SIMULATION OF SEMICONDUCTOR DEVICES AND PROCESSES Vol. 4
Edited by W. Fichtner, D. Aemmer - Zurich (Switzerland) September 12-14, 1991 - Hartung-Gorre

New Developments in Iterative Methods for Device Simulation

Claude Pommerell and Wolfgang Fichtner
Integrated Systems Laboratory, ETH-Zentrum, 8092 Zurich, Switzerland

Abstract

We have investigated the performance of a variety of new algorithms for the solution of large systems of linear equations arising in multi-dimensional numerical device simulation. These algorithms include novel techniques for preconditioning and equation solution that are particularly suited for very ill-conditioned problems where conventional methods fail. In our software implementation, they can be combined in a modular fashion, permitting even an automatic way to solve a particular problem. As most stable algorithm, a special variant of CGS called Bi-CGSTAB, together with a new ILU drop tolerance preconditioner, shows particular promise in handling even the worst-case problems.

1 Introduction

For complex multi-dimensional simulation problems, the solution of the linear system of equations lies at the heart of the overall computational burden. The choice of an optimal solution algorithm will largely depend on the properties of the matrix corresponding to these systems and the size of the problem itself. In most interesting device simulation problems, these matrices are nonsymmetric with huge condition numbers. Linear equations solvers based on a variant of Gaussian Elimination, possibly augmented with some kind of pivoting, have enjoyed enormous popularity for these situations. Once the problem size leads to linear systems of 10k or more unknowns, however, memory considerations preclude the utilization of direct schemes.

In this paper, we present information on several new algorithms together with some representative results on the implementation and performance of a software package developed specifically for the iterative solution of badly conditioned systems arising in device simulations. We have tested conventional Conjugate Gradients methods as well as all significant extensions to nonsymmetric systems, including recent developments such as Bi-CGSTAB [1] or QMR [2]. Our tests include a whole set of preconditioners with differing storage and timing costs, suited for moderately to strongly ill-conditioned systems.

2 A new iterative method: Bi-CGSTAB

The linear systems that arise in semiconductor device simulation are known to be very ill-conditioned. Many of the iterative methods that have been proposed to solve unsymmetric systems fail on these problems [3, 4]. Even GMRES [5], a method that many authors recommend as the best iterative solver for unsymmetric systems, cannot solve most of our linear systems to the accuracy needed for the outer nonlinear solver. The full curve in Figure 1 shows the convergence history of GMRES(10) for a moderately ill-conditioned problem. The residual norm drops only very slowly after fast initial convergence. In this case, unrestarted GMRES would reach higher accuracies in reasonable time, but at unreasonable storage costs.

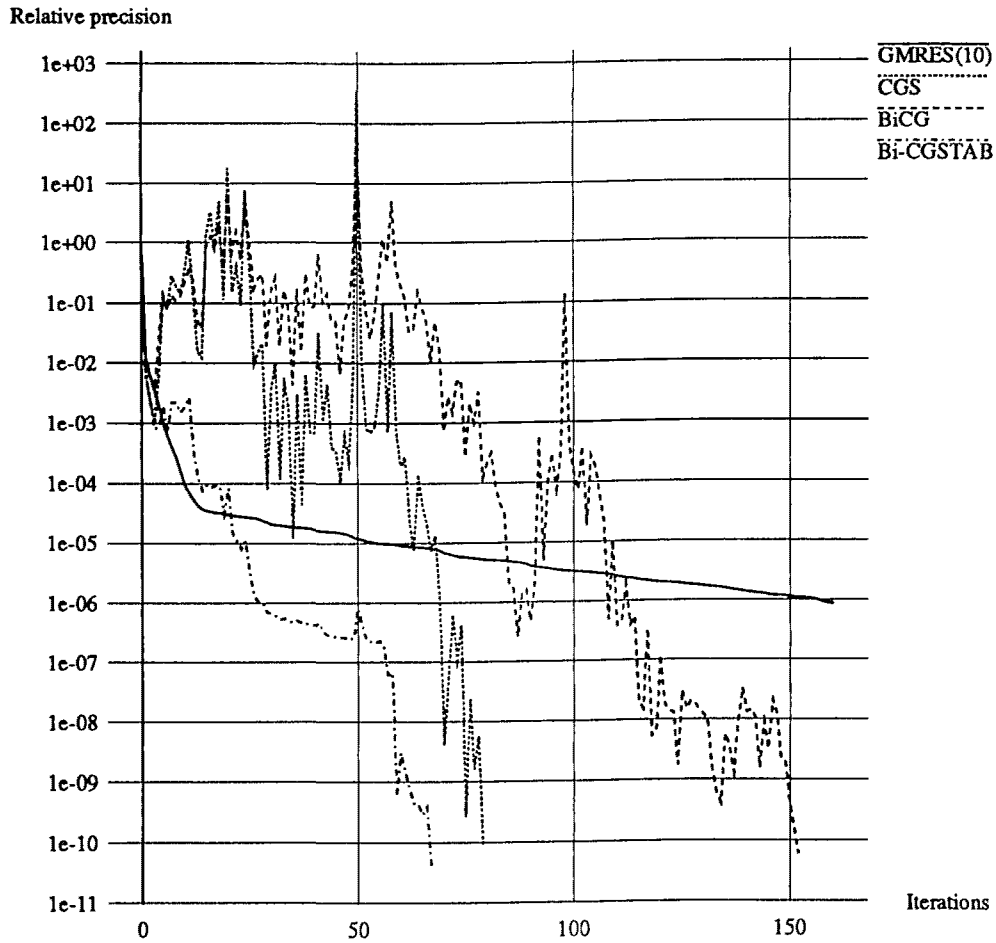

Figure 1: *Evolution of the residual norm in the iterative solution of a typical, moderately ill-conditioned linear system. The linear system has 46,692 unknowns and occurs inside the coupled nonlinear solver in the simulation of a trench DRAM cell on an irregular 3-D grid. This figure shows the behavior of the new Bi-CGSTAB method as well as the well-known methods GMRES(10), BiCG, and CGS, all with right D-ILU preconditioning.*

GMRES has the desirable property that it improves the residual at every single iteration. The rate of convergence, however, can be very slow, and sometimes gets stuck even completely. Other methods, like CGNR, Orthomin [6], or row projection methods [7], showed a similar behavior. The only methods that do succeed in solving our linear systems come from the family of biorthogonalization methods [8, 9], which includes Biconjugate Gradients [10] and its variants.

The *BiCG* method has no global minimization property. This means that the residual does not converge in every single iteration; some iterations can increase the norm of the iteration. The algorithm can still converge to a small residual and thus to a good approximation of the solution if there both some converging and some diverging iterations. In that case, the plot of the convergence history has local peaks. We call this the *zigzagging* effect in BiCG. The dashed curve in Figure 1 shows the zigzagging effect for BiCG.

The basic idea behind the *CGS* method [11] is to square the matrix polynomials used inside BiCG. This works out well when BiCG converges: CGS is sometimes almost twice as fast. The pitfall of the squaring shows up in iterations where BiCG diverges: CGS usually diverges up to twice as much in the same iteration. The dotted curve in Figure 1 shows the convergence history of CGS for the same example. We can see clearly how the convergence and divergence in BiCG is amplified by CGS in every single iteration. CGS still usually converges faster to the requested precision even in the presence of diverging iterations.

Bi-CGSTAB is a new method that has been proposed recently by van der Vorst [1]. Bi-CGSTAB is based on the same matrix polynomials as CGS, but instead of being squared, this polynomial is premultiplied by another polynomial. This second polynomial damps the effect of divergence in the BiCG polynomial. For convergent BiCG steps, the second polynomial increases convergence effect in a similar way to the squaring in CGS.

The dash-dotted curve in Figure 1 shows the convergence history of Bi-CGSTAB on our test case. There are still small peaks left which are due to divergence in the BiCG process, but the method shows a much smoother convergence behavior.

3 A new preconditioner: numerical dropping

Even with the best iterative schemes, convergence can be excruciatingly slow due to ill-conditioned matrices. This fact can be improved through proper preconditioning.

A preconditioner Q is an approximation to the system matrix A that is easily invertible, i.e. applying the operator Q^{-1} should be relatively cheap in terms of CPU and memory cost. Instead of the original linear system $Ax = b$, the *(right) preconditioned system* $[AQ^{-1}]\,[Qx] = b$ is solved. The preconditioned matrix AQ^{-1} should have a better condition (with respect to the iterative algorithm) than the original matrix A (e.g., a better eigenvalue spectrum).

The most common preconditioner for unsymmetric linear systems is *incomplete LU-factorization without fill (ILU)*. The preconditioner is constructed as $Q = (I + L)(D + U)$, where the strict lower triangular matrix L and the strict upper triangular matrix U have the same sparsity structure as the original matrix A. L, D, and U are computed by a normal LU-factorization procedure, but off-diagonal nonzero entries l_{ij} or u_{ij} in the factors occurring at positions where the corresponding entry a_{ij} in A is zero are ignored and discarded immediately.

The *diagonal ILU (D-ILU)* preconditioner can be written as $Q = (D + L_A)D^{-1}(D + U_A)$, where L_A and U_A are the strict lower and strict upper triangles of A, respectively. D is computed by an incomplete LU-factorization procedure as above, but here even updates to off-diagonal entries at nonzero positions in A are ignored. Although the resulting condition may not be as good as for ILU, this variant has the advantage that the application of the preconditioner

can be implemented much faster when combined with a matrix-vector product (the so-called Eisenstat trick [12]).

Although ILU is widely used, it is far from optimal for "bad" systems, especially in the coupled equation solution. *Numerical dropping (ND(τ))* is a new preconditioning method that is also based on *LU*-factorizations. Each nonzero entry in the factors (whether there has been a nonzero at the position in the matrix A or not) is compared to its row or column maximum. If the entry is smaller in magnitude than the maximum times a tolerance τ, it is dropped and not considered anymore in the factorization process.

Figure 2 shows timings for the solution of a very ill-conditioned linear system on a CONVEX C-220. The linear system occurs in the fully coupled simulation of an MOS-controlled thyristor (MCT) [13]. It has 209,874 unknowns, and the matrix contains 4,086,348 nonzero entries. Several hundreds of similar systems have to be solved for a transient simulation of this MCT, and the entire simulation takes about one week on the C-220.

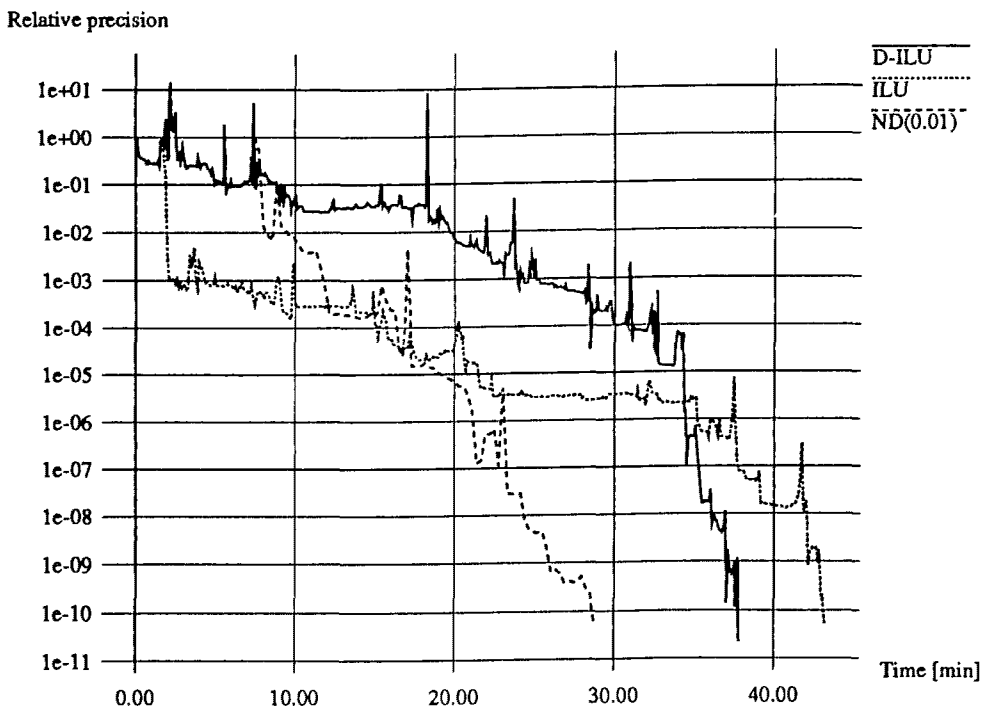

Figure 2: *Evolution of the residual norm for the solution of a large and very ill-conditioned linear system, using right-preconditioned Bi-CGSTAB with different preconditioners. The X-axis shows the time in minutes on a CONVEX C-220.*

The full line shows the convergence history of right D-ILU preconditioned Bi-CGSTAB. Note that the X-axis shows minutes of execution time. Setting up the D-ILU preconditioner comes almost for free, this is why the full line starts just after the Y-axis. One iteration with D-ILU takes about 4.4 seconds.

The dotted line shows right ILU preconditioned Bi-CGSTAB. The setup overhead for the ILU preconditioner takes about 100 seconds (so the dotted line starts further right on the plot). The first four iterations give ILU here an advantage over D-ILU. However, D-ILU finally catches

up for high accuracies, as ILU requires more time (6 seconds) per iteration.

The setup for the LU-factorization with numerical dropping is rather complicated and takes a lot of time. In above case, the setup for a drop tolerance of 0.01 takes seven and a half CPU minutes. The factors that make up the preconditioners only have 32% more nonzero entries than the original matrix (and as the ILU factors). One Bi-CGSTAB iteration takes 16.6 seconds with this preconditioner. The dashed line in Figure 2 shows how quickly ND(0.01) recovers from its late start due to the long setup time.

The remarkable thing in Figure 2 is that neither the setup part nor the application of the numerical dropping preconditioner run in vector mode. The inherent parallelism would be just to small to obtain any vectorization speedup. The D-ILU and ILU preconditioners profit fully from the vector capabilities of the CONVEX C-220. Nevertheless the (ND(τ))-preconditioned algorithm converges faster.

Numerical dropping did not fail to solve any of our very ill-conditioned systems up to today, and it always did it in a reasonable amount of time. With the degree of parallelism that current supercomputers can exploit, it is even competitive for moderately ill-conditioned systems, despite of its inherent lack of parallelism in the current formulation. A more parallel implementation might be possible albeit at much higher storage costs.

During the last few months, we have carefully tested our new iterative solution schemes inside several software packages for two- and three-dimensional numerical device simulation. Through a hierarchical solution approach (i.e. a sequence of increasingly robust algorithms and preconditioners)) we have completely eliminated the convergence failures experienced before.

References

[1] H. A. van der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of CG-S for the solution of nonsymmetric linear systems." submitted to SIAM J. Sci. Stat. Comput.

[2] R. W. Freund and N. M. Nachtigal, "QMR: a quasi-minimal residual method for non-Hermitian linear systems," Tech. Rep. 90.51, RIACS, NASA Ames Research Center, Dec. 1990.

[3] R. E. Bank, W. M. Coughran, Jr., M. A. Driscoll, R. K. Smith, and W. Fichtner, "Iterative methods in semiconductor device simulation," *Computer Physics Communications*, vol. 53, pp. 201–212, 1989.

[4] G. Heiser, C. Pommerell, J. Weis, and W. Fichtner, "Three dimensional numerical semiconductor device simulation: Algorithms, architectures, results," *IEEE Trans. Computer-Aided Design*, to appear 1991.

[5] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, pp. 856–869, July 1986.

[6] P. K. W. Vinsome, "ORTHOMIN – an iterative method for solving sparse sets of simultaneous linear equations," in *Proc. Fourth SPE Symposium on Reservoir Simulation*, (Los Angeles), pp. 149–160, Society of Petroleum Engineers, Feb. 1976. Paper SPE 5739.

[7] R. Bramley and A. Sameh, "Row projection methods for large nonsymmetric linear systems," CSRD Rpt. 957, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Jan. 1990.

[8] M. H. Gutknecht, "A completed theory of the unsymmetric Lanczos process and related algorithms Part I," IPS Research Report 90-10, Interdisciplinary Project Center for Supercomputing, ETH Zürich, June 1990. submitted to SIAM J. Matrix Anal. Appl.

[9] M. H. Gutknecht, "A completed theory of the unsymmetric Lanczos process and related algorithms Part II," IPS Research Report 90-16, Interdisciplinary Project Center for Supercomputing, ETH Zürich, Sept. 1990. submitted to SIAM J. Matrix Anal. Appl.

[10] R. Fletcher, "Conjugate gradient methods for indefinite systems," in *Proc. of the Dundee Biennal Conference on Numerical Analysis* (G. A. Watson, ed.), (New York), Springer-Verlag, 1975.

[11] P. Sonneveld, "CGS, a fast Lanczos-type solver for nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 10, pp. 36–52, Jan. 1989.

[12] S. C. Eisenstat, "Efficient implementation of a class of preconditioned conjugate gradient methods," *SIAM J. Sci. Stat. Comput.*, vol. 2, pp. 1–4, Mar. 1981.

[13] B. J. Baliga, "Evolution of MOS-bipolar power semiconductor technology," *Proc. IEEE*, vol. 76, pp. 409–418, 1988.