Numer Algor (2007) 45:331–343
DOI 10.1007/s11075-007-9101-7

ORIGINAL PAPER

An approach to the Gummel map by vector extrapolation methods

Roberto Bertelle · Maria Rosaria Russo

Received: 15 December 2006 / Accepted: 2 May 2007 /
Published online: 20 June 2007

Abstract The numerical approximation of nonlinear partial differential equations requires the computation of large nonlinear systems, that are typically solved by iterative schemes. At each step of the iterative process, a large and sparse linear system has to be solved, and the amount of time elapsed per step grows with the dimensions of the problem. As a consequence, the convergence rate may become very slow, requiring massive cpu-time to compute the solution. In all such cases, it is important to improve the rate of convergence of the iterative scheme. This can be achieved, for instance, by vector extrapolation methods. In this work, we apply some vector extrapolation methods to the electronic device simulation to improve the rate of convergence of the family of Gummel decoupling algorithms. Furthermore, a different approach to the topological ε-algorithm is proposed and preliminary results are presented.

Keywords Gummel map · Semiconductor device simulation · Vector extrapolation methods · ε-algorithm

1 Introduction

In the last 20 years microprocessors spread over all fields of everyday life. Nowadays each microprocessor, even if it is made by millions of few basic

R. Bertelle · M. R. Russo (✉)
Department of Pure and Applied Mathematics, University of Padova,
Via Trieste 63, Padova 35121, Italy
e-mail: mrrusso@math.unipd.it

R. Bertelle
e-mail: bertelle@math.unipd.it

components, such as MOSFET (Metal-Oxide-Semiconductor Field-Effect-Transistor), is built in some square centimeters. This is due, mainly, to the fast improvements of the physical technologies, which allows a progressive dimensions reduction of the basic components. This is according to the Moore's law, an empirical observation that the transistor density doubles every couple of years [18]. Numerical simulation plays a role, sometimes fundamental, in this fast and continuous development process.

In this paper, we deal with one step of numerical device simulation which refers to the single, basic, component. This requires the physical model of the component to be based on a system of Partial Differential Equations (PDE). The PDE system is solved via the Gummel map using a nonlinear iterative method [11]. Some acceleration techniques have been applied to the generated vector sequence [8, 10, 20]. An overview of some vector extrapolation methods is given in [5, 23], while a more recent numerical comparison is given in [14].

The paper is organized as follows. In Section 2, we recall some vector extrapolation methods belonging to two different families: the polynomial methods and the ε-algorithms; then we see how these methods could be applied to solve a linear system, and we give some numerical comparisons. In Section 3, we explain the basic concepts for the Gummel map, and propose a way to apply extrapolation algorithms in order to accelerate the corresponding rate of convergence. In the last part of this section we present some numerical experiments for two particular devices, the MOS capacitor and the p–n junction diode.

In conclusion, this work presents a new algorithm for the extrapolation of the Gummel map, based on a restarting approach. Preliminary numerical tests are shown and further investigations and theoretical results will follow.

2 Vector extrapolation methods

In this Section, we present the essential background on extrapolation methods. For a deeper understanding of applications and numerical examples, the interested reader should consult [3, 23] for more details.

Many iterative processes used in numerical analysis lead to vector sequence. If such a sequence converges too slowly, then it may be transformed, by vector extrapolation methods, into another sequence converging to the same limit, but faster than the initial one. An important property of these methods is that they do not require an explicit knowledge of the sequence generator, so they could be applied directly to the solution of linear and nonlinear systems. In particular, for nonlinear problems these methods do not need the use of the Jacobian of the function, and they have a quadratic rate of convergence, under some assumptions [13]. In this paper, we will consider four vector extrapolation methods which can be classified in two categories: the polynomial methods and the ϵ-algorithms.

2.1 Polynomial methods

Let (x_n) be a given sequence of vectors of $\mathbb{R}^N$ or $\mathbb{C}^N$ and denote its limit or antilimit by s. Using the vectors x_n, the polynomial methods produce an approximation of s as follows. If we define

$$u_i = \Delta x_i = x_{i+1} - x_i,$$
$$w_i = \Delta u_i = \Delta^2 x_i, \quad i = 0, 1, 2, \ldots,$$

and k is an integer less than N, then the approximation $s_k^{(n)}$ of s is given by solving the linear system

$$\Lambda \zeta = \alpha$$

where Λ is an $(N \times k)$ matrix, α is an N-dimensional column vector, and $\zeta = (\zeta_0, \zeta_1, \ldots, \zeta_{k-1})^T$ is the vector of the coefficients. This is an overdetermined system which has to be solved by using linear least squares. The choice of the matrix Λ and of the vector α depend on the particular extrapolation method used.

For the Minimal Polynomial Extrapolation (MPE) method [7], $\Lambda = [u_n, u_{n+1}, \cdots, u_{n+k-1}]$ and $\alpha = -u_{n+k}$. After solving the system and setting $\zeta_k = 1$, we compute

$$\gamma_j = \frac{\zeta_j}{\sum_{i=0}^{k} \zeta_i}, \quad j = 0, 1, \ldots, k$$

and the desiderated approximation of s is

$$s_k^{(n)} = \sum_{j=0}^{k} \gamma_j x_{n+j}. \tag{1}$$

For the Reduced Rank Extrapolation (RRE) method [9, 17], Λ and α are, respectively, $\Lambda = [w_n, w_{n+1}, \cdots, w_{n+k-1}]$ and $\alpha = -u_n$. The least squares approximation of the system leads to

$$s_k^{(n)} = x_n + \sum_{j=0}^{k-1} \zeta_j u_{n+j}. \tag{2}$$

From the previous formulae we could assert that a polynomial algorithm finds the approximation of s as a weighted average of $k+1$ terms, related with the given sequence, where the k independent weights are found by solving a linear system of dimension $(N \times k)$.

2.2 The ε-algorithms

Let (x_n) be a scalar sequence which converges to s. A simple recursive formula to implement the *Shanks transformation* [21], is the scalar ε-algorithm proposed by [26], defined as

$$\varepsilon_{-1}^{(n)} = 0, \qquad \varepsilon_0^{(n)} = x_n, \quad n = 0, 1, \ldots \tag{3}$$

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \left[\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}\right]^{-1} \qquad k, n = 0, 1, \ldots. \tag{4}$$

In order to generalize the scalar ε-algorithm to the vector case, Wynn [27] suggested the interpretation of the 'inverse' of a column vector z as the *Samelson inverse*

$$z^{-1} = z/\|z\|^2,$$

and the vector version of (3–4) is called the *Vector Epsilon Algorithm* (VEA). This algorithm suffers the theoretical defect that it is difficult to find its kernel, that is the set of sequences for which the exact limit is obtained. Due to this drawback, another generalization of ε-algorithm for vector sequences, the *Topological Epsilon Algorithm* (TEA), has been proposed later [4]. It can be expressed as a ratio of determinants, and computed by the following recursive algorithm:

$$\begin{aligned}
\varepsilon_{-1}^{(n)} &= 0, \quad \varepsilon_0^{(n)} = x_n, \quad n = 0, 1, \ldots \\
\varepsilon_{2k+1}^{(n)} &= \varepsilon_{2k-1}^{(n+1)} + y/\left(y, \Delta\varepsilon_{2k}^{(n)}\right) \\
\varepsilon_{2k+2}^{(n)} &= \varepsilon_{2k}^{(n+1)} + \Delta\varepsilon_{2k}^{(n)}/\left(\Delta\varepsilon_{2k+1}^{(n)}, \Delta\varepsilon_{2k}^{(n)}\right)
\end{aligned}$$

where y is a chosen fixed vector, $(\cdot, \cdot)$ is the Euclidian inner product, and Δ is the usual forward difference operator which, in our case, acts on the upper indexes.

2.3 Applications to linear systems

It will be of interest, before to enter into the main subject of this paper (the acceleration of the Gummel map), to present some practical examples of the use of extrapolation methods for solving linear systems. We consider a system $Ax = b$, and an iterative method for solving it. That is, the solution x is written as the limit of a sequence $x_{n+1} = Mx_n + q$, where M and q depend on the chosen iterative method. The starting point x_0 is always set as the zero vector.

For a faster convergence, the sequence (x_n) is accelerated using the extrapolation techniques presented in the previous section. We decide to apply the extrapolation algorithm by using the so called 'restarting technique.' First, a fixed number of successive vectors of the sequence (x_n) are collected. Then, an extrapolation step is performed and the obtained extrapolated vector is used

as a new starting point. These two steps are repeated until a stopping criteria based on the residual $r_n = b - Ax_n$ is satisfied, i.e. $||r_n||_\infty \leq \varepsilon$, where ε is some prescribed tolerance.

The effectiveness of each extrapolation technique depends on the number of collected vectors N_a used in the extrapolation step. When the sequence (x_n) is linearly generated, the theory provides an exact value for N_a. For the polynomial methods, for example MPE, the approximation of x, is obtained by (1) where the γ_j are the coefficients of the *minimal polynomial* $P(\lambda)$, of the matrix M with respect to the vector $x - x_0$, and k its degree. Furthermore, if VEA or TEA is applied to the sequence (x_n) then, thanks to the theory [3], $\varepsilon_{2k}^{(n)} = x = A^{-1}b$, where k is, in the most simple case, the degree of the characteristic polynomial of M. So, for both cases, the theoretical value of N_a is equal to k. The exact value of N_a is sometimes difficult to know and, in our examples, an approximate value is based on an experimental tuning.

The previous extrapolation methods are applied for solving two different kind of linear systems. The first is a tridiagonal system of dimension $N = 500$ with $A = \text{tridiag}([-1, 2.1, -1])$, and b is chosen so that the solution has all its components equal to 1. Similar results are obtained with a random choice of the vector b. The Gauss–Seidel iterative method is used with $\varepsilon = 10^{-8}$. The tridiagonal matrix is chosen, for, at least, two reasons. First, it allows an easy control of the rate of convergence, changing the main diagonal value. Latter, a matrix as the one used, should come out from the finite element discretization of a one-dimensional elliptic differential equation, for example the Poisson equation used in the Gummel map. Some numerical results, which refer to $N_a = 20$, are presented in Fig. 1.

The second example refers to the finite element solution of the following partial differential equation

$$\begin{cases} \Delta u = -10, & (x, y) \in \Omega = [0, 1] \times [0, 1] \\ u(x, 0) = 0 \\ u(x, 1) = 10, \end{cases}$$

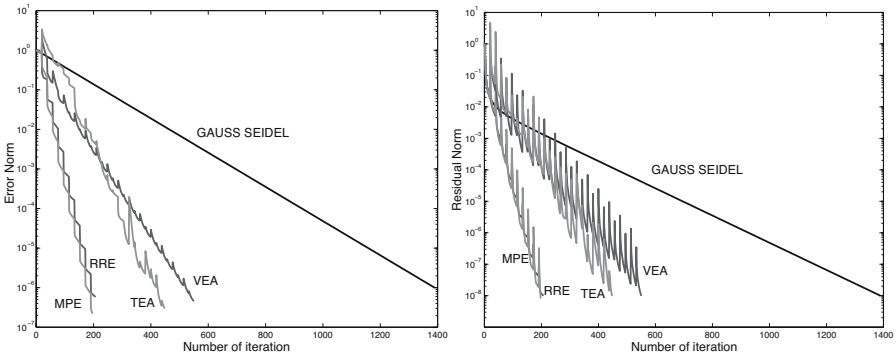

Fig. 1 Behaviour of error and residual for different acceleration methods applied to the solution of a tridiagonal linear system

and with homogeneous Neumann conditions on the rest of the boundary. The discretization gives a sparse stiffness matrix of dimension $N = 19,201$ with $133,377$ non zero elements. The Gauss–Seidel iterative method is used again with $\varepsilon = 10^{-8}$ to solve the corresponding linear system. Some numerical results, which refer to $N_a = 50$, are presented in Fig. 2. For both examples the TEA algorithm uses a new vector y at each restarting, where y is randomly generated using a normal distribution function with a zero mean value and a unitary variance. The corresponding plot is obtained using the best among several plots. Since the solution is known, we plot both the residual and the error behaviour, and we note that they are in good agreement each other. The polynomial methods perform quite well and, from our tests, which are not reported here, this still remains the case for different values of N_a. The ε-algorithms seem to be worse and also more sensitive to the choice of N_a.

As an experimental result, the performance of TEA is closely related to the choice of the auxiliary vector y. By now, little is known about the appropriate selection of y. Here, a modified version of the topological ε-algorithm of Brezinski is introduced; that is, the vector y is changed at each extrapolation step, instead of being a fixed one during the whole convergence process. The change of the vector y is done according to one of the following ways: (a) y is randomly generated once, at the beginning of the algorithm and remains the same at each restarting (y fix); (b) y is randomly generated at each restarting (y rand); (c) y is set equal to the residual at each restarting, $y = b - Ax_n$ (y res); (d) $y = x_{n+1} - x_n$ (y diff) [6]. Some results are shown in Fig. 3. As a result, TEA performs pretty well if the vector y is chosen as the residual at each restarting; also, a randomly chosen vector gives a good algorithm. With a fixed y, the algorithm could perform well but this happens with a very small probability. The results presented still exhibit this behaviour in all own tests. The work is still in progress, and theoretical results will follow.

We can assert that, to own knowledge, no known extrapolation methods are equivalent to TEA when the vector y is chosen as the preceding residual

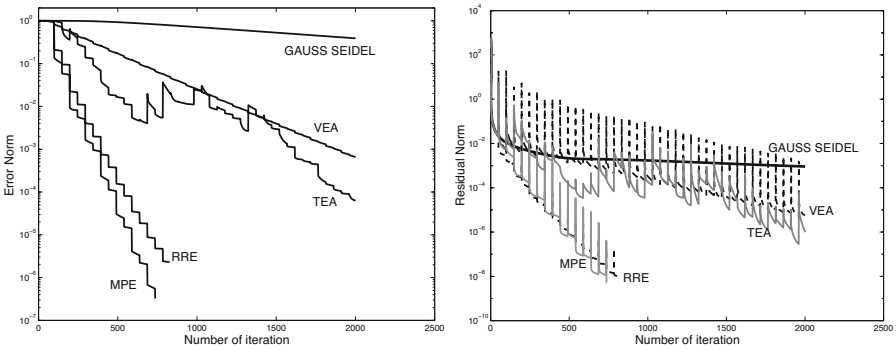

Fig. 2 Behaviour of error and residual for different acceleration methods applied to the solution of a linear system coming from the finite element discretization of a Poisson equation

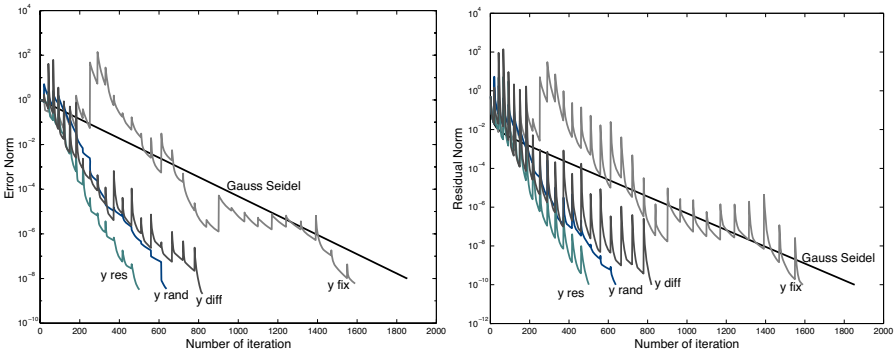

Fig. 3 Behaviour of error and residual for different choices of the vector y used in the TEA acceleration technique (with reference to the tridiagonal matrix)

at each restart. However, when the vector sequence is generated by a linear iteration, the methods MPE, RRE and TEA are mathematically related to some known Krylov methods, as first stated and proved by Sidi [22], and later by [12].

3 Extrapolation algorithms for the Gummel map

Within certain range of applications, the steady-state semiconductor simulation is based on the drift–diffusion equations, first derived by Van Roosbroeck [25], which are presented here in the scaled version due to Markowich [16]

$$\begin{cases} -\lambda^2 \triangle \psi = n_i e^{\phi_p - \psi} - n_i e^{\psi - \phi_n} + C \\ \nabla \cdot \mathbf{J}_n = R \\ \nabla \cdot \mathbf{J}_p = -R, \end{cases} \tag{5}$$

where the unknowns are the electrostatic potential ψ and the electron and hole quasi–Fermi potentials, ϕ_n and ϕ_p, respectively. The first equation of (5) is the Poisson equation; here, λ is a constant and C is the doping profile, which is a given function. The right hand side of the Poisson equation is the charge density: $n = n_i \exp(\psi - \phi_n)$ is the free electron concentration (negative charge), $p = n_i \exp(\phi_p - \psi)$ is the hole concentration (positive charge) and C is the concentration of fixed ions, which may be positive or negative. The second and third equations are the continuity equations for electrons and holes, respectively. In these equations, R is the Shockley–Read–Hall generation–recombination function, which is a nonlinear function of ψ, ϕ_n and ϕ_p

$$R = n_i \frac{e^{\phi_p - \phi_n} - 1}{\tau_p (e^{\psi - \phi_n} + 1) + \tau_n (e^{\phi_p - \psi} + 1)}$$

where τ_n and τ_p are characteristic lifetimes for holes and electrons, respectively, and n_i is the intrinsic free carrier concentration, which may be taken as a

constant for any given device temperature [24]. We do not consider Auger nor impact ionization mechanisms in this work. Defining the Slotboom variables $u = \exp(-\phi_n)$ and $v = \exp(\phi_p)$, the electron and hole current densities, $\mathbf{J}_n$ and $\mathbf{J}_p$ are, respectively

$$\mathbf{J}_n = n_i \mu_n e^{\psi} \nabla u$$

$$\mathbf{J}_p = -n_i \mu_p e^{-\psi} \nabla v, \tag{6}$$

where μ_n and μ_p are the electron and hole mobilities. Substituting (6) into (5) yields to a system of three, generally strongly coupled, equations in the variables ψ, ϕ_n and ϕ_p. A decoupling procedure may be used to numerically solve this system [11]. That is, the Poisson and the two continuity equations are solved sequentially in an iterative fashion. The Scharfetter–Gummel stabilization technique is used to avoid numerical oscillations in the solution of each current continuity equation [19]. Boundary conditions are chosen to make the device self-consistent. That is, $\mathbf{J}_n$ and $\mathbf{J}_p$ have a zero outward component in the Neumann part of the boundary. Furthermore, ψ has a value on each Dirichlet part of the frontier, whereas, in the remaining parts, an homogeneous Neumann condition is used.

The algorithm we proposed in this paper is a modified version of the classical Gummel map [11, 15], where the novelty resides in the possibilities of applying the extrapolation steps. Each variable (ψ, ϕ_n or ϕ_p) may be, on demand, separately extrapolated, but using the same kind of extrapolation method. This allows a better control on the convergence of the Gummel map. Consider, for example, ψ. During the k-th iteration, $\psi^{(k+1)}$ is stored in the vector u_{n_ψ}, where n_ψ is a counter from 1 to N_ψ (the maximum desired number of vectors to be used for extrapolation). When the number of the stored vectors reaches N_ψ, then an extrapolation step is performed by an auxiliary routine which returns the new value for $\psi^{(k+1)}$. This routine has, among other inputs, the extrapolation technique that has to be used. The extrapolated vector is then used instead of the last computed one, thus leading to a kind of restarting. The same procedure is used for ϕ_n and ϕ_p collecting N_{ϕ_n} and N_{ϕ_p} vectors, respectively, where N_{ϕ_n} and N_{ϕ_p} are fixed, positive, integers. Denoting by $||\cdot||$ the infinity norm of a vector, setting

$$\varepsilon_\psi = \frac{||\psi^{(k+1)} - \psi^{(k)}||}{||\psi^{(k)}||} \qquad \varepsilon_{\phi_n} = \frac{||\phi_n^{(k+1)} - \phi_n^{(k)}||}{||\phi_n^{(k)}||} \qquad \varepsilon_{\phi_p} = \frac{||\phi_p^{(k+1)} - \phi_p^{(k)}||}{||\phi_p^{(k)}||},$$

and given a small tolerance ε, the stopping criteria, usually adopted in devices simulations, will be

$$\max\{\varepsilon_\psi, \varepsilon_{\phi_n}, \varepsilon_{\phi_p}\} < \varepsilon. \tag{7}$$

Algorithm

Initializations

guess $\psi^{(0)}, \phi_n^{(0)}, \phi_p^{(0)}$

set $N_\psi, N_{\phi_n}, N_{\phi_p}$

$k \leftarrow 0$

$n_\psi \leftarrow 0, n_{\phi_n} \leftarrow 0, n_{\phi_p} \leftarrow 0$

repeat

1. solve $-\lambda^2 \triangle \psi^{(k+1)} = -n_i e^{\psi^{(k+1)} - \phi_n^{(k)}} + n_i e^{\phi_p^{(k)} - \psi^{(k+1)}} + C$

   if acceleration for ψ is asked

   $n_\psi \leftarrow n_\psi + 1$

   $u_{n_\psi} \leftarrow \psi^{(k+1)}$

   if $n_\psi = N_\psi$

   extrapolate z by using $u_1, \ldots, u_{N_\psi}$

   $\psi^{(k+1)} \leftarrow z$

   $n_\psi \leftarrow 1, u_{n_\psi} \leftarrow z$

   endif

   endif

   $\varepsilon_\psi \leftarrow \|\psi^{(k+1)} - \psi^{(k)}\| / \|\psi^{(k)}\|$

2. solve $\nabla \cdot [n_i \mu_n e^{\psi^{(k+1)}} \nabla u^{(k+1)}] = R(\psi^{(k+1)}, \phi_n^{(k)}, \phi_p^{(k)})$

   if acceleration for ϕ_n is asked

   $n_{\phi_n} \leftarrow n_{\phi_n} + 1$

   $v_{n_{\phi_n}} \leftarrow \phi_n^{(k+1)}$

   if $n_{\phi_n} = N_{\phi_n}$

   extrapolate z by using $v_1, \ldots, v_{N_{\phi_n}}$

   $\phi_n^{(k+1)} \leftarrow z$

   $n_{\phi_n} \leftarrow 1, v_{n_{\phi_n}} \leftarrow z$

   endif

   endif

   $\varepsilon_{\phi_n} \leftarrow \|\phi_n^{(k+1)} - \phi_n^{(k)}\| / \|\phi_n^{(k)}\|$

3. solve $\nabla \cdot [n_i \mu_p e^{-\psi^{(k+1)}} \nabla v^{(k+1)}] = R(\psi^{(k+1)}, \phi_n^{(k)}, \phi_p^{(k)})$

   if acceleration for ϕ_p is asked

   $n_{\phi_p} \leftarrow n_{\phi_p} + 1$

   $w_{n_{\phi_p}} \leftarrow \phi_p^{(k+1)}$

   if $n_{\phi_p} = N_{\phi_p}$

   extrapolate z by using $w_1, \ldots, w_{N_{\phi_p}}$

   $\phi_p^{(k+1)} \leftarrow z$

   $n_{\phi_p} \leftarrow 1, w_{n_{\phi_p}} \leftarrow z$

   endif

   endif

   $\varepsilon_{\phi_p} \leftarrow \|\phi_p^{(k+1)} - \phi_p^{(k)}\| / \|\phi_p^{(k)}\|$

4. $k = k + 1$

until $\max\{\varepsilon_\psi, \varepsilon_{\phi_n}, \varepsilon_{\phi_p}\} < \varepsilon$

3.1 Examples

This section shows some applications of the previous algorithm. The examples presented here are chosen to cover some of the more basic electronic devices, namely the *p–n junction diode* and the *MOS capacitor*. Some short comments on the electronic devices are given at the beginning of each example. For a more deeper understanding on the device physics, the interested reader is referred to [16, 24]. Let us remark some numerical considerations. The choice of the variables to accelerate is a difficult task. As an experimental result, coming out from our examples, ψ seems to be the most effective one. That is, the acceleration works well if ψ is used, whereas no, or very small, improvements are gained using ϕ_n or ϕ_p or any combination of variables involving the quasi-Fermi potentials. This question is under a theoretical investigation and is delayed to future works.

Referring to an acceleration of ψ, another not easy question is how to select, a priori, an appropriate value for N_ψ. For all our examples, this value turns out from an experimental tuning. The extrapolation is performed storing, as a first guess, a small number of vectors, say $N_\psi = 5$. If the number of Gummel iterations needed to fulfill the stopping criteria is considerable smaller than for a reference solution, well-known for these basic devices, then the current value of N_ψ is chosen, a reduction of 30% of iterations can be considered acceptable. Otherwise, N_ψ is increased, say to $N_\psi + 5$. This procedure may be time consuming, since every different choice of N_ψ needs a new run of the algorithm.

3.1.1 The MOS capacitor

A MOS capacitor is a thin layer of insulator between two, distinct, (semi–) conductor materials. One contact is a conductor sheet made on one face

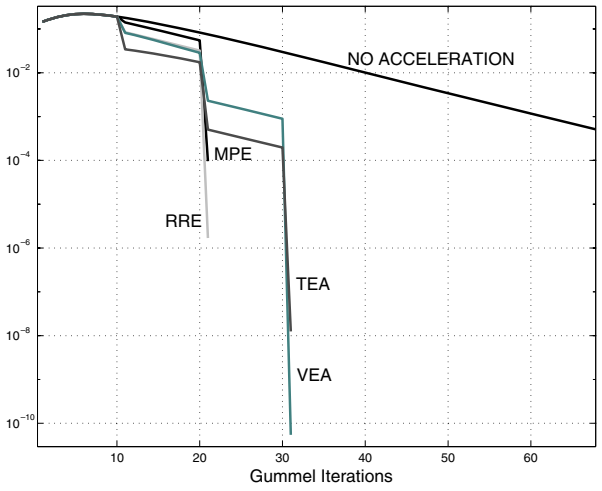

Fig. 4 A typical MOS capacitor convergence progress

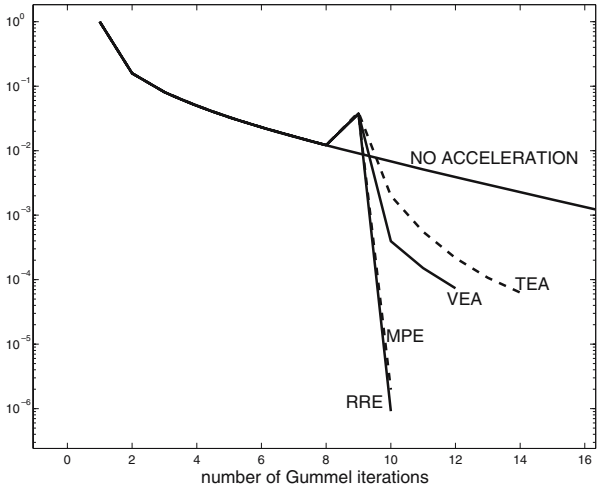

Fig. 5 Convergence progress for an abrupt p–n junction diode

of the insulator. The other contact, made on the opposite face, is a thick semiconductor layer. In this example, the insulator has a thickness $t_i = 0.1\ \mu m$ and a relative dielectric constant $\kappa_i = 3.9$. The semiconductor has a constant doping $C = -10^{22}\ m^{-3}$, a relative dielectric constant $\kappa_s = 11.7$ and is $t_s = 1\ \mu m$ thick. Two external voltages V_i and V_s are applied to the capacitor throughout the two contacts, one on the insulator and another one on the semiconductor. The total voltage applied V is defined as $V = V_i - V_s$. For a steady-state solution, the insulator insures $\mathbf{J_n} = \mathbf{J_p} = \mathbf{0}$ and $R = 0$. So, (5) reduces to the Poisson equation only. This equation is discretized using the Finite Element Method with linear elements and leads to a non-linear system of equations which is solved via the pure Newton iterative method. The stopping criteria is based on the relative norm of the difference between two consecutive iterations, i.e. the algorithm stops if $\varepsilon_\psi < \varepsilon$ where $\varepsilon = 10^{-4}$. $\psi^{(0)} = V_s$ inside the semiconductor and is linearly varying inside the insulator between V_i and V_s. The mesh has 200 points and is finely tuned to the problem to get a better solution. The first 50 points are equally spaced in the insulator domain. The second 100 points are equally spaced in the first 0.2 μm of the silicon domain. Finally, the last 50 points are equally spaced in the remaining part of the silicon domain.

A typical convergence progress is shown in Fig. 4, which refers to an applied voltage $V = 2$ Volt, the x-axis refers to the Gummel iterations, whereas the y-axis reports ε_ψ The stopping criteria is fulfilled with more than 60 iterations without any acceleration technique. Using RRE and VEA to accelerate ψ,

Table 1 Comparison of the CPU time for different extrapolation methods

	N_ψ	RRE	MPE	VEA	TEA
CPU time [s]	5	20.5	20.8	81.5	94.6
CPU time [s]	10	17.3	17.2	22.7	59.7
CPU time [s]	20	36.2	37.1	36.1	56.0

with $N_\psi = 10$, the convergence progress is greatly improved. Similar results are obtained changing the mesh size, N_ψ and V. Furthermore, RRE and TEA lead to similar results as MPE and VEA, respectively. Depending on the mesh and on the specific acceleration technique used, the CPU time using the acceleration is reduced by a factor of three to four.

3.1.2 The p–n junction diode

We consider a one-dimensional, abrupt, p–n junction diode, that is a domain $[-L, L]$ where the doping function has a negative, constant value for $x < 0$ and a positive, constant value for $x \geq 0$. Say, $C(x) = C_0 \text{sign}(x)$, where C_0 is a given positive value. Two Ohmic contacts, located at $x = \pm L$, provide an external bias $V = V(-L) - V(L)$ to the device. $V(-L)$ and $V(L)$ are, respectively, the external potentials applied to the left and right Ohmic contacts. R is neglected but similar results are obtained considering it. For this example, the non-linear Poisson equation is solved via a damped-Newton method [1], where the pure Jacobian is replaced by its lumped version. $\psi^{(0)}$, $\phi_n^{(0)}$ and $\phi_p^{(0)}$ are chosen according to [2]. A characteristic convergence behaviour is shown in Fig. 5, which refers to $L = 1\ \mu m$, $C_0 = 10^{22}\ m^{-3}$, $V = -0.5$ Volt (the p–n junction is reverse biased) and an acceleration of ψ with $N_\psi = 10$. The x-axis refers to the Gummel iterations, the y-axis reports ε_ψ. The mesh uses 50 points equally spaced in the domain. The stopping criteria refers to (7) with $\varepsilon = 10^{-4}$. This figure still preserves, for reverse bias, its qualitative behaviour against variations of V and the number of mesh points. For a direct bias the number of iterations without acceleration is about four to seven and so the acceleration tecniques are less interesting. The Table 1 shows the CPU time for three different values of N_ψ. The same simulation without any acceleration technique requires 46.7 s. From this table, it is interesting to argue the existence of a value for N_ψ which minimizes the CPU time.

4 Conclusions

From our experimental results, the extrapolation techniques seem to be successfully applicable to the acceleration of the convergence process of the Gummel map. These preliminary results are stimulating but obviously need further investigations and theoretical work, which are both in progress.

Acknowledgements We would like to thank greatly professors C. Brezinski and M. Redivo-Zaglia for their constant encouragement and suggestions. We also wish to thank professor M. Morandi Cecchi for her helpful remarks.

References

1. Bank, R.E., Rose, D.J.: Global approximate newton methods. Numer. Math. **37**, 279–295 (1981)
2. Fichtner, W., Rose, D.J., Bank, R.E.: Semiconductor device simulation. IEEE Trans. Electron Devices **30**(9), 1018–1030 (1983)

3. Brezinski, C., Redivo-Zaglia, M.: Extrapolation methods theory and practice. North-Holland, Amsterdam (1991)
4. Brezinski, C.: Généralisation de la transformation de Shanks, de la table de Padé, et de l'ϵ-algorithme. Calcolo **12**, 317–360 (1975)
5. Brezinski, C.: Convergence acceleration during the 20th century. J. Comput. Appl. Math. **122**, 1–21 (2000)
6. Brezinski, C., Redivo-Zaglia, M.: Private communication. In: 1st Dolomites Workshop on Constructive Approximation and Applications, Alba di Canazei, Trento (Italy), 8–12 September (2006)
7. Cabay, S., Jackson L.W.: A polynomial extrapolation method for finding limits and antilimits of vector sequences. SIAM J. Numer. Anal. **13**, 734–751 (1976)
8. de Falco, C.: Quantum Corrected Drift-Diffusion Models and Numerical Simulation of Nanoscale Semiconductor Devices, Ph.D. thesis, MOX, Politecnico di Milano, Italy (2005)
9. Eddy, R.P.: Extrapolation to the limit of a vector sequence. In: Wang, P.C.C. (ed.) Information Linkage Between Applied Mathematics and Industry, pp. 387–396. Academic, New York (1979)
10. Guerrieri, R., Rudan, M., Ciampolini, P., Baccarini, G.: Vectorial convergence acceleration techniques in semiconductor device analysis. Proceedings of the NASECODE IV Conference, pp. 293–298. Boole Press, Dublin (1985)
11. Gummel, H.K.: A self-consistent iterative scheme for one-dimensional steady state transistor calculations. IEEE Trans. Electron Devices **11**, 455–565 (1964)
12. Jaschke, L.: Preconditioned Arnoldi Methods for Systems of Nonlinear Equations. Ph.D. thesis, Swiss Federal Institute of Technology Zurich (2003)
13. Jbilou, K., Sadok, H.: Some results about vector extrapolation methods and related fixed-point iterations. J. Comput. Appl. Math. **36**, 385–398 (1991)
14. Jbilou, K., Sadok, H.: Vector extrapolation methods. Applications and numerical comparison. J. Comput. Appl. Math. **122**, 149–165 (2000)
15. Jerome, J.W.: Analysis of Charge Transport. Springer Verlag, New York (1996)
16. Markowich, P.A., Ringhofer, C.A., Schmeiser, C.: Semiconductor Equations. Springer-Verlag, Berlin (1990)
17. Mesina, M.: Convergence acceleration for the iterative solution of $x = Ax + f$. Comput. Methods Appl. Mech. Eng. **10**, 165–173 (1977)
18. Moore, G.: Cramming more components onto integrated circuits. Proc. IEEE **86**(1), 82–85 (1998)
19. Scharfetter, D.L., Gummel, H.K.: Large signal analysis of a silicon read diode oscillator. IEEE Trans. Electron Devices **161**, 64–77 (1969)
20. Schilders, W.H.A., Gough, P.A., Whight, K.: Extrapolation techniques for improved convergence in semiconductor device simulation. Proceedings of the NASECODE VIII Conference, Boole Press, Dublin (1992)
21. Shanks, D.: Non linear transformation of divergent and slowly convergent sequences. J. Math. Phys. **34**, 1–42 (1955)
22. Sidi, A.: Extrapolation vs. projection methods for linear system of equations. J. Comput. Appl. Math. **22**, 71–88 (1988)
23. Smith, D.A., Ford, W.F., Sidi, A.: Extrapolation methods for vector sequences. SIAM Review **29**, 199–233 (1987); Correction. SIAM Rev. **30**, 623–624 (1988)
24. Sze, S.M.: Physics of Semiconductor Devices. Wiley, New York (1981)
25. Van Roosbroeck, W.V.: Theory of flow of electrons and holes in germanium and other semiconductors. Bell Syst. Tech. J. **29**, 560–607 (1950)
26. Wynn, P.: On a device for computing the $e_m(s_n)$ transformation. MTAC **10**, 91–96 (1956)
27. Wynn, P.: Acceleration techniques for iterated vector an matrix problems. Math. Comput. **16**, 301–322 (1962)