A Semi-Coarsening Multigrid Algorithm for SIMD Machines

J. E. Dendy, Jr.
M. P. Ida
J. M. Rutledge

CRPC-TR91122
April, 1991

Center for Research on Parallel Computation
Rice University
P.O. Box 1892
Houston, TX 77251-1892

A SEMI-COARSENING MULTIGRID ALGORITHM FOR SIMD MACHINES*

J. E. Dendy, Jr.†
M. P. Ida‡
J. M. Rutledge§

Abstract. A semi-coarsening multigrid algorithm suitable for use on SIMD architectures has been implemented on the CM-2. The method performs well for strongly anisotropic problems and for problems with coefficients jumping by orders of magnitude across internal interfaces. The parallel efficiency of this method is analyzed, and its actual performance is compared with its performance on some other machines, both parallel and nonparallel.

1. **Introduction.** Some previous papers have examined multigrid methods for their suitability for calculation on machines with SIMD architectures. Frederickson and McBryan [FM] developed and analyzed a method which was designed to keep all the processors busy. Decker analyzed [D1] the performance on SIMD machines of more traditional multigrid methods. Both of these papers were restricted to Poisson's equation with periodic boundary conditions; neither paper attempted to address the sort of problem we are interested in, namely

$$-\nabla \cdot (D(x,y)\nabla U(x,y)) + \sigma(x,y)U(x,y) = F(x,y) \tag{1.1}$$

in a bounded region Ω of R^2, where $D = (D^1, D^2)$, D^i is positive, $i = 1, 2$, and D^i, σ, and F are allowed to be discontinuous across internal boundaries Γ of Ω; moreover, $D_1 \gg D_2$ and $D_1 \ll D_2$ in different subregions of Ω is possible.

Another method was advocated by Hackbusch [H] and was shown to be robust for constant coefficient, periodic, anisotropic problems. It can be argued that this method — or at least its precursor — may be found in [T]. Maintaining our political neutrality, we will refer to this method as the Brandt-Hackbusch-Ta'assan method if only to arrive at the t.l.a. (three letter acronym) B.H.T. The B.H.T. method, like the Frederickson-McBryan algorithm, has the property of preserving the busyness of the processors, and has the added advantage of only needing point relaxation for anisotropic problems. However, as shown in Section 4, the B.H.T. method, as described by Hackbusch, will not handle problems like (1.1), and it is unclear how to give the method this capability.

* Received by the editors and in revised form . The work of the first two authors was performed under the auspices of the U.S. Department of Energy under contract W-7405-ENG-36 and was partially supported by the Center for Research on Parallel Computation through NSF Cooperative Agreement No. CCR-8809615.

† Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545.

‡ Cal Tech, Pasadena, CA 91125

§ Chevron Oil Field Research Co., P. O. Box 446, La Habra, CA 90631

What about the more traditional methods of dealing with (1.1)? The first multigrid method to handle such problems successfully was given in [ABDP] and expanded in [D2]; it used standard coarsening (discussed below), interpolation induced by the operator, Galerkin coarsening, and alternating red-black line relaxation. An alternative was first discussed in [DMRRS] for three-dimensional problems. (However, we must point out that the robustness of line relaxation coupled with semi-coarsening for constant coefficient anisotropic problems was realized in [W].) The method discussed in this paper is the two-dimensional analogue of the method in [DMRRS]; it uses semi-coarsening in y, interpolation induced by the operator, Galerkin coarsening, and red-black line relaxation by lines in x. Additionally, the method in this paper uses a technique due to Schaffer [S]; without this technique, the semi-coarsening method would not be competitive. The method discussed in this paper is in large part the same as the method given in [SW]. This fact should not be too surprising since both papers had their genesis in a code written by the first author of this paper.

One potential liability of the method considered in this paper is the necessity to perform line relaxation. The BHT method, were it robust, would get around this difficulty. The suggestion was made in [B] that anisotropies could be avoided by the use of local grid refinement, under the assumption that physical problems are isotropic and that anisotropies arise from nonuniform gridding. One way to avoid nonuniform gridding is to use local grid refinement. In [D3], it was shown how to generalize [D2] to the case of local grid refinement. However, many person years have been invested in codes which do not use local grid refinement, and all these codes would have to be rewritten to use this approach. Moreover, it is not clear how local grid refinement algorithms will perform on SIMD machines. Finally, there are important physical problems which are strongly anisotropic; an example is petroleum reservoir engineering [L]. For these problems, local grid refinement, although it may be desirable for other reasons, will not lead to isotropic problems on the local grids; thus, it appears that the issue of anisotropic problems must be directly attacked, not avoided.

Yet another method due to Mulder [M] seems to have possibilities. The idea is that each grid has two offspring, one obtained by semi-coarsening in x, the other by semi-coarsening in y. When two offspring are of the same size in x and y, they are declared to be the same offspring. Although this method requires only point relaxation, the number of auxiliary grids proliferates to the point that it appears that this method would require as much work as the method we propose below, with considerably more complexity; this point is discussed further in Section 2

2. **Standard Versus Semi-Coarsening.** In [ABDP, D2] standard coarsening was used; that is, given a Cartesian grid, the coarser grid is obtained by deleting the even x- and y-lines. In this paper semi-coarsening is used; that is, the coarser grid is obtained by deleting the even y-lines. To handle general anisotropic situations with standard coarsening seems to require alternating line relaxation, whereas with semi-coarsening only line relaxation by lines in x is required. (We comment that there are some situations in both cases that cannot be handled by these choices of relaxation [ABDP], but for our purposes these cases are pathological.) What is the sequential relaxation work for the two meth-

ods? Given an $nx \times ny$ grid, the relaxation work for semi-coarsening is of the order of $2(nx)(ny)(1 + 1/4 + \cdots) = (8/3)(nx)(ny)$. For semi-coarsening, the relaxation work is of the order of $(nx)(ny)(1 + 1/2 + \cdots) = 2(nx)(ny)$.

To do this same counting argument for a SIMD architecture requires a short discussion of the assumptions. In the simplest model for the CM-2, it is important that the arrays be "compatible" that is, of the same size; otherwise great inefficiencies in communication result. Thus given a fine and coarser grid, it is assumed that the data for each grid are stored in arrays which are compatible. Thus every other row of the coarse grid matrix contains no useful information. Moreover, on a relaxation sweep, a mask is employed which makes the processors for these rows idle. The assumption therefore is that for every grid, the amount of work to solve the collection of tridiagonal systems on that grid depends only on the number of $x-$ or $y-$points on the finest grid. Thus if $nx = ny$, the relaxation work in the standard coarsening case is $2W(nx)log_2ny$, and the relaxation work in the semi-coarsening case is $W(nx)log_2ny$, where W(nx) is the work to solve a $nx \times nx$ tridiagonal system. (There are log_2ny grids, and on each grid the relaxation work is $2W(nx)$ or $W(nx)$, respectively.) If sparse Gaussian elimination (a.k.a. the Thomas algorithm) is used, $W(nx) = O(nx)$. If straightforward cyclic reduction is used $W(nx) = O(log_2nx)$. Whatever the method, the relaxation work is halved with the semi-coarsening approach. It was assumed at the outset of this work that the relaxation work would be the bottleneck for multigrid on the CM-2, and this assumption led to semi-coarsening as the algorithm of choice. The results in section 4 show that the relaxation work indeed dominates the computational time.

We give a brief discussion of how the interpolation operators are derived, even though the description in [SW] is excellent. Let us denote the interpolation operator from the coarse grid G^{k-1} to the fine grid G^k by I^k_{k-1}. (The coarse grid operator L^{k-1} is given by Galerkin coarsening from the fine grid operator L^k by forming $(I^k_{k-1})^* L^k (I^k_{k-1})$. We are interested in five point or nine point discretizations of (1.1); hence, we want the coarse grid operators also to be five or nine point operators.) In [ABDP, D1, D3], I^k_{k-1} was described as follows: At coarse grid points coinciding with fine grid points, I^k_{k-1} is just the identity. At a fine grid point lying vertically between two coarse grid points, let the template of the operator be given by

$$\begin{pmatrix} NW & N & NE \\ W & C & E \\ SW & S & SE \end{pmatrix}. \tag{2.1}$$

Then I^k_{k-1} at $v_{i,j}$ is given by $av_{i,j-1} + bv_{i,j+1}$, where

$$a = -(SW + S + SE)/(W + C + E) \quad \text{and} \quad b = -(NW + N + NE)/(W + C + E).$$

That is, one thinks of summing away the x-dependence to obtain a three point relation. A problem with this approach, when using standard coarsening, is that if $\rho = C - NW - N - NE - W - E - SW - S - SE$ is small, then instead of using $W + C + E$ in (2.1), one should use $SW + S + SE + NW + N + NE$ instead; this point is discussed in [D3]. With standard coarsening, results are relatively insensitive to switching between formulas based on the

size of ρ; however, for semi-coarsening such is not the case. With standard coarsening, there is also interpolation being performed in the $x-$direction as well; hence, there is a possibility of "sins" being averaged out in that direction. What is needed apparently in the semi-coarsening case is some mechanism for averaging in the $x-$direction. Schaffer [S] also came to this conclusion and discovered the following scheme: Let

$$A^- v^- + A^0 v^0 + A^+ v^+ = 0$$

be the equation that would give the row $v^0 = (v_{i,j}, i = 1, \cdots, nx)$ in terms of the rows $v^- = (v_{i,j-1}, i = 1, \cdots, nx)$ and $v^+ = (v_{i,j+1}, i = 1, \cdots, nx)$. Then

$$v^0 = -(A^0)^{-1}(A^- v^- + A^+ v^+). \tag{2.2}$$

Unfortunately, use of (2.2) would lead to a nonsparse interpolation, leading to nonsparse coarse grid operators. Schaffer's idea is to assume that $-(A^0)^{-1}A^-$ and $(-A^0)^{-1}A^+$ can each be approximated by diagonal matrices in the sense that B^- and B^+ are diagonal matrices such that

$$-(A^0)^{-1}A^- e = B^- e \text{ and } (-A^0)^{-1}A^+ e = B^+ e,$$

where e is the vector $(1, \cdots, 1)$. To find B^- and B^+ requires just two tridiagonal solves.

It is worth commenting at this point that the SIMD relaxation work of Mulder's algorithm [M] appears to be comparable to the SIMD relaxation work of our method. The reason for this is that the number of grids in Mulder's method, if $nx = ny$, is approximately $(log_2 ny)^2$, which is directly proportional to the number of red-black point relaxations performed per V-cycle, since there does not appear to be any easy way to perform relaxation simultaneously on all grids at a given level, as in the Frederickson-McBryan or the B.H.T. method. The interpolation and residual weighting work is also proportional to this factor, whereas in our algorithm, this work is proportional to $log_2 ny$. The Mulder method is, however, under investigation by Van Rosendale and Naik (to be identified with the author of [D1]). In [NR], it is shown that the subgrids in the Mulder method can be efficiently organized for relaxation on all the grids simultaneously or organized for efficient communication between levels, but it is so far unclear how to achieve both these objectives together.

3. **Implementation.** Up to this point we have made no assumptions about the number of grid points residing on a processor. When the number of grid points exceeds one per physical processor, the CM-2 makes use of virtual processors (VP), the virtual processor ratio (VP ratio) being defined as the number of grid points per physical processor. In effect, serial do loops (VP loops) are created on each processor. This fact has two important implications for our algorithm. The first implication is that the multigrid coarsening aspect of our algorithm has to be carefully coded to avoid inefficiency. Let us denote the i-index as the tridiagonal solver direction and the j-index as the multigrid coarsening direction. If the j-index is declared as parallel, then the code will compile so that the the

VP loop on a physical processor always remains the same size, regardless of the coarse grid size. Thus, if the j-index is declared such that the VP loop size is 64, it will remain so, instead of decreasing to 32, 16, etc., thus reflecting the inactive j-direction grid elements on coarser grids. One way to get around this difficulty is to hand code the VP loop by splitting the j-index into parallel and serial parts; this kind of splitting was done for the CM-2 implementation (by Gyan Bhanot of Thinking Machines) of Jameson's FLOW 67 code, a time-stepping multigrid code. We have not yet done this kind of splitting for our entire multigrid code, since we hope to avoid it when a new utility is made available that allows access to the same array with different layouts. We have coded the relaxation routine this way (which currently takes over 90% of the computational time) and timing results are presented in Section 3.

The second implication of the VP ratio is that a hybrid algorithm [J] becomes the one of choice for solving the tridiagonal systems; this algorithm performs the traditional Thomas algorithm sequentially on the serial loop part of the i-index on each physical processor and uses cyclic reduction to solve between physical processors. This algorithm is also used in [SW]. It is ironic that the same algorithm is the efficient one for these different, MIMD and SIMD, architectures. Moreover, for the higher VP ratios, line relaxation will perform with nearly the same efficiency as point relaxation, suggesting that there is no compelling reason to investigate the alternatives in Section 1 that use point relaxation.

There are at least two versions of the Thomas algorithm. One computes the LU-decomposition on the tridiagonal matrix as it is needed. Other versions save the LU-decomposition (one such version was exploited in [ABDP] to avoid expensive divides on the CDC-7600). There is an analogous situation with respect to cyclic reduction. We have found that a version which saves the LU-decomposition runs three times faster on the CM-2 than a version which recomputes the LU-decomposition [J]. However, for cyclic reduction, the LU-decomposition must be stored at each level of the parallel reduction. The result, for the i-index, is that the requirement for storage will be proportional to $nx(log_2 nx)$, where nx is the number of i-grid points. However, for the hybrid version [J] the storage requirement of the LU-decomposition is just proportional to nx, assuming that one does not save the 2-cyclic LU-decomposition needed for the processor boundary grid points. (For high VP ratios, this assumption is reasonable since the cyclic reduction part of the tridiagonal solves is a small fraction of the overall computational time.)

4. **Results.** We present timing comparisons between several machines in Table 1. The timing results are given as seconds per V-cycle and were obtained by running five V-cycles (including set up time) and dividing by five. Thus the timing results are independent of the difficulty of the problem run. (Some convergence factor data are presented in [SW] for problems of varying degrees of difficulty.) All of the CM-2 timings reported in this section were done using 16K 1-bit processors using the "slicewise" Connection Machine Fortran compiler. Because this compiler treats each Weitek floating point unit as a physical processor (and there are 32 1-bit processors with each Weitek), we will say that the benchmarks of this section were established on a 512 Weitek processor machine. The front end for the CM-2 was a Sun 4/90. The iPSC/2 machine had 64 nodes of 386 type processors; several configurations of these processors were considered for each problem size;

here we have reported the timings only for the best [SW]. The timing results on the Cray Y-MP were obtained in a time-sharing environment; with a dedicated Y-MP, we could have easily run problems larger than 2048 × 2048; however, if we had used a 2048 Weitek processor CM-2 we could have also run problems larger than 4096 × 4096. The results on the Y-MP show that great gains are easily made from vectorization for the smaller problems, but for the larger problems, the asymptote of time being linearly proportional to problem size has nearly been reached.

TABLE 1:TIME PER V-CYCLE ON THREE MACHINES

Size of Problem	iPSC/2	CRAY-YMP	CM-2 recompute LU	CM-2 save LU	CM-2 split axis: serial/parallel
32 × 32	.3	.01	–	––	––
64 × 64	.7	.04	.66	.36	.77
128 × 128	2.0	.09	2.12	.88	1.25
256 × 256	––	.27	8.57	3.22	2.45
512 × 512	––	.95	38.2	14.38	5.25
1024 × 1024	––	3.69	17	++	12.82
2048 × 2048	––	––	++	++	36.81

++ too large
–– no information

The first two columns of the CM-2 timings were made with a version that had the arrays declared entirely parallel; one can see the log base two nonlinear increase with problem size, in that each time the size is quadrupled, the required time is slightly more than quadrupled. The last column gives the time required for the relaxation routine only for the implementation with arrays being split into serial and parallel parts as discussed in Section 2.5. This calculation was not done by saving the LU-decomposition for the tridiagonal factors, and thus could be further improved. By doubling the number of relaxation sweeps at each grid level for the complete multigrid implementation, we found that the relaxation routine took 96% of the computational time, and thus argue that these relaxation timings are representative of the entire code with the split axis implementation. Note that for the last column, the asymptotic rate has not been reached and between the last two entries, the computational time is only slightly less than tripled. We also report in Table 2 the effect of changing the size of the coarsest grid direct solve. The direct solve is done with a band solver on the front end with the LU-decomposition of that matrix being precomputed once. The problem in Table 2 has 256 × 256 grid points. Note that there is a minimum for both the case of saving and not saving the cyclic reduction LU-decomposition. The reason for this is that on the coarser grids, so few Weitek processors are active that the front end (which is considerably faster than one Weitek processor) is more efficient, even when the time to transfer the data from the CM-2 to the front end is taken into account.

TABLE 2:TIME PER V-CYCLE VARYING COARSEST GRID SIZE

Size of coarsest grid	Number of grid levels	Recompute LU	Save LU
256×1	9	8.57	3.22
256×4	7	6.53	2.66
256×8	6	5.87	2.69
256×16	5	5.98	3.50
256×32	4	8.41	6.62

Let us now consider the BHT method, first for the problem

$$\begin{cases} -\Delta U + U = 1 \text{ in } (0,1) \times (0,1), \\ U \text{ doubly periodic.} \end{cases}$$

On a 16×16 grid we compare the result of using the method of [D4] (standard coarsening, operator induced interpolation, Galerkin coarsening, and red-black point relaxation) to the method of [H]. (The reason that the method in [D4] is used is that we have not yet extended the method in this paper to handle periodic boundary conditions.) Both methods achieve an average convergence factor, over ten V-cycles, of .05 per V-cycle.

Now let us consider the problem

$$\begin{cases} -\nabla \cdot (D \nabla U) + U = F \text{ on } (0,16.) \times (0.,16.) \\ U \text{ doubly periodic,} \end{cases} \tag{4.1}$$

where D and F are as shown in figure 1. Using a 16×16 grid, we again compare [D4] to [H]; the average convergence factor per V-cycle is .06 vs. .54, respectively. If we modify the method in [H] to attempt to use the same operator induced interpolation used in [D4], we obtain an average convergence factor of .34 instead. The problem is that within the context of the method in [H], it is no longer clear what operator induced interpolation should be.

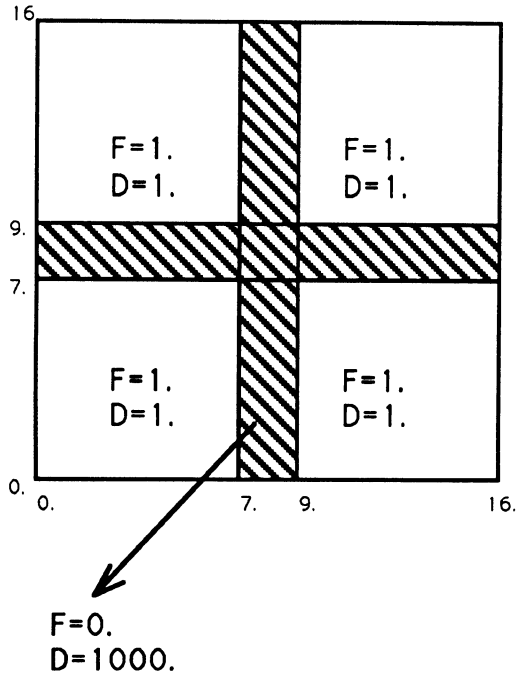

Figure 1: Diffusion coefficients and right hand side for (4.1)

Finally let us comment that we believe that some multigrid method is likely to be the fastest algorithm for solving problems like (1.1) on SIMD machines. We intend to substantiate this belief by timing the algorithm of this paper against some possible competitors like preconditioned conjugate gradient methods. In this comparison, the convergence factor per unit time must be considered for various problems, since clearly conjugate gradient with no conditioning will win against multigrid on SIMD machines for well-conditioned elliptic problems. In one sense the contest is over before it starts, since we already have an example in which a preconditioned conjugate gradient method stagnates badly, but for which the method of this paper is robust. Nevertheless, we hope to perform a comparison for a set of problems with a wide range of difficulty.

5. **Conclusions.** In this paper we have examined several multigrid methods in an attempt to find one that performs well on SIMD machines for problems with rough and anisotropic coefficients. We chose a semi-coarsening multigrid algorithm for implementation on the CM-2 and have shown that it does perform well on that machine. We expect even better performance from this algorithm when a utility is made available that will allow us to access the same array with different layouts. Such a utility will also make the coding of this algorithm much easier and result in more readable code.

REFERENCES

[B] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp. 31(1977), pp. 333-390.

[ABDP] R. E. Alcouffe, A. Brandt J. E. Dendy, Jr., and J. W. Painter, *The multi-grid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Stat. Comp. 2(1981), pp. 430-454.
[D1] N. Decker, *On the parallel efficiency of the Frederickson-McBryan multigrid*, SIAM J. Sci. Stat. Comp. 12(1991), pp. 208-220.
[D2] J. E. Dendy, Jr., *Black box multigrid*, J. Comp. Phys. 48(1982), pp. 366-386.
[D3] J. E. Dendy, Jr., *A priori local grid refinement in the multigrid method*, Elliptic Problems Solvers II, G. Birkhoff, A. Schoenstadt, eds., Academic Press, New York, 1984, pp. 439-452.
[D4] J. E. Dendy, Jr. *Black box multigrid for periodic and singular problems*, Appl. Math. Comp. 25(1988), pp. 1-10.
[DMRRS] J. E. Dendy, Jr., S. F. McCormick, J. W. Ruge, T. F. Russell, S. Schaffer, *Multigrid methods for three-dimensional petroleum reservoir simulation*, Proceedings of the Tenth Symposium on Reservoir Simulation, Houston, TX, Feb. 6-8, 1989, pp. 19-25.
[FM] P. O. Frederickson, O. A. McBryan, *Normalized convergence rates for the PSMG method*, SIAM J. Sci. Stat. Comp., 12(1981), pp. 221-229.
[H] W. Hackbusch, *The frequency decomposition multigrid method, part I: application to anisotropic equations*, Numer. Math. 56(1989), pp. 229-245.
[J] S. L. Johnsson, *Solving tridiagonal systems on ensemble architectures*, SIAM J. Sci. Stat. Comp. 8(1987), pp. 354-392.
[L] L. W. Lake, *The Origins of Anisotropy*, Journal of Petroleum Technology, April, 1988, pp. 395-396.
[M] W. A. Mulder, *A new multigrid approach to convection problems*, J. Comp. Phys., 83(1989), pp. 303-329.
[NR] N. Naik, J. Van Rosendale, *Robust Parallel Multigrid Using Semi-coarse Grids*, presented at Fifth Copper Mountain Multigrid Conference, March 31-April 5, 1991, Copper Mountain, Colorado, to appear.
[S] S. Schaffer, private communication, paper in preparation.
[SW] R. A. Smith, A. Weiser, *Semicoarsening multigrid on a hypercube*, Submitted to SIAM J. Sci. Stat. Comp.
[T] S. Ta'assan, *Multigrid methods for highly oscillatory problems*, Ph. D. Thesis, Weizmann Institute of Science, 1984.
[W] G. Winter, *Fourienanalyse zur Konstruktion schneller MGR-Verfahren*, Ph. D. Thesis, Rheinischen Friedrich-Wilhelms-Universität zu Bonn, 1982.