Research Institute for Advanced Computer Science
NASA Ames Research Center

Normalized Convergence Rates for the PSMG Method

Paul O. Frederickson Oliver A. McBryan

RIACS Technical Report 90.21 **October 1990**

To appear, SIAM Journal on Scientific and Statistical Computing (SISSC)

Normalized Convergence Rates for the PSMG Method

Paul O. Frederickson **Oliver A. McBryan**

The Research Institute of Advanced Computer Science is operated by Universities Space Research Association, The American City Building, Suite 311, Columbia, MD 244, (301)730-2656

Work reported herein was supported by the NAS Systems Division of NASA and DARPA via Cooperative Agreement NCC 2-387 between NASA and the University Space Research Association (USRA). Work was performed at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, CA 94035.

NORMALIZED CONVERGENCE RATES FOR THE PSMG METHOD

PAUL O. FREDERICKSON* AND OLIVER A. MCBRYAN†

Abstract. In a previous paper we have introduced an efficient multiscale PDE solver for massively parallel architectures, which we called Parallel Superconvergent Multigrid, or PSMG. In this paper we derive sharp estimates for the normalized work involved in PSMG solution - the number of parallel arithmetic and communication operations required per digit of error reduction. PSMG is shown to provide fourth-order accurate solutions of Poisson type equations at convergence rates of .00165 per single relaxation iteration, and with parallel operation counts per grid level of 5.75 communications and 8.62 computations for each digit of error reduction. We show that PSMG requires less than half as many arithmetic and one fifth as many communication operations, per digit of error reduction, as a parallel standard multigrid algorithm (RBTRB) presented recently by N. Decker.

* The work of the first author was supported by the NAS Systems Division of NASA and DARPA via Cooperative Agreement NCC 2-387 between NASA and the University Space Research Association (USRA). Work was performed at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, CA 94035.

† Department of Computer Science, University of Colorado, Boulder, CO 80309. Work of the second author was supported by the Air Force Office of Scientific Research, under grant AFOSR-89-0422.

1. Introduction. The PSMG totally parallel multiscale algorithm was introduced in [1] where rigorous upper bounds on the convergence rates attainable were presented. In this paper we present actual PSMG multigrid convergence rates, verified on grids of up to 4 million points. We also provide details of the methods we use for implementing the various PSMG linear operators in order to minimize parallel operation counts. Based on these, we then quantify PSMG performance in terms of the number of parallel operations required per digit of error reduction.

Most high performance (floating point) massively parallel computers require substantially longer to communicate a single number than to perform an arithmetic operation. Consequently we regard normalized communication performance as the most relevant parameter. PSMG requires only 5.75 parallel communication operations per grid level for each digit of convergence. Even if computation speed was much slower than communication, PSMG uses only 8.62 arithmetic operations on each grid level per digit of convergence. These normalized rates assume that there is one processor per fine grid point.

As remarked in [2], PSMG is an example of an *intrinsically parallel algorithm*. It is highly efficient if sufficient processors are available, but is extremely inefficient on serial or low-parallelism computers. In situations where there are substantially more fine grid points than processors, an efficient approach might use a hybrid algorithm - using standard multigrid on the finest grids, but switching to PSMG on grid levels where the number of processors approximates or exceeds the number of grid points. Perhaps the most interesting feature of the PSMG method is not so much the computational efficiency as the ease with which it can be analyzed theoretically, see for example [3].

Recent papers by N. Decker [4, 5] describe a parallel version of a variant (RBTRB) of the standard red-black multigrid algorithm of Stüben and Trottenberg [6]. We compare that algorithm with PSMG in the Appendix, and show that it requires over twice the arithmetic and over five times the communication, for the same level of error reduction. We also briefly analyze in the Appendix the parallelization of the conventionally accepted fastest standard multigrid method for Poisson's equation [6]. While faster than RBTRB it is still up to 4 times slower than PSMG.

A brief but complete description of the PSMG algorithm is presented in section 2 below while section 3 summarizes the results for convergence rates and normalized operation counts. Section 4 describes the methods used to verify the convergence rates, and section 5 provides the algorithms used to minimize the operation counts.

2. The PSMG Algorithm. The PSMG algorithm works with a single grid of points $G^{(L)}$ of size 2^L in each dimension (called the level L grid, or the *fine* grid), but utilizes operators with different scales $l \leq L$ on that grid. Thus the algorithm is strictly speaking multiscale rather than multigrid. There are three basic operators: a finite difference operator A, an interpolation operator Q and a smoothing operator Z. All operators are periodic on the grid in each coordinate direction. The PSMG algorithm extends naturally to both Neumann and Dirichlet boundary conditions, with no increase in convergence rate. The simplest approach to implementing Neumann or

Dirichlet boundary conditions is to use reflection or anti-reflection boundary conditions and an extended grid. However we will discuss only the periodic case here for simplicity.

The operators at scale level l, denoted $A^{(l)}$, $Q^{(l)}$, and $Z^{(l)}$, couple points at a distance $a_l \equiv 2^{L-l}$. Each level l operator is defined at all points of the grid $G^{(L)}$. The basic steps involved at level l, $0 < l \leq L$, for the solution of $A^{(L)}U = f$, starting with an initial guess u, are described by:

Algorithm PSMG(l,u,f):

1. Compute residual $r = f - A^{(l)}u$
2. Project residual to coarse grid: $r = r$ (trivial injection).
3. Solve coarse grid residual equation using PSMG: $e = PSMG(l-1, 0, r)$
4. Interpolate to fine grid: $e' = Q^{(l)}e$
5. Apply a relaxation: $e'' = (I - Z^{(l)}A^{(l)})e' + Z^{(l)}r$
6. Compute and return the new solution: $u'' = u + e''$

An exact solver is utilized on the coarsest grid. The PSMG strategy is to choose $Q^{(l)}$ and $Z^{(l)}$ as functions of $A^{(l)}$ in such a way as to optimize the convergence rate of the above algorithm. Explicit choices for $Q^{(l)}$ and $Z^{(l)}$ are given in [1] for the cases where $A^{(l)}$ represents either the standard 5-point or Mehrstellen discretizations of the Laplacian. In each case we provided upper bounds on the convergence rate for the procedure which are uniform in l.

3. PSMG Performance. In our paper [1] we did not indicate good performance for PSMG on a 5-point operator. The 5-point case was presented in [1] only because the PSMG formalism was easiest to explain in that setting, while the details extended naturally to more complex operators. Indeed the stated multigrid convergence rate upper bound of .2115 per iteration given in [1] for PSMG5-9 was very uninspiring, although we show here that actual 5-point convergence rates are substantially better. In this note, as in [1], we will focus on multigrid convergence rates. For V-cycles, as considered here, it is well known that two-grid rates are often a poor indicator of multigrid convergence rates[6].

We indicated much improved performance in [1] for the Mehrstellen discretization of the Laplacian, where convergence rate bounds as good as .0045 were shown. Thus we had the interesting conclusion that in most cases someone wishing to solve Poisson's equation would be better off using the Mehrstellen discretization which would yield a solution in much shorter time than if a 5-point discretization were used, while providing for free a better (isotropic to fourth order) approximation to the Laplacian. Indeed a fourth order accurate solution is obtained if the right hand side is preprocessed suitably[7].

We have analyzed both 5-point and 9-point discretizations using the definition of computation model given in [4, 5]. For several PSMG methods we present asymptotic convergence rates, the number of parallel arithmetic and communication operations

required on each grid per iteration, and also the normalized operation count for arithmetic and communication. If the asymptotic convergence rate of a method is ρ and the method requires w operations per iteration, the normalized operation count is defined as $w/log_{10}\rho$, and measures the parallel work required per grid level to reduce the error by a factor of 10, see [6]. We summarize the results for several simple cases in Table 1.

TABLE 1: PSMG CONVERGENCE RATES

Method	Convergence Rate	Steps per Level		Normalized Steps	
		Comp.	Comm.	Comp.	Comm.
PSMG 5-9	.08867	14	12	13.31	11.40
PSMG 5-25	.02504	22	16	13.74	9.99
PSMG 9-9	.02165	16	12	9.61	7.21
PSMG 9-25	.00165	24	16	8.62	5.75

The corresponding coefficients for the interpolation operator Q and the smoothing operator Z are (in the notation of [1]):

PSMG5-9: q_0=.25 q_1=.125 q_{11}=.0625
z_0=.278079 z_1=.0534577 z_{11}=.0125615

PSMG5-25: q_0=.361017 q_1=.11458 q_{11}=.0625
q_2=-.0309162 q_{12}=.00521024 q_{22}=.00316188
z_0=.361452 z_1=.0891718 z_{11}=.0293793

PSMG9-9: q_0=.25 q_1=.125 q_{11}=.0625
z_0=.300589 z_1=.0432465 z_{11}=.0139994

PSMG9-25: q_0=.34152 q_1=.0995677 q_{11}=.0625
q_2=-.0199225 q_{12}=.0127161 q_{22}=-.00295755
z_0=.283286 z_1=.0323815 z_{11}=.00835795

Using these coefficients and the definition of the PSMG algorithm given in section 2, one can verify the stated convergence rates. We have in fact verified all of the rates for systems of up to 4 million processors. The method used to verify these rates is described in section 4. The methods used to optimize and compute the numbers of communication and computation steps presented in Table 1 are described in section 5.

4. Verification of Convergence Rates. To verify the rates in Table 1, we have *exactly* computed the spectral radius of the self-adjoint PSMG multigrid error reduction operator $\mathrm{M}^{(L)}$ for all grids $G^{(L)}$ with L ranging from 0 to 11. Here $G^{(L)}$ is a square grid with $n \equiv 2^L$ points on a side as in [1]. The verification is based on the iterative formula:

$$M^{(l)} = Z^{(l)} + (I - Z^{(l)}A^{(l)})\, Q^{(l)}\, M^{(l-1)}, \quad 1 \le l \le L, \tag{1}$$

for the multigrid iteration operator $M^{(l)}$, presented in section 2.4 of [1]. The recursion begins on level 0 with $M^{(0)} = A^{(0)^{-1}}$. The operator $M^{(l)}$ is exactly the same as the operator $\mathbf{PSMG}(l, 0, f)$ defined in section 2 above. The multigrid error correction contraction $\mathbf{M}^{(l)}$ and the multigrid convergence rate μ are defined in [1] as:

$$\mathbf{M}^{(l)} = I - M^{(l)}A^{(l)}, \qquad \mu = \sup_{L} ||\mathbf{M}^{(L)}|| . \tag{2}$$

As in [1], we use italics for the multigrid iteration operator $M^{(l)}$ which attempts to approximate the inverse of A, while the multigrid error reduction operator $\mathbf{M}^{(l)}$ is distinguished with bold typeface.

In a translation invariant problem (1) is most easily studied by Fourier transformation, in which case all quantities in (1) become multiplication operators by functions ${A^{(l)}}_k$, ${Q^{(l)}}_k$ and ${Z^{(l)}}_k$, where $k \equiv (k_1, k_2)$ are the relevant frequencies for grid $G^{(L)}$: $0 \leq k_i < n$. Explicit formulae for these functions ("kernels") are given in [1] for the various 5, 9 and 25 point operators used in the PSMG procedures in Table 1. One final issue relates to the fact that the Poisson equation with periodic boundary data is singular. The correct domain for the translation invariant Poisson equation is the set of grid functions orthogonal to constants. In Fourier space this means that points aliased to $(0,0)$ are omitted from the computation. Equivalently one defines $M^{(0)} \equiv 0$ in (1).

In the paper [1] we introduced a quantity μ^* which is a rigorous upper bound for μ (see section 4.2 of [1]), and then proceeded to give numerically derived estimates for μ^*. The bound μ^*, evaluated in frequency space, involves the supremum over all possible grid levels of a functional of the kernels A, Q, Z, which in turn are simple trigonometric polynomials of the allowed frequencies on a grid - indeed polynomials in the quantities $x_i = cos(2\pi k_i d_l/n)$ where k_1 and k_2 are the frequencies. The numerical approximation used in [1] was to evaluate μ^* by choosing 1000 values for each x_i, uniformly distributed in [-1,1]; i.e. we searched for the maximum on a 1000 × 1000 grid embedded in the domain $[-1,1] \times [-1,1]$ of the (trigonometric) polynomials. Thus there were two sources of inaccuracy in the convergence rates: 1) the fact that μ^* was only an upper bound and 2) the fact that μ^* was itself approximated numerically.

In order to get a more realistic view of PSMG convergence rates, as needed for an effective comparison to the papers [4, 5], we have attempted to compute μ directly rather than through upper bounds. To be specific, we exactly compute the quantity:

$$\mu^{(L)} = ||\mathbf{M}^{(L)}|| \tag{3}$$

on grid $G^{(L)}$. We compute $\mu^{(L)}$ by evaluating the recurrence (1) from $l = 0$ to $l = L$ in Fourier space for every frequency pair k_1,k_2 appropriate to $G^{(L)}$ - i.e. for $0 \leq k_i < 2^L$. The only approximation in this procedure is that the kernels are evaluated in double precision rather than infinite precision arithmetic.

The convergence rates given in Table 1 are the maximum values of $\mu^{(L)}$ for $0 \leq L \leq 11$ and therefore bound the exact convergence rates for all grids up to size 4 million points. In practice we find that the convergence rates $\mu^{(L)}$ are unchanged to several digits of precision beyond about level $L = 6$. We therefore are confident, although this

does not constitute a proof, that the convergence rates in Table 1 extend to arbitrary numbers of grid levels. As a final check we have solved the Poisson equation using PSMG, with zero right hand side and a random initial guess, and in each case verified the convergence rates of Table 1.

We have used the same nonlinear optimizer in these measurements as was used in [1]. The only difference from [1] is that here we are optimizing $\mu^{(L)}$ rather than the substantially less expensive bound μ^*. Indeed we introduced μ^* in [1] solely to provide a cheap evaluation function for the optimization. The fact that actual convergence rates were substantially faster than indicated by μ^* was known from numerical tests of the PSMG algorithm. The parameters given above that yield the convergence rate of .00165, yield a value of .00365 for μ^* in close agreement with the value given in [1]. We have computed μ^* exactly for all grids up to 4 million points, thereby removing the error 2) discussed above introduced by numerical approximation in [1].

5. Operation Counts. We have followed Decker [4, 5] in our definition of the model computational problem and operation counts. In particular a single computation is defined as a parallel add, a multiply, or an add/multiply pair [4, 5]. Furthermore direct communication is allowed only with four nearest neighbors. All arithmetic and communication operations are required to be SIMD. As in [2,3] the operation counts will be those for intermediate grids and the communication unit is grid level dependent - the cost for communication between "nearest neighbors" at that grid level. Notation in this section for the A, Q and Z operators and their parameters follows [1].

Lemma 1 relates the cost of a PSMG intermediate level to the costs of the individual A (difference), Q (interpolation) and Z (smoothing) operators that are utilized. In Table 2 below, we present the operation counts for each of the individual operators encountered in the four PSMG algorithms of Table 1. Thus the operation counts in Table 1 may be verified by combining Lemma 1 with Table 2.

Some of the operation counts in Table 2 may be found in [4, 5], or are slightly sharper than counts given in [4, 5]. To maintain brevity, we will present in Lemmas 2 and 3 a complete accounting of all of the operators in Table 2 that are required to verify the costs for the two best algorithms in Table 1: PSMG9-9 and PSMG9-25. The counts for the 5-point cases are derived from similar but simpler lemmas.

LEMMA 1. *At intermediate grid levels* $(0 < l < L)$, *the parallel communication and computation costs for the PSMG algorithm are given by:*

$$comm(PSMG(l)) = comm(Q) + comm(A) + comm(Z).$$
$$comp(PSMG(l)) = comp(Q) + comp(A) + comp(Z) + 2.$$

Proof: The PSMG algorithm was described as 6 separate steps in section 2 above to which we now refer. For intermediate grids $(0 < l < L)$ the initial guess is taken to be 0 so that step 1 is not needed. Similarly step 6 is relevant only to the top level grid L. Step 2 is free because PSMG uses injection, while step 3 is counted at level $l - 1$ or lower. Thus only steps 4-5 are to be counted at level l. Apart from the 3

operators involved, we note that no communication is required in these steps while two computations are required in step 5. This completes the proof.

Based on Lemma 1, the critical issue is therefore the count of computation and communication costs for the application of the A, Q and Z operators. A careful analysis provides the following values for the various cases of A, Q and Z operators used in our algorithms.

TABLE 2: OPERATION COUNTS FOR A, Q, AND Z		
Operator	Comp. Steps	Comm. Steps
5-pt A (Laplacian)	3	4
9-pt A (Laplacian)	5	4
9-pt Q (Interpolation)	4	4
25-pt Q (Interpolation)	12	8
9-pt Z (Relaxation)	5	4

We present here all of the algorithms needed to verify the counts in Table 2 for the case of the optimal PSMG9 9 and PSMG9-25 algorithms. We will prove the counts of Table 2 for the cases 9-pt A, Q, Z in Lemma 2 and for the 25-pt Q in Lemma 3. Decker [4, 5] has also presented an optimal algorithm for the case of bilinear interpolation.

Each of $A^{(l)}$, $Q^{(l)}$, $Z^{(l)}$ is translation invariant, and is represented as a *difference star* in the notation of [6]. We assume a symmetric form for Q and Z, given by the following representations of the upper right quadrants of the stars:

$$Q_{25}{}^{(l)} = \begin{bmatrix} q_2 & q_{12} & q_{22} \\ q_1 & q_{11} & q_{12} \\ q_0 & q_1 & q_2 \end{bmatrix}, \quad Z_9{}^{(l)} = \begin{bmatrix} z_1 & z_{11} \\ z_0 & z_1 \end{bmatrix}.$$

LEMMA 2. *Evaluation of $w = Zu$ where Z is the symmetric 9-point operator with three parameters z_0, z_1, z_{11}, may be accomplished in parallel with 5 computational and 4 communication steps. The computations reduce to 4 if $z_1{}^2 = z_0 z_{11}$, as is the case for the 9-pt Q.*

Proof: If $z_{11} \neq 0$, we precompute $a = z_1/z_{11}$ and $b = z_0 - z_1{}^2/z_{11}$. The algorithm then consists of the following 3 steps, beginning with a function u on the grid and ending with the function $w = Zu$. Each step is executed in parallel on all processors (points i, j). In case $z_1{}^2 = z_0 z_{11}$, the third step is not needed. (The case $z_{11} = 0$, not relevant here, is even simpler, and may be accomplished with 3 computations and 4 communications).

Operation	Comp.	Comm.
$v(i,j) = z_{11} * (u(i-1,j) + u(i+1,j)) + z_1 * u(i,j)$	2	2
$w(i,j) = v(i,j-1) + v(i,j+1) + a * v(i,j)$	2	2
$w(i,j) = w(i,j) + b * v(i,j)$	1	0
Total Cost for all Steps:	5	4

LEMMA 3. *Evaluation of $w = Qu$ where Q is the symmetric 25-point operator with six parameters q_0, q_1, q_{11}, q_2, q_{12}, q_{22}, may be accomplished in parallel with 12 computational and 8 communication steps.*

Proof: If $q_{12} \neq 0$, we precompute $a = q_{22}/q_{12}$, $b = q_{11}/q_{12}$, $c = q_2/q_{12}$, and $d = q_1/q_{12}$. The algorithm then consists of the following 9 steps, beginning with a function u on the grid and ending with the function $w = Qu$. Each step is executed in parallel on all processors (points i,j). (The case $q_{12} = 0$, not relevant to this paper, is even simpler, involving only a 17-point operator, and may be accomplished with 10 computations and 8 communications).

Operation	Comp.	Comm.
$f(i,j) = u(i+1,j)$	0	1
$g(i,j) = u(i-1,j)$	0	1
$r(i,j) = z_{12} * (f(i,j) + g(i,j))$	1	1
$s(i,j) = z_{12} * (f(i+1,j) + g(i-1,j))$	1	2
$t(i,j) = a * s(i,j) + r(i,j) + z_2 * u(i,j)$	2	0
$x(i,j) = s(i,j) + b * r(i,j) + z_1 * u(i,j)$	2	0
$y(i,j) = x(i,j) + t(i,j+1)$	1	1
$z(i,j) = x(i,j) + t(i,j-1)$	1	1
$w(i,j) = y(i,j+1) + z(i,j-1) + c * s(i,j)$ $+ d * r(i,j) + z_0 * u(i,j)$	4	2
Total Cost for all Steps:	12	8

6. Conclusions. PSMG is shown to provide solutions of Poisson type equations at convergence rates of .00165 per iteration, and with parallel operation counts per digit of error reduction as low as 5.75 communications and 8.62 computations on each grid level.

7. Appendix: Comparisons to Standard Multigrid. Recent papers by N. Decker [4, 5] describe a parallel version of a variant (RBTRB) of the standard red-black multigrid algorithm of Stüben and Trottenberg [6]. The papers [4, 5] provide an extremely efficient parallel RBTRB implementation, using a very clever rolling together

of several phases of the multigrid procedure for RBTRB. Furthermore those papers introduce the correct method of comparing PSMG and standard multigrid - by comparison of normalized convergence rates for communication and computation, rather than comparing absolute convergence rates. We differ however with [5] and [4] on several key points, as outlined respectively in the following two paragraphs The effect is to produce a qualitative difference from [4] and a smaller quantitative but still significant difference from the more recent paper [5][1].

While [5] used the convergence rate upper bounds from [1], PSMG actually converges faster than those theoretical bounds as shown here in Table 1. Furthermore, optimal parallel algorithms for various PSMG steps (relaxation and interpolation) were not used in [5], but are provided here in Lemmas 2 and 3. The effect of these differences is to reduce the normalized PSMG9-25 costs from 12.3 computations and 8.5 communications as reported in [5] to 8.62 computations and 5.75 communications as reported here. The differences for the 5-point operators are even larger, approaching a factor of two.

We note in passing that several further differences existed between the original technical report [4] and the published paper [5]. These resulted in the report's conclusion that RBTRB is as fast as PSMG even when there are as many processors as fine grid points. In part this was because the report [4] omitted from consideration the PSMG9-5 and PSMG9-25 methods, known from [1] to have the fastest convergence rates, and shown here to be the most efficient. Additionally [4] used a two-grid RBTRB rate of .074 in the comparisons rather than the multigrid rate of .19 used here and in [5]. Finally the report assigned the same cost to diagonal communications as to nearest neighbor communications in analyzing the RBTRB method. While these differences qualitatively affected the results of [4], the basic approach to comparing the PSMG and standard multigrid methods outlined in [4] is not in dispute, and the RBTRB algorithm as presented in [4] stands as the most efficient standard MG implementation that we have seen.

A comparison of Table 1 in [4, 5] and in this paper shows the effects of these differences on PSMG performance relative to RBTRB. As shown in [5], RBTRB requires 13 arithmetic and 21 communication operations for a multigrid convergence rate of .19, yielding normalized values of 18.02 parallel arithmetic and 29.12 parallel communication operations per digit of error reduction. Comparing with the PSMG figures presented here in Table 1, we see that RBTRB requires 2.09 times as much arithmetic and 5.15 times as much communication as PSMG. The algorithm BRTBR has a multigrid convergence rate of .12, and would appear to provide a slightly better method than RBTRB, although communication costs differ somewhat.

One possibility for a faster parallel standard multigrid would be to use what is con-

[1] In June 1990, the SISSC editor provided a paper described as the final version of [5] accepted for publication. However changes to that version occurred in October 1990. Their nature was communicated to us by the editor although we have not seen the final text. The comments in the Appendix are based on the information we have about the October version. A previous version of this report, dated June 1990, differed by referring to the June 1990 Decker paper, but our results on PSMG are unchanged.

ventionally regarded as the fastest serial multigrid solver for the Poisson equation [6]. The algorithm, which we will call BR3, uses two BR sweeps, half-weighting restriction, bilinear interpolation and one final BR sweep (we are using reverse operator notation here as in [5]). This algorithm has a multigrid convergence rate of .059 [6]. For the first red (R) sweep the initial guess is 0 on intermediate grids, allowing the red sweep to be folded in with the black sweep ($u = h^2 f/4$) as in [5], which requires 3 arithmetic and 4 communication steps, abbreviated as 3+4 below. Each of the following red and black steps is a 4-point average which also requires 3+4 operations (similar to application of the 5-point A). The residual computation, $r = f - Au$, requires 4+4 operations. Because the black residuals are zero, the half-weighting restriction reduces to injection of red values, multiplied by .5. This requires zero operations as the multiply can be absorbed in the previous residual step. The bilinear interpolation is an application of the 3×3 Q and therefore costs 4+4 operations. However this may be reduced to 2+4 by observing that only the values at black points are actually required for the following red sweep. The final BR sweep consumes 3+4 operations for each phase. The total operation count for intermediate grids is therefore 21+28, with corresponding normalized rates of 17.09 arithmetic and 22.78 communications per digit of error reduction, a substantial improvement over RBTRB, especially if communication dominates. It is likely that with care these numbers may be optimized somewhat. However it is doubtful that they could approach the 5.75 communication rate of PSMG 9-25.

We do not address in this paper the question of whether other standard multigrid cycles may give better parallel performance than BR3. We refer to [8] for a detailed comparison of several hundred red-black standard multigrid methods, including several that are somewhat more efficient than the BR3 method.

REFERENCES

[1] P. FREDERICKSON AND O. MCBRYAN, *Parallel superconvergent multigrid*, in Multigrid Methods, S. McCormick, ed., Marcel Dekker, New York, 1988.

[2] O. MCBRYAN, *New Architectures: Performance highlights and new algorithms*, Parallel Computing 7, (1988) 477-499.

[3] P. FREDERICKSON AND O. MCBRYAN, *Recent Developments for Parallel Multigrid*, Proceedings of the Third European Conference on Multigrid Methods, October 1990, to appear.

[4] N. DECKER, *On the Parallel Efficiency of the Frederickson-McBryan Multigrid*, ICASE Report No. 90-17, Feb 1990.

[5] N. DECKER, *A Note on the Parallel Efficiency of the Frederickson-McBryan Multigrid Algorithm*, SIAM Journal on Scientific and Statistical Computing, to appear. (See footnote 1 in the Appendix of this paper).

[6] K. STÜBEN AND U. TROTTENBERG, *Multigrid Methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Berlin, 1981, Springer-Verlag, pp. 1–176. Lecture Notes in Mathematics 960.

[7] L. COLLATZ, *The numerical treatment of differential equations*, Springer-Verlag 1966, pp. 542.

[8] O. MCBRYAN, *Sequential and Parallel Efficiency of Multigrid Fast Solvers*, University of Colorado CS Dept. Tech Report, Sept 1990.