A Band Relaxation Algorithm for Reliable and Parallelizable Circuit Simulation

A. Lumsdaine J. White D. Webber[1] A. Sangiovanni-Vincentelli[1]

Research Laboratory of Electronics
Dept. of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

ABSTRACT

A variable-band relaxation algorithm for solving large linear systems is developed as an alternative to Gauss-Jacobi relaxation. This algorithm seeks to improve the reliability of Gauss-Jacobi relaxation by extracting a variable-sized band from the matrix and solving that band directly. This leads to a relaxation algorithm with provably better convergence properties. Furthermore, this algorithm can be used effectively on a massively parallel computer, because band matrices can be solved in $log(n)$ time on $\frac{n}{2}$ processors. Test results are presented which compare the convergence properties of variable-band relaxation to Gauss-Jacobi relaxation.

1 Introduction

Designers of high performance integrated circuits make extensive use of circuit simulation programs like SPICE and ASTAP [NAG, WEE] in order to tune their designs before fabrication. These circuit simulation programs often require hours or days to complete a single simulation because they use computationally expensive, but very reliable, numerical techniques. Since many simulations are performed to design a given integrated circuit, slow simulator turn-around time can significantly increase overall design time. For this reason, using parallel processors to reduce the execution times of circuit simulation programs has been the focus of much current research[COX, JAC].

Programs like SPICE and ASTAP use implicit multistep integration algorithms to convert the differential equation system to a sequence of algebraic problems, one for each integration timestep. The algebraic problems are solved using an iterative Newton method, each step of which involves linearizing the circuit about some guessed solution, and solving the generated sparse linear system. Good parallel speed increases have been achieved for the linearized system construction, but not for the sparse linear system solution, particularly if there are many (more than 128) processors.

For machines with many processors, a technique that is effective for MOS circuits is to use Gauss-Jacobi relaxation(GJ). Since the GJ algorithm solves a system of equations by repeatedly solving each equation independently for its associated unknown and then passing around the computed values, GJ easily exploits as many processors as there are equations to be solved. Although guaranteed to converge under certain conditions, GJ can be inefficient if the timestep required to achieve convergence is very small. Also, GJ can be unreliable because in some cases convergence may be indicated when the result is far from the correct solution. An approach to improving the reliability of GJ while maintaining a very parallel algorithm is to extract a variable-sized diagonal band of the matrix and to solve that band directly. This leads to a relaxation method with provably better convergence properties. Band relaxation is still very parallelizable because band matrices can be solved in order $log(n)$ steps, given order n processors where n is the number of equations in the system.

In Section 2, we develop the variable-band relaxation algorithm as a generalization of GJ. A method for sorting a matrix to improve the performance of the banded-relaxation is presented in Section 3, and we discuss how to incorporate the variable-band algorithm into a circuit simulation program. In Section 4, GJ and the band relaxation algorithm are used in a circuit simulator and the results from the transient analysis of several different circuits are compared. Finally, we present our conclusions and some suggestions for future research.

2 Variable-Band Relaxation

As mentioned above, the iterative Newton method is used to solve the nonlinear algebraic problem associated with each timestep of the transient analysis of a circuit, and each iteration of the Newton method involves computing the solution to a sparse linear system. Specifically, for each Newton iteration an x must be found such that $Ax = b$, where $A \in \Re^{n \times n}$ is the Jacobian of the nonlinear system, $x \in \Re^n$ is usually the vector of node voltage updates, and $b \in \Re^n$ is usually the vector of sums of currents entering each node. In most circuit simulators the matrix problem is solved with some form of sparse Gaussian elimination, which is difficult to parallelize[WIN].

More easily parallelized methods for solving the sparse matrix problem are the iterative relaxation algorithms, the simpliest of which is the well-known GJ relaxation. In GJ, the solution to a system of equations is computed by solving each equation independently for its associated unknown, and then exchanging the computed values, repeating the process until a consistent solution throughout is achieved. The element update equation for GJ can be written compactly as

$$x_i^{k+1} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_j^k \right], \qquad (1)$$

[1]Dept. of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

where k is the iteration index and a_{ij} is the ij^{th} entry of the matrix A.

As the equations represented by the rows in A are solved independently, GJ can easily exploit n processors effectively. In addition, GJ is guaranteed to converge when applied to solving the matrices generated by the transient analysis of MOS circuits, if the timestep is small enough and there is a capacitor to ground at each node. The difficulty with GJ is that it is inefficient if the timestep required to achieve convergence is very small, and GJ is unreliable because false convergence is common due to the limited spread of information with each iteration[DUE,SAL,WEB,WHI].

To see how to construct more reliable relaxation methods with nearly as much parallelism, it is helpful to cast relaxation into a more general frame. Any two matrices $M, N \in \Re^{n \times n}$ are said to be a splitting of the matrix A if $A = M - N$. The iteration equation for a relaxation algorithm based on a given splitting is then

$$Mx^{k+1} = Nx^k + b, \tag{2}$$

and the asymptotic rate of convergence is the spectral radius of $(M^{-1}N)$[VAR]. The goal is to select a splitting in which M is easy to factor and the convergence of (2) is fast. Furthermore, since the objective is to produce an efficient parallel algorithm, M should be easy to factor in a parallel fashion.

For the case of GJ, M is a diagonal matrix whose elements are the diagonals of A and is already in factored form. This leads to a very efficient parallel algorithm, but one which has poor convergence properties. One would expect that movement of off-diagonal elements of A from N to M would accelerate the convergence, because more of the system is being solved directly. This conjecture can be proven for the case of an A which is diagonally dominant with positive diagonals and negative off-diagonals[VAR]. If M is selected to be a band matrix with a band-size much less than n, then by using parallel cyclic reduction or nested dissection[DUF], M can be factored in order $log(n)$ steps on a parallel processor with order n processors[DON]. The relaxation algorithm generated by using a banded M will be referred to as "band relaxation".

3 Equation Ordering and Band-Size Selection

If a band relaxation scheme is used, the ordering of the equations in A will determine which off-diagonal elements are in the band, so it is possible to change the convergence properties of the relaxation by reordering A. We desire a good heuristic algorithm that will order A so as to reliably improve the convergence properties of the relaxation, while at the same time allowing the use of an M with as small a band-size as possible. Because of the physical significance of the matrix A, one would expect that placing the matrix entries corresponding to tightly-coupled circuit nodes into M would improve the convergence of the band relaxation.

The matrix M is derived by first constructing a graph based on A which has edges between pairs of nodes only if the pair of nodes is "tightly coupled." The so-constructed graph of the matrix A is reordered to minimize the bandwidth of M using the Reverse-Cuthill-McKee (RCM) algorithm[DUF,GEO]. This produces an ordering for A which will insure that the tightly coupled equations lie together in an easily extracted band, i.e. in M. Note that the band-size for M is automatically produced in the ordering process.

A suggested heuristic[WHI] for finding tightly coupled pairs of nodes i, j is to examine the spectral radius, ρ, of the GJ relaxation iteration matrix applied to the to a 2×2 problem generated by deleting all but the i^{th} and j^{th} rows and columns of A. In particular,

$$\rho = \sqrt{\left|\frac{a_{ij}a_{ji}}{a_{ii}a_{jj}}\right|}. \tag{3}$$

Nodes i and j are considered to be tightly-coupled if ρ is close to or greater than unity.

However, consideration of only the spectral radius is not a good method for improving the rate of convergence of GJ. For example, consider a lower-triangular A with large subdiagonal entries. The spectral radius of $(M^{-1}N)$ is zero, but order n GJ iterations will be needed to achieve convergence. The transient analysis of MOS circuits can produce a highly non-symmetric A, and the coupling test implied by (3) may miss nodes which are tightly-coupled in only one direction. Requiring that nodes i and j be grouped together if

$$\frac{\max^2(a_{ji}, a_{ij})}{|a_{ii}a_{jj}|} \geq \alpha \tag{4}$$

will ensure that strong one-way couplings are included into M.

The last step in developing the variable-band relaxation algorithm is selecting a value for α. For a given circuit, selecting a fixed value for α is a somewhat arbitrary process. While it might seem like a desirable goal to be able to select an α which will produce a desired rate of convergence, it is not clear in large systems how the choice of α will affect the convergence rate. However, one way to "tune" the value of α to a particular circuit is to adaptively adjust the value of α and monitor the effect on the band-size of M. Since we are generally solving large, very sparse systems, if an ordering existed such that $M = A$ (e.g. a tridiagonal A), the band-size of M would still be small compared to n. This suggests that maximum efficiency can be obtained if we require that the band-size of M be at least $\lceil \frac{m}{n} \rceil$, where m is the total number of entries in A and $\lceil \frac{m}{n} \rceil$ is the smallest integer greater than or equal to $\frac{m}{n}$. To accomplish this, on the initial ordering of A, an α should be selected which produces a band-size of M equal to $\lceil \frac{m}{n} \rceil$. Thereafter, and for all subsequent reorderings, α can be temporarily decreased from its initial value in order to include the maximum number of elements into M without increasing the initial band-size.

The following are the adaptive sorting and the band relaxation algorithms (expressed in pseudo-code):

Algorithm 1 (AS) Given $A, M \in \Re^{n \times n}$, A having m elements, and M being the band to be extracted from A, this algorithm orders A so as to put as many tightly-coupled elements as possible onto a band with size of at least $\lceil \frac{m}{n} \rceil$.

```
MRCM(A, α)
While band-size(M) < ⌈m/n⌉
    Shrink(α)
    MRCM(A, α)
η := α
β := band-size(M)
While band-size(M) = β
    Shrink(η)
    MRCM(A, η)
```

The function MRCM(A, α) sorts A with the modified Reverse-Cuthill-McKee algorithm mentioned above, using a coupling criterion of α.

Algorithm 2 (BR) Given $x, b \in \Re^n$ and $A, M \in \Re^{n \times n}$, M being the band of tightly coupled elements contained in A, this algorithm solves $Ax = b$ iteratively for x.

band-LUdecomp(M)
$x^0 := 0$
For $k = 1, 2, ...$
  For $i = 1, 2, ..., N$
    sum := 0
    For $j = 1, 2, ..., N$
      If $j < i - p$ or $j > i + p$
        sum := sum + $a_{ij} x_j^k$
    $b'_i := b_i -$ sum
  band-solve $Mx^{k+1} = b'$
  If x^{k+1} converged with x^k, break
  If $k >$ max-iter, signal failure

The routines band-LUdecomp and band-solve perform the obvious functions, but are optimized for banded systems.

Since the circuits being simulated are nonlinear, the values of the elements A will change during the course of the simulation, possibly worsening the convergence characteristics of BR. While it may be possible to improve the convergence of A by taking smaller time steps, it has been empirically observed that it is best to reorder whenever BR fails to converge. Note that in this dynamic reordering process, the band-size of M is likely to change. Combining AS with BR gives our final result, the variable-band relaxation algorithm:

Algorithm 3 (VBR) Given a circuit simulation program using an iterative Newton method to solve the nonlinear algebraic problem associated with each timestep of the transient analysis of a circuit, where each iteration of the Newton method involves computing the solution to a sparse linear system, $Ax = b$, $A \in \Re^{n \times n}$ being the Jacobian of the nonlinear system, this algorithm provides a method of incorporating BR into the simulation program.

- Sort A once according to AS at the beginning of the simulation.
- Use BR to solve the linear system $Ax = b$ generated at each Newton iteration.
- If the BR fails, reorder A according to AS and signal Newton non-convergence.

Note that VBR can also be used in a nonlinear relaxation algorithm for the Newton method solution. Nonlinear relaxation uses the same iterative solution technique as the standard Newton method, but instead of solving the linear system exactly at each time step, one relaxation iteration is performed. Since VBR has better convergence properties than GJ in the linear case, one would expect that it would also accelerate nonlinear relaxation.

4 Test Results

The algorithms above were coded in C and incorporated into a circuit simulation program for a serial computer. The following circuits were selected as test examples:

- *dac*: DAC circuit, $n = 149$, $\lfloor \frac{m}{n} \rfloor = 5$.
- *lin*: Linear RC line, $n = 601$, $\lfloor \frac{m}{n} \rfloor = 3$.
- *opamp*: Opamp circuit, $n = 52$, $\lfloor \frac{m}{n} \rfloor = 7$.
- *pla*: PLA circuit, $n = 66$, $\lfloor \frac{m}{n} \rfloor = 5$.
- *rcc1*: High-speed CMOS static RAM control circuit, row access simulation, $n = 149$, $\lfloor \frac{m}{n} \rfloor = 5$.
- *rcc2*: Identical to *rcc1*, but with parasitic resistances at every MOSFET drain and source, $n = 703$, $\lfloor \frac{m}{n} \rfloor = 5$.
- *shmem*: Shared memory read circuit, $n = 759$, $\lfloor \frac{m}{n} \rfloor = 5$.

Note that *rcc2* is an especially difficult circuit for a relaxation-based simulator, due to the large number of parasitic elements. The transient analysis conducted on each circuit example was the same regardless of the linear solution method. Relaxation failure was indicated if the relaxation method failed to converge within 32 iterations.

The following tables of results demonstrate the effectiveness of the VBR algorithm. The number of Newton iterations required for direct, GJ, and VBR solution methods, as well as the number of relaxation iterations required for GJ and VBR are shown in Table 1. Table 2 shows the number of relaxation failures for GJ and VBR, $\lfloor \frac{m}{n} \rfloor$, and the maximum band-size of M used during VBR. Note that there is very little difference in the number of Newton iterations between VBR and direct methods, but that there is a significant difference in the number of Newton and relaxation iterations between GJ and VBR. As a relaxation method, VBR combines the parallelism of GJ with the efficiency and reliability of direct methods.

	Newton Iterations			Relaxation Iterations	
Circuit	Direct	GJ	VBR	GJ	VBR
dac	2013	1999	1978	10871	7856
lin	148	8364[1]	148	128005[1]	230
opamp	347	765	397	11077	1620
pla	737	2423	729	31558	3448
rcc1	3223	3266	3233	21143	13247
rcc2	4837	36455[1]	5068	157598[1]	29260
shmem	624	812	655	11205	2317

Table 1: Number of Newton Iterations for Direct, GJ, and VBR Solution Methods; Number of Relaxation Iterations for GJ and VBR Methods.

	Relaxation Failures			
Circuit	GJ	VBR	$\lfloor \frac{m}{n} \rfloor$	Max Band-Size
dac	6	1	5	13
lin	2000[1]	0	3	3
opamp	143	1	7	13
pla	341	0	5	9
rcc1	14	0	5	5
rcc2	2000[1]	4	5	11
shmem	89	0	5	13

Table 2: Number of Relaxation Failures for GJ and VBR, Value of $\lfloor \frac{m}{n} \rfloor$, and Maximum Band-Size Used During VBR.

[1] Simulation terminated before completion due to excessive number of relaxation failures (2000).

Finally, the VBR algorithm was used in conjunction with a nonlinear relaxation scheme, in which only one relaxation iteration was made for each Newton iteration. Table 3 shows the total number of Newton iterations required for GJ and VBR. Note that VBR shows a dramatic improvement in most of the circuits, a factor of almost 100 for *pla*. In fact, the nonlinear VBR method uses no more Newton iterations than the direct method, but in a parallel implementation each iteration would be much cheaper.

Circuit	Direct	GJ	VBR
dac	2013	3522	2813
lin	148	179890[1]	148
opamp	347	6958	818
pla	737	119791	1319
rcc1	3223	9223	4264
rcc2	4837	[2]	20726
shmem	624	19497	1082

Table 3: Number of Newton Iterations for Direct Solution Method and Number of Nonlinear Relaxation Iterations for GJ and VBR Methods.

One may ask if it is possible to find a static ordering *a priori* which can be used throughout the transient simulation. From our experimental results, the answer is no. One logical candidate for a good static ordering would be the final ordering used in a transient simulation. We tested this hypothesis with *rcc2*, since it was the only circuit tested which required more than one reordering. The result was that the VBR had to reorder one more time than in the first case; it had to *undo* the initial ordering it was given, even though this was the ordering it used later on in the simulation. This is not to say that a static ordering does not exist, only that it does not seem easy to find using the methods described in this paper.

An obvious extension to variable-band relaxation is to use a banded pre-conditioner with conjugate-gradient methods. Some preliminary experiments were conducted along these lines, but the results were not encouraging, and are not included here.

5 Conclusion

In this paper, a variable-band relaxation algorithm for solving large linear systems was developed as an alternative to Gauss-Jacobi relaxation. This algorithm improved the reliability of GJ, while preserving the easily exploitable parallelism, by extracting a variable-sized band from the matrix and solving that band directly. Test results were presented which compared the convergence of variable-band relaxation to Gauss-Jacobi relaxation.

There are some extensions to the work presented in this paper that may be worth exploring. For instance, the heuristic used for grouping nodes together on the band could be made more sophisticated; something similar to the conductance partitioning idea in [WHI] might work well. An implementation of the VBR algorithm is currently being designed for the Connection Machine, but implementations for other architectures should be pursued as well.

[1] Simulation terminated before completion due to excessive number of relaxation failures (2000).

[2] Simulation terminated before completion due to Newton non-convergence.

ACKNOWLEDGEMENTS

This work was supported by the Defense Advanced Research Projects Agency contract N00014-87-K-825. The authors would like to thank the professors and students in the Custom Integrated Circuits Group at MIT and the personnel at Thinking Machines, especially Lennart Johnsson.

References

[COX] P. Cox, R. Burch, B Epler, "Circuit Partitioning for Parallel Processing," *IEEE Int. Conf. on Computer-Aided Design*, pp. 186-189, Nov. 1986.

[DEU] J. T. Deutsch, A. R. Newton, "MSPLICE: A Multiprocessor-Based Circuit Simulator," *Int. Conf. Parallel Processing*, pp. 207-214, May, 1984.

[DON] J. Dongarra and S. Lennart Johnsson. "Solving banded systems on a parallel Processor," *Parallel Computing*, 5(1&2):219-246, 1987.

[DUF] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, 1986.

[GEO] A. George and J. W-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, 1981.

[JAC] G. Jacobs, D. Pederson, "An Empirical Analysis of the Performance of a Multiprocessor-based Circuit Simulator," *Proc. of the Design Automation Conference*, Las Vegas, Nevada, June 1986.

[NAG] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," Electronics Research Lab Report, ERL M520, Univ. of Calif., Berkeley, May 1975.

[SAL] R. Saleh and A. R. Newton, "An Event-Driven Relaxation-Based Multirate Integration Scheme for Circuit Simulation," *Proc. Int. Symp. on Circuits and Systems*, Philadelphia, Pennsylvannia, May 1987.

[VAR] R. Varga, *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs, New Jersey, 1962.

[WEB] D. M. Webber, A. Sangiovanni-Vincentelli, "Circuit Simulation on the Connection Machine," *24th ACM/IEEE Design Automation Conf.*, pp. 108-113, June 1987.

[WEE] W. T. Weeks, A. J. Jimenez, G. W. Mahoney, D. Mehta, H. Quasemzadeh, T. R. Scott, "Algorithms for ASTAP - A Network Analysis Program," *IEEE Trans. on Circuit Theory*, pp. 628-634, Nov. 1973.

[WHI] J. K. White, A. Sangiovanni-Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits*, Kluwer Pub., Boston, 1986.

[WIN] O. Wing, J. W. Huang, "A Computation Model of Parallel Solution of Linear Equations," *IEEE Trans. on Computers*, pp. 632-638, July 1980.