NASA Contractor Report 181999
ICASE Report No. 90–17

ICASE

ON THE PARALLEL EFFICIENCY OF THE FREDERICKSON-McBRYAN MULTIGRID

Naomi H. Decker

Contract No. NAS1–18605
February 1990

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23665–5225

Operated by the Universities Space Research Association

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

(NASA-CR-181999) ON THE PARALLEL EFFICIENCY OF THE FREDERICKSON-MCBRYAN MULTIGRID Final Report (ICASE) 11 p CSCL 12A

N90-19791

Unclas
G3/64 0269652

On the parallel efficiency of the Frederickson-McBryan Multigrid Algorithm

Naomi H. Decker *

Abstract

To take full advantage of the parallelism in a standard multigrid algorithm requires as many processors as points. However, since coarse grids contain fewer points, most processors are idle during the coarse grid iterations. Frederickson and McBryan claim that retaining all points on all grid levels (using all processors) can lead to a 'superconvergent' algorithm. Has the 'parallel superconvergent' multigrid algorithm, PSMG, of Frederickson and McBryan solved the problem of implementing multigrid on a massively parallel SIMD architecture? How much can be gained by retaining all points on all grid levels, keeping all processors busy?

The purpose of this work is to show that the PSMG algorithm, though it achieves perfect processor utilization, is no more efficient than a parallel implementation of standard multigrid methods. PSMG is simply a new and perhaps simpler way of achieving the same results.

*Research supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-18605 while in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23665-5225.

1. Introduction

The parallel multigrid algorithm of Frederickson and McBryan[3] is frequently mentioned as an efficient method for implementing multigrid on a fine-grained SIMD architecture, specifically the Connection Machine. At first glance, their use of multiple coarse grids (using the same number of grid points on all grid levels) looks very promising since

1. the processor utilization rate is high
2. the convergence rate is very good, at least for simple model problems
3. it eliminates the aliasing between modes, so that a lower order restriction may be used

How much more efficient is this parallel 'superconvergent' multigrid algorithm (PSMG) than the standard sequential algorithms implemented efficiently in parallel? In this note, we compare the efficiency of PSMG to that of the standard red-black Gauss-Seidel multigrid algorithm. This standard algorithm is unattractive for massively parallel architectures, since most of the processors will be idle on the coarsest grids. That is, though an equal amount of time is spent on all grid levels, there is not enough work on the coarse levels to saturate the architecture. The hope of PSMG is that there is *useful* work to be done by otherwise idle processors on the coarse levels. However, a careful calculation of computation and communication costs shows that, for the Poisson equation model problem, for which the PSMG algorithm was originally devised, the standard RB (red-black) algorithm and the original version of PSMG have virtually identical efficiencies. Although the convergence rate of PSMG is very good, the relaxation employed there is relatively expensive in both communication and computation. Modifications of the original PSMG, using less expensive relaxations, have been proposed, see [1], but it is shown in [2] that improvement over the standard methods is limited - most of the processors are doing useless work on the coarse grids.

2. Assumptions

In order to compare parallel superconvergent multigrid (PSMG) to a standard multigrid cycle implemented in parallel, it is clearly insufficient to give processor utilizations or megaflop rates. We must have a measure of processor utilization which reflects the amount of *useful* work being done.

For direct methods one can estimate the total number of computation and communication steps required to solve the problem completely. Similarly, for the FMG algorithm (full multigrid algorithm), one can count the total number of steps required to solve the problem to within truncation (discretization) error. When using an iterative technique, like a V-cycle multigrid algorithm, which does not solve to truncation error, the efficiency of the algorithm must be expressed as convergence rate per iteration, combined with an operation count per iteration. This information generally suffices for comparing similar multigrid algorithms, since the error reduction tends to be fairly uniform throughout the iteration process. (Unlike, for example, conjugate gradient algorithms, where error reduction tends to occur in occasional sudden jumps). Thus for the kinds of multigrid being considered here we need to measure the error reduction per iteration, and might also like an estimate of the number of iterations required to bring convergence error below truncation error.

The discretization of the model problem determines the numerical stencils used, and hence the data dependencies in the parallel loops. On parallel architectures, small changes in the discretization can have a major impact on the behavior of both the numerical method. We begin by assuming a five point discretization of the two dimensional Laplacian, but we make no claims that this is the best possible.

The comparison analysis also depends on the model of computation. We wish to compare standard multigrid to PSMG on its home turf, a massively parallel machine in which there are at least as many processors as there are grid points (unknowns) in the discrete equations. If there were fewer processors than grid points, then the efficiency of PSMG would be much poorer than standard red-black algorithms.

As well as being dependent on the machine architecture and the data dependencies dictated by the underlying p.d.e., the computation and communication counts are very sensitive to algorithmic details. There are the obvious choices to be made about interpolation and restriction operators and the order of the red-black sweeps. There is also the usual trade-off between the computation cost and the communication cost. In the case of PSMG, the computation and communication appear to be minimized by the same algorithm. In presenting the standard methods, where there are a number of simple variations on the basic method, we consider only those algorithms which are relatively efficient and can be easily compared to the PSMG algorithm. Although there are ways to further optimize the standard multigrid algorithm, either reducing communication at the expense of computation or vice versa, we have sacrificed a little efficiency in the interest of simplicity and practicality.

2.1 Machine Assumptions

For computing the parallel computation and communication we have made the following assumptions:

1. each processor can fetch only one value at a time

2. each processor can do only one add/multiply at a time

3. at any given time, every processor must either execute the same instruction as all other processors or do nothing

4. each processor can use locally stored constants, which may differ from processor to processor, to be used in the computational steps

5. there is no overlap of computation and communication

We also assume that the cross data communication traffic from the simultaneous relaxations on the multiple coarse grids doesn't degrade the overall performance of PSMG relative to the standard RB algorithms.

2.2 Work measures

Using the above assumptions, we define our work units as follows.

computation

We define one (parallel) computation step to be one instruction sent to a subset of the processors involving at most one add and one multiply.

communication

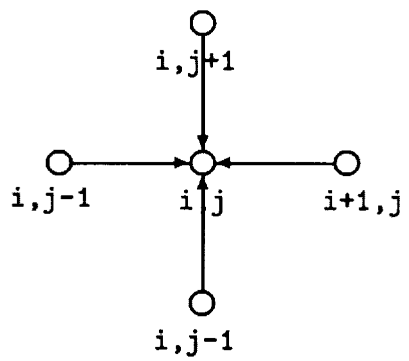
We define one (parallel) communication step on level k to be one instruction sent to a subset of the processors involving at most one fetch of a value from a nearest neighbor (on level k) processor.

To illustrate the method of counting computation and communication steps and to give an example of the type of optimizing which has been assumed in the comparisons of the two algorithms, consider the cost of finding the average of values of nearest neighbors at every grid point,

$$\bar{u}_{ij} = (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})/4.$$

We assume that there are exactly as many grid points as processors. Suppose that each grid point, (i,j), has been assigned to a processor, p_{ij}, and that p_{ij} has u_{ij} in its local memory. At each processor, p_{ij}, the following four operations can be performed simultaneously, the average being stored in $\bar{u}$:

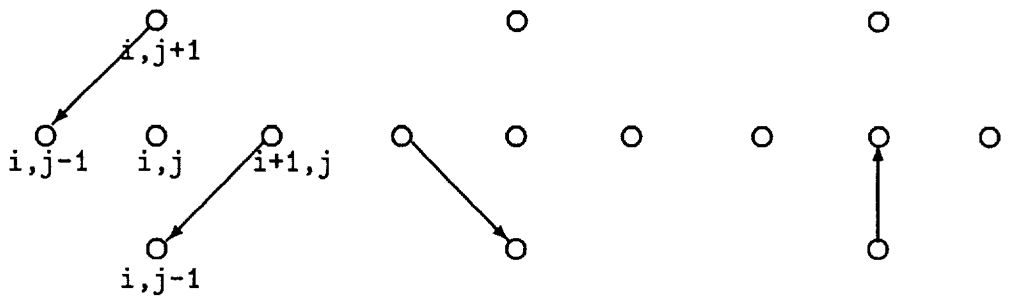
1. fetch $u_{i,j+1}$; store in t_{ij}
2. fetch $u_{i-1,j}$; add to t_{ij}; store in t_{ij}
3. fetch $u_{i,j-1}$; add to t_{ij}; store in t_{ij}
4. fetch $u_{i+1,j}$; add to t_{ij} and divide by 4; store in $\bar{u}_{ij}$



Using the above definitions, this algorithm takes three computation and four communication steps.

However, the same calculation can be done with the following algorithm. At each processor:

1. fetch $u_{i+1,j+1}$; add to u_{ij}; store in t_{ij}
2. fetch $t_{i-1,j+1}$; add to t_{ij} and divide sum by 4; store in t_{ij}
3. fetch $t_{i,j-1}$; store in $\bar{u}_{ij}$



Which takes only two computation and three communication steps.

We also note the following:

- The algorithm we consider involves a fixed number of iterations of the relaxation to be performed on each grid level. Thus approximately the same amount of time is spent on each grid.
- In general, we cannot afford to store $\log(N)$ worth of information on each processor, so the use of simple injection is very important in the PSMG algorithm if all $\log(N)$ grids are used.
- Because of the varying length scales, the communication costs are grid-dependent. All comparisons of communication costs are made between corresponding levels.

2.3 Efficiency measures

The usual measures of efficiency (p.53, [5]) are:

$$op(\epsilon) := w\frac{\log \epsilon}{\log \rho} \tag{1}$$

or

$$\rho_{eff} := \rho^{1/w}$$

where ρ is either the asymptotic convergence factor or some norm convergence factor and w is the number of work units.

We find it more convenient to use the first measure which we refer to as the *normalized work unit.* Using common logarithms, this is a measure of the work required per factor of ten reduction in the error. We take ρ to be the asymptotic convergence factor unless otherwise indicated.

3. The algorithms

Our comparison is made for the same model problem for which the original PSMG algorithm was proposed: Poisson's equation in the unit square with periodic boundary conditions.

We assume that on each intermediate coarse grid the initial iterate is set to zero. Since the same strategy can be used for both methods to solve, or approximately solve, the equations on the coarsest grid(s), it is enough to compare the computation and communication requirements on all other grids. The comparison will be based on a count of the number of computation and communication steps required for any two intermediate grids: for the pre-relaxation (if used), coarse grid correction (residual calculation, residual restriction, the projection of the coarse grid correction to the fine grid and update of u^h) and the post-relaxation. These computation and communication counts per grid level, per cycle, are then normalized to give the work units per factor of ten reduction of the error.

Standard Multigrid

One of the most efficient sequential algorithms is obtained by using red-black Gauss-Seidel sweeps as the relaxation. The operation count of RB multigrid depends on the order in which the sweeps are performed. For example, if a black sweep precedes a residual transfer, the residual,which is then zero at black points, needs to be computed only at red points. The restriction operator can also ignore black point residual values. We have considered various sweep order strategies, and invite the reader to try to find particular combinations which further reduce the costs. We consider standard restriction and projection operators, namely, full weighting (FW), half weighting (HW) and their adjoints relative to the discrete L^2 inner product, FW* (bilinear interpolation), and HW*.

For example, consider using a black-then-red Gauss-Seidel iteration for the pre- and post-smoothing steps, FW restriction and HW* projection. We denote the coarse grid correction

	asymptotic convergence rate	computation steps per grid level	communication steps per grid level	normalized comp. steps per level	normalized comm. steps per level
original PSMG					
five point	.063	17	15	14.2	12.5
mehrstellen	.018	20	18	11.5	10.3
standard RB					
RB T RB	.074	13	12	11.5	10.6
		14	10	12.4	8.8

Table 1: Comparisons for model problem

operator by T and represent this particular algorithm by the notation $RB\,T\,RB$. The red sweep prior to the coarse grid correction insures that the residual is zero at all red points, thus simplifying the restriction to a four point formula. The black sweep immediately following the coarse grid correction eliminates the need to compute the projected correction at black points. Thus the projection of the coarse grid correction at all red points which correspond to coarse grid points is simply its value on the coarse grid, and at the other red points it is zero. The RB T RB can be implemented with 13 parallel computation and 12 parallel communication steps per intermediate grid level and 17 computation and 15 communication steps on the finest level.

It is also possible to reduce the communication to 10 steps by adding another computation step. Which of these implementations to choose obviously depends on whether the computation or communication is more expensive, which is grid level dependent. See Table 1.

PSMG

For the five point discrete Laplacian, the relaxation used by Frederickson and McBryan in PSMG is given by:

$$u^h \leftarrow u^h + \begin{bmatrix} z_2 & z_1 & z_2 \\ z_1 & z_0 & z_1 \\ z_2 & z_1 & z_2 \end{bmatrix} r^h$$

where

$$r^h = f^h - \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix} u^h,$$

for some parameters, $z_i, i = 0, 1, 2$.

In the PSMG algorithm, there is no smoothing in the fine to coarse grid portion of the cycle. In fact, assuming an initial guess of zero on each grid, the residuals need only be computed on the finest grid. In addition, the residual transfer is straight injection, and hence the residual on any coarse grid is precisely the residual at that point on the fine grid. Their prolongation, when combined with the averaging of the four coarse corrections, is given by:

$$v^h \leftarrow \begin{bmatrix} q_2 & q_1 & q_2 \\ q_1 & q_0 & q_1 \\ q_2 & q_1 & q_2 \end{bmatrix} v^{2h}$$

where v^{2h} is $v^{(0,0)}$, $v^{(1,0)}$, $v^{(0,1)}$ or $v^{(1,1)}$, as appropriate. We assume that the optimal z_i and q_i are known, but cannot be assumed to be zero or have any other fixed relationship to one another.

The number of computation steps and the number of communication steps can be minimized simultaneously, requiring a total of 17 computation and 15 communication steps on intermediate grids and 21 computation and 18 communication steps on the finest grid.

The table lists the asymptotic convergence rates for the RB standard multigrid algorithm, calculated as in, for example, [4], and the asymptotic convergence rates given in [3] for PSMG. Changing the discretization of the Laplacian by using the "mehrstellen" nine point formula, they give a convergence rate of .018, but the number of computation and communication steps increases. This is a slightly more efficient algorithm.

Normalizing the costs, i.e., finding the computation and communication costs per factor of ten reduction in error (see equation 1), we see that the standard RB algorithms are more efficient than the five point version of PSMG, and are essentially equivalent to the nine point version of PSMG.

4. Conclusions

The use of multiple coarse grids per level is unattractive both from the point of view of complicating the treatment of boundary conditions and from the point of view of an FMG cycle where monitoring residuals provides valuable information about when sufficient work has been done on a given level. A global check of the residuals on each of the coarse grids (recall, there are $O(4^k)$ coarse grids on the $k+1$st level) is a communication-intensive calculation, and relying on a single residual from each level could be unreliable.

Thus, we see that PSMG, which manages to keep N processors busy solving discretized PDE's with N unknowns, is not significantly better than a reasonably efficient parallelized version of the standard multigrid algorithms. PSMG is just as limited by the inherent constraints of the multigrid techniques as the standard algorithms are.

References

[1] T. Chan and R. Tuminaro. Analysis of a parallel multigrid algorithm. In J. Mandel and S. McCormick, editors, *Fourth Copper Mountain Conference of Multigrid Methods*, 1989.

[2] N. Decker. Parallel multigrid algorithms: Multiscale vs. multigrid. Technical report, Institute for Computer Applications in Science and Engineering, 1989. To appear.

[3] P. Frederickson and O. McBryan. Parallel superconvergent multigrid. In S. McCormick, editor, *Multigrid Methods*. Marcel Dekker, New York, 1988.

[4] K. Stüben and U. Trottenberg. On the construction of fast solvers for elliptic equations. Technical Report IMA-Report Mr. 82.0201, GMD MBH Bonn, 1982.

[5] K. Stüben and U. Trottenberg. Multigrid methods: Fundamental algorithms, model problem analysis and applications. In W. Hackbusch and U. Trottenberg, editors, *Multigrid Methods*, pages 1–176, Berlin, November 1981. Springer-Verlag. Lecture Notes in Mathematics 960.

Report Documentation Page

National Aeronautics and Space Administration

1. Report No. NASA CR-181999 ICASE Report No. 90-17	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle ON THE PARALLEL EFFICIENCY OF THE FREDERICKSON-McBRYAN MULTIGRID		5. Report Date February 1990
		6. Performing Organization Code
7. Author(s) Naomi H. Decker		8. Performing Organization Report No. 90-17
		10. Work Unit No. 505-90-21-01
9. Performing Organization Name and Address Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225		11. Contract or Grant No. NAS1-18605
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225		13. Type of Report and Period Covered Contractor Report
		14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor:
Richard W. Barnwell

Submitted to SIAM Journal on Scientific & Statistical Computing

Final Report

16. Abstract

To take full advantage of the parallelism in a standard multigrid algorithm requires as many processors as points. However, since coarse grids contain fewer points, most processors are idle during the coarse iterations. Frederickson and McBryan claim that retaining all points on all grid levels (using all processors) can lead to a 'superconvergent' algorithm. Has the 'parallel superconvergent' multigrid algorithm, PSMG, of Frederickson and McBryan solved the problem of implementing multigrid on a massively parallel SIMD architecture? How much can be gained by retaining all points on all grid levels, keeping all processors busy?

The purpose of this work is to show that the PSMG algorithm, though it achieves perfect processor utilization, is no more efficient than a parallel implementation of standard multigrid methods. PSMG is simply a new and perhaps simpler way of achieving the same results.

17. Key Words (Suggested by Author(s)) parallel multigrid, PSMG, superconvergent multigrid	18. Distribution Statement 64 - Numerical Analysis Unclassified - Unlimited

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 10	22. Price A02