 [1] can accurately predict circuit performance. As the size of the circuit increases, the cost and memory requirements of such an analysis become prohibitive. Fortunately, a large fraction of a typical *LSI* system is digital in nature. For this reason, certain simplifications may be made during the