CAUCHY-NEWTON METHODS FOR THE SOLUTION OF THE NONLINEAR EQUATIONS OF SCD NUMERICAL MODELLING

by

John S. Campbell
Colin Lyden

National Microelectronics Research Centre,
University College,
Cork, Ireland.

0. ABSTRACT

Numerical modelling of semiconductors entails the solution of a large set of highly nonlinear equations. The problem of solving this equation set may be supplanted by an equivalent unconstrained minimization which would typically use the Newton or Cauchy steepest descent vector as a search direction. Whereas the Newton approach has superlinear convergence in the neighbourhood of the solution, the Cauchy method has a slower rate of convergence. However the convergence radius of the Cauchy method is superior to the Newton method.

In the paper we consider blending methods which combine the Cauchy and Newton methods in an attempt to realise the benefits of both. Line and path Cauchy-Newton methods are presented which include, as special cases, the Model-Trust and Bank & Rose methods.

1. INTRODUCTION AND PROBLEM DEFINITION

Let $u = [u_1, u_2, u_3, \ldots\ldots u_N] \in R^N$ and consider the set of nonlinear equations $g(u) = 0$ --- (1.1)
where $g(u) = [g_1, g_2, g_3 \cdots g_N] \in R^N$

Before describing any algorithm for finding the solution vector u^* it is useful to recast the above as an equivalent unconstrained minimization problem [14]. Thus we consider a real valued function V(u), defined here as

$$V(u) = \tfrac{1}{2} g^t g \qquad \text{--- (1.2)}$$

and we note that $V'_k = \nabla V_k = J_k g_k$, in which the Jacobian matrix J_{ij} as components $J_{ij} = \partial g_i / \partial u_j$.

We make two assumptions for the object function V(u):

A(i) The function V(u) is unimodal: thus $g(u_k) = \nabla V_k = 0$ except for the single case denoted here by $u = u^*$.
A(ii) The function V(u) is convex.

Clearly, the vector $u = u^*$, for which the equality 1.1 is satisfied, also minimizes V(u) and, for the above object function,

$$V(u^*) = \min V(u) = 0 \quad ; \quad u \in R^N$$

The minimum of V(u) may be sought using the sequence of trial vectors $u_0, u_1, u_2 \ldots$ defined by

$$u_{k+1} = u_k + t_k x_k \quad ; \quad x_k \in R^N \quad ; \quad t_k \in R^1 \qquad \text{--- (1.3)}$$

The value of t_k is selected such that the successive trial vectors reduce the objective function V(u), i.e. $V(u_{k+1}) < V(u_k)$.

Two celebrated methods for minimizing a function are Cauchy's steepest descent method and Newton's method. In Cauchy's method the change vector x is directed along the negative gradient vector, that is

$$x_k = - \nabla V_k \qquad \text{--- (1.4)}$$

It is important to note that the rate of convergence of Cauchy's method is greatly dependent upon scaling and we will assume hereinafter that some beneficial scalingn scheme has been incorporated when we utilise the Cauchy vector.

In Newton's method the change vector x_k is defined as

$$x_k = - J_k^{-1} \nabla V_k \qquad \text{--- (1.5)}$$

with $t_k = 1$ for the classical-standard Newton method and $t_k < 1$ for the damped Newton case. We note that for the object function defined by equation 1.2 then $(x_k^c)^T x_k^N = 2V > 0$, which ensures that the object function V(u) decreases along the Newton direction x_k^N.

More generally we can define a change vector x as

$$x_k = -M_k^{-1} \nabla V_k \qquad \text{--- (1.6)}$$

where M_k is chosen such that $V(u_{k+1}) < V(u_k)$.

Whenever the change vector x_k has been chosen we may execute a line search in which the graph of V is scanned, along the path $p(t) = u_k + t\, x_k$.

This linear search technique seeks to find an approximate value $t = t^*$ at which V(t) takes its minimum value.

The solution algorithm of equation 1.4 therefore entails two major tasks:

T(i) Find a good search direction x_k; such that $x_k^T \nabla V_k < 0$

T(ii) Find a good estimate of $t = t^*$.

The well known Bank & Rose "t_k" algorithm of reference 7 is essentially a line search strategy in which t_k is computed as

$$t_k = (1 + K || g_k ||)^{-1} \qquad \text{--- (1.7)}$$

The method begins with small K (large t_k) and progressively increases K until $|| g_{k+1} || < || g_k ||$: thus although the method has the flavour of a line search it ends as soon as any suitable damping factor t_k has been found (whereas true linear search seeks an estimate of the best t_k value).

2. BLEND VECTOR METHODS

It is well known that, whereas the Newton method has a high superlinear convergence property in the neighbourhood of u^*, the Cauchy steepest descent method has a slower linear convergence. However the convergence radius of the Cauchy method is greater than the corresponding value for Newton's method. The Cauchy method typically makes good progress in the early stages and thereafter zig-zags slowly onwards. Conversely the Newton method is usually poor in the early iterations and extremely good later. A blending method attempts to combine the Cauchy and Newton methods to obtain the benefits of both.

An optimal blending method would ensure

(i) that divergence was avoided,

(ii) that superlinear convergence was achieved, where possible.

Bank & Rose [8] proposed a blending vector x_k^B given by

$$x_k^B = - [J_k + \alpha_k I] \nabla V = - M_k \nabla V \quad \text{--- (2.1)}$$

which is a special case of the formula proposed in reference 16

$x_k^B = - [J_k + \alpha_k D] \nabla V$, where D is a diagonal enhancement matrix.

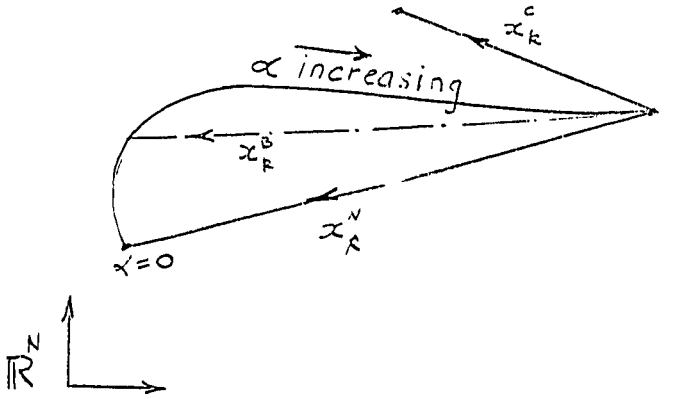

Figure 1 Blend vector $x_k^B = -[J_k + \alpha I]\nabla V_k$

The blend vector of equation 2.1 has the following properties (see figure 1).

p(i) as $\alpha_k \to 0$ $x_k^b \to x_k^N$

p(ii) as α_k increases $|| x_k^B ||$ decreases

p(iii) as $\alpha_k \to \infty$ $|| x_k^B || \to 0$ and $|| x_k^B ||$ tends to colinearity with x_k^c

p(iv) $(x_k^c)^t x_k^B > 0$; $V(u_k + x) < V(u_k)$ for some $\alpha_k > 0$.

p(v) The eigenvalues λ^M of the M matrix are simply the eigenvalues λ^J of the J matrix shifted upwards by an amount α_k [3]: thus increasing α_k improves the condition number of matrix M and as $\alpha_k \to \infty$ the condition number (M) $\to$ 1.

Concerning the economics of the method, it is important to note that for each discrete value of α_k the computation to find the blend vector x_k^B entails the solution of the equation set $M_k(\alpha_k)\ x_k^B = -\ \nabla V_k$; thus a full Gauss reduction is necessary to determine x_k^B.

An alternative strategy is to attempt to mimic the blend vector in equation 2.1 using a nonlinear combination of the Cauchy and Newton vectors x_k^c and x_k^N. Thus for example McCartin [11] used a blending

$$x_k^B = (\eta-1)^2 x_k^n + C_c\ x_k^c \qquad \text{--- (2.2)}$$

in which $\eta \in [0,1]$ and the non-negative constant C_c is prescribed. Blending function 2.2 exhibits the properties $p(i) - p(v)$, where η supplants the α_k parameter.

3. MODEL TRUST ALGORITHM

In the model trust algorithm [11,12] a blending function of the general form 2.2 is used. The sequence of trial solutions is generated as $u_{k+1} = u_k + x_k^B$; in which $x_k^B = x_k^B(\eta)$ is found by imposing the move constraint $\| x_k^B \| < \delta_k$, where δ_k is a prescribed local neighbourhood radius.

The method requires the 'a priori' prescription of the local neighbourhood radius δ_k : the quality of this quessed value is appraised by comparing the changes $\nabla V = V_{k+1} - V_k$ of the true function V with the change $\nabla Q = Q_{k+1} = Q_k$ of a quadratic funtion Q which approximates V in the neighbourhood of u_k. The neighbourhood is then expanded or contracted, based upon the outcome of the comparison. (see ref. 11 for details).

4. ALGORITHMS COMBINING BLENDING AND SEARCHING

Generation of the Newton change vector x_k^N is relatively expensive because it entails a full Gauss reduction step for the M_k matrix. Whenever the vector x_k^N has been determined it seems prudent to extract the maximum improvement (reduction) in V(u). This suggests the use of a line search or similar operation, such as in the Bank & Rose "t_k" algorithm.

On the other hand the model trust algorithm, which again entails the expense of determining the Newton vector x_k^N, does not perform any line searches.

The two algorithms presented below attempt to combine the benefits of both methods. These two algorithms are

(i) Blend-and-search algorithm,
(ii) Blend-then-search algorithm.

4.1 Blend-And-Search Algorithm

Consider a general blend vector

$$x_k^B = C_N \ (\eta-1)^2 \ x_k^N + C_c \cdot \eta \ x_k^c \quad \cdots \quad (4.1)$$

in which $\eta \in [0,1]$, $C_c > 0$ is prescribed and C_N satisfies $C_N \| x_k^N \| = \delta_k^{max}$. Here δ_k^{max} is a prescribed value representing the current maximum permissible length of the blend vector.

The central idea in this algorithm is to perform a linear search along the path $P(\eta) = u_k + x_k^B(\eta)$; thus scanning the function $V(\eta)$ to obtain the location η^* at which $V(\eta)$ is minimum (see figure 2).

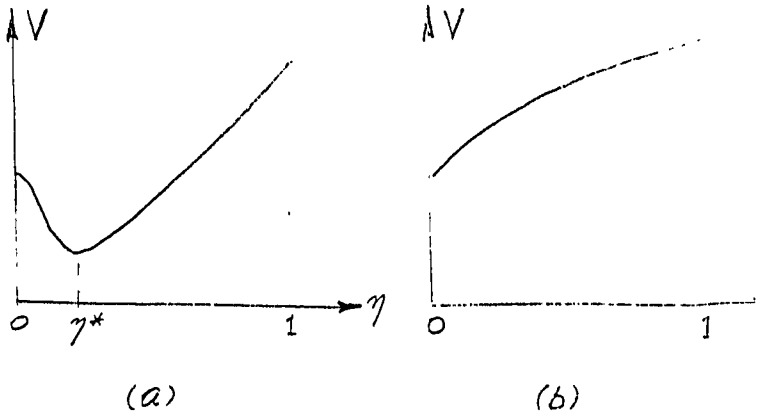

Figure 2 Line search along the path $P(\eta)$

(a) case $\eta^*_* > 0$
(b) case $\eta^* = 0$, so that $x_k^B = C_N x_k^N$

<<Blend-And-Search>>

(1) k <- o; prescribe u_o

(2) prescribe δ_k

(3) for η= 0 determine $x_k^B(0)$ and $V(u_k + x_k^B(0))$

(4) compute $V'(o) = \tau^{-1}[V(u_k+x_k^B(\tau)) - V(u_k + x_k^B(o))]$

(5) if $V'(o) > 0$ (as in figure 2b)

(6) then $\eta^* = 0$

(7) else compute x_k^c and call <<search(η^*)>> to determine η^* (as in figure 2a)

(8) u_{k+1} <- $u_k + x_k^B(\eta^*)$

(9) if converged

(10) then exit

(11) else k <- k+1; go to step 2.

4.2 Blend-Then-Search Algorithm

In this case we firstly find the blend solution $x_k^B(\eta)$ as in the model trust algorithm. Then a linear search is executed along the path $P(t)=u_k + t\, x_k^B(\eta)$

<<Blend-Then-Search>>

(1) k <- 0, prescribe u_o and δ_o

(2) compute x_k^N and x_k^c

(3) determine $\eta \in [0,1]$ such that $|| x_k^b(\eta) || < \delta_k$

(4) call <<search (t^*)>> to perform a line search along the path p(t) and hence determine an estimate t^* of the optimal t.

(5) U_{k+1} <- $U_k + t^* x_k^B(\eta)$

(6) if converged

(7) then exit

(8) else k <- k+1

(9) set to neighbourhood radius δ_k

(10) go to step 2

5. USE OF AN APPROXIMATE JACOBIAN MATRIX

The classical Newton step is often supplanted by the Newton-Richardson step $J_i.x_k^N = -\nabla V_k$, $i < k$; that is an earlier Jacobian matrix J_i is utilized over a number of iterations. The evaulation of x_k^N for the case $i = k$ entails a full Gauss reduction step whereas the case $i<k$ entails only a Gauss partial-reduction or "resolution". The relative cost of these is $\frac{\text{Cost(full reduction)}}{\text{Cost(partial reduction)}} \simeq 10^m$

where, typically, the cost index $m \in (0.5, 2.0)$. Obviously we can extend the idea to the Cauchy vector also, so that the Cauchy step becomes $x_k^c = -J_i \nabla V_k$.

6. APPRAISAL AND CLOSURE

A number of Cauchy-Newton type algorithms have been presented which include all the currently reported algorithms as special cases.

The Cauchy-Newton and Bank & Rose methods were used in the simulation of an off-state 5V reversed bias diode. The hole and electron quasi-fermi levels were assumed constant throughout the device and the behaviour was modelled by a single, highly nonlinear Poisson equation. The mesh comprised 338 nodes in 603 linear triangular finite elements. The equation solving phase, in which the Newton change vector x_k^N was determined, used an iterative, minimum-residual algorithm. In early stages, to reduce cost, a loose convergence criterion was applied yielding an approximate Newton vector. Under these circumstances the Cauchy-Newton algorithm was unconditionally convergent whereas the Bank & Rose algorithm frequently failed.

7. REFERENCES/BIBLIOGRAPHY

1. J. M. Ortega, W.G. Poole, "Introduction to numerical methods for differential equations" Pitman Publ. London, 1981 ISBN 0-273-01637-7.

2. W.F. Ames, "Numerical methods for partial differential equations", Nelson, London, 1977, ISBN 0-17-7710086-1

3. G. Strang, "Linear algebra and its applications", Academic Press, London, 1976, ISBN 0-12-673660-X.

4. R.L. Fox, "Optimization methods of engineering design". Addison-Wesley, 1971.

5. Wilde and Beightler, "Foundations of optimization", Prentice-Hall, 1967.

6. R. Fletcher, "Optimization", Academic Press, 1969.

7. R.E. Bank & D.J. Rose, "Global Approximate Newton Methods", Numer Maths, 37, 279-295, 1981.

8. R.E. Banks & D.J. Rose, "Parameter selection for Newton-like methods", SIAM J. Numer Anal, 17, 806-821, Dec 1980.

9. R.E. Banks & D.J. Rose, "Analysis of multilevel iterative methods for nonlinear equations" Mathematics of Computation, 39(160), 453-465, Oct 1982.

10. O.E. Akcusa, "Convergence properties of Newton's method", Solid State Devices, 27(4), 319-328, 1984.

11. B.J. McCartin, R.H. Hobbs, R.E. LaBarre, & P.E. Kirschner; "Solution of the discrete semiconductor device equations", Proceedings of NASECODE conference, Dublin, June 1985.

12. J.E. Dennis & R.B. Schnabel; "Numerical methods for unconstrained optimization and nonlinear equations", Prentice-Hall, 1983.

13. F. Freudenstein & B. Roth, "Numerical solution of systems of nonlinear equations", J. ACM, 10(4), 550-665, 1963

14. A.D. Booth, "Numerical methods", 3rd edition, Butterworths, London, 1966.

15. G.R. Walsh, "Methods of optimization", John Wiley, London, 1975.

16. J.S. Campbell, "The solution of the nonlinear semiconductor equations", National Microelectronics Research Centre, University College, Cork, Ireland. Technical Note NuMOS/TN003.2, Nov. 1984.