TOTALLY PARALLEL MULTILEVEL ALGORITHMS

Paul O. Frederickson

November 23, 1988

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 88.34

NASA Cooperative Agreement Number NASW-4234

RIACS

Research Institute for Advanced Computer Science

TOTALLY PARALLEL MULTILEVEL ALGORITHMS

Paul O. Frederickson

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 88.34
November 23, 1988

Abstract

Four totally parallel algorithms for the solution of a sparse linear system have common characteristics which become quite apparent when they are implemented on a highly parallel hypercube such as the CM2. These four algorithms are Parallel Superconvergent Multigrid (PSMG) of Frederickson and McBryan, Robust Multigrid (RMG) of Hackbusch, the FFT based Spectral Algorithm, and Parallel Cyclic Reduction. In fact, all four can be formulated as particular cases of the same totally parallel multilevel algorithm, which we will refer to as TPMA. In certain cases the spectral radius of TPMA is zero, and it is recognized to be a direct algorithm. In many other cases the spectral radius, although not zero, is small enough that a single iteration per timestep keeps the local error within the required tolerance.

Keywords: multilevel algorithm, multigrid, cyclic reduction, spectral method.

Work reported herein was supported in part by Contract NASW-4234 from the National Aeronautics and Space Administration (NASA) and to the Universities Space Research Association (USRA).

TOTALLY PARALLEL MULTILEVEL ALGORITHMS

Paul O. Frederickson*

RIACS, Ames Research Center
Moffett Field, CA 94035

Abstract

Four totally parallel algorithms for the solution of a sparse linear system have common characteristics which become quite apparent when they are implemented on a highly parallel hypercube such as the CM2. These four algorithms are Parallel Superconvergent Multigrid (PSMG) of Frederickson and McBryan, Robust Multigrid (RMG) of Hackbusch, the FFT based Spectral Algorithm, and Parallel Cyclic Reduction. In fact, all four can be formulated as particular cases of the same totally parallel multilevel algorithm, which we will refer to as TPMA. In certain cases the spectral radius of TPMA is zero, and it is recognized to be a direct algorithm. In many other cases the spectral radius, although not zero, is small enough that a single iteration per timestep keeps the local error within the required tolerance.

Keywords: multilevel algorithm, multigrid, cyclic reduction, spectral method.

1. Introduction

The multilevel algorithms we are discussing are all designed to solve large and highly parallel problems on massively parallel computers in as short a time as possible. We say that a *problem* is highly parallel if the defining equations can be applied efficiently to a proposed solution on a massively parallel computer. The operative definition of the phrase

* Work reported herein was supported in part by Contract NASW-4234 from the National Aeronautics and Space Administration (NASA) to the Universities Space Research Association (USRA)

massively parallel computer may well change with time as increasingly parallel machines are built. We expect, however, that the Connection Machine [1] is sufficiently parallel that the algorithms we demonstrate on it now will be effective on the still larger machines that we will see in the near future. Because parallel supercomputers are now becoming available it is reasonable to begin the development of algorithms capable of solving those very large problems which have, in the past, been too large and complex to consider seriously.

We describe the general algorithm TPMA (Totally Parallel Multilevel Algorithm) in section three, after establishing some notation in section two. In sections four through seven we show that this algorithm has as special cases the algorithms PSMG (Parallel Superconvergent Multigrid) of Frederickson and McBryan [2,3], RMG (robust Multigrid) of Hackbusch [4], the Fast Fourier Transform based Spectral Method [5,6], and Parallel Cyclic Reduction [7-9]. It seems plausible that the common framework we establish will lead to a clearer understanding of the functioning of all of these algorithms, and aid in the development of improved algorithms through hybridization of the old.

To put the algorithm TPMA in a proper perspective we will first describe a class of problems for which one version or another of this general algorithm appears to be optimal, depending on the circumstances. We will consider, for this purpose, a class of problems that arise in computational models of three dimensional fluid flow based on partial differential equations. Numerical solution of the generic problem in this class often requires repeated solution of an equation of the form

$$\mathcal{F}(\, u \,) \;=\; v, \tag{1.1}$$

where $\mathcal{F}$ denotes a system of nonlinear differential operators, v is a collection of vector and scalar fields representing the problem data, and u is the desired solution, also a collection of vector and scalar fields. There are no direct algorithms for solving problems of this class. Thus we will need to develop a sequence u_n of approximate solutions which define u as their limit. All known methods for the construction of a convergent sequence of approximate solutions to a nonlinear problem of this sort require the evaluation of a residual

$$r_n = v - \mathcal{F}(u_n) \tag{1.2}$$

at each step, followed by a *defect correction* step in which u_{n+1} is specified by constructing the increment $u_{n+1} - u_n$ as a function of the residual r_n.

2. Approximate Inverse Operators. The typical algorithm for the construction of $u_{n+1} - u_n$ as a function of r_n is linear in r_n. It is an algorithmic representation of a linear operator $\mathcal{B} : \mathcal{Y} \to \mathcal{X}$, and we are able to write

$$u_{n+1} - u_n = B\, r_n. \tag{2.1}$$

Since any matrix representation of the restriction of $\mathcal{B}$ to finite dimensional subspaces of $\mathcal{X}$ and $\mathcal{Y}$ would only be of theoretical interest for problems of the size that interest us here, we will restrict our attention to algorithmic representations B of $\mathcal{B}$.

For the best of the algorithms we are able to provide an error bound of the form

$$\| r_n - A\,(u_{n+1} - u_n) \| < \epsilon \| r_n \|, \tag{2.2}$$

in which the operator A approximates the linearization of $\mathcal{F}$ about u_n. It follows from (2.1) and (2.2) that the operators A and B are related by

$$\| I - AB \| < \epsilon. \tag{2.3}$$

We will refer to B as an *ϵ-approximate inverse* of the operator A if it satisfies (2.3). Our goal is to construct an approximate inverse operator which is highly parallel, has low operation count, and is highly accurate. The algorithm TPMA that we describe below appears to meet these needs in many situations of interest.

3: The Totally Parallel Multilevel Algorithm TPMA

Multilevel algorithms for the solution of a large sparse linear system are characterized by their effective use of solutions to a hierarchy of smaller and simpler linear systems to construct a solution to the given system. The general multilevel algorithm for the solution of a problem at level k in the hierarchy, which we will denote by B^k, involves the following four steps:

Projection. The problem data is projected onto one or more smaller systems at level $k-1$ using a linear operator that we will denote P^k.

Subsystem Solution. Each of these smaller systems is solved using the algorithmic linear operator B^{k-1}.

Interpolation. These solutions are combined, using a generalized interpolation operator Q^k, into an approximate solution at level k.

Smoothing. A smoothing operator S^k, usually an approximate inverse in its own right, is applied to the resulting residual at level k.

Since each of P^k, Q^k and S^k is linear, we are able to describe the algorithm B^k for $k > 0$ with the equation

$$B^k = S^k + (I - S^k A^k) Q^k B^{k-1} P^k \tag{3.1}$$

while at the lowest level $B^0 = (A^0)^{-1}$. In some cases, as we shall see, the algorithm TPMA is exact with $S^k = 0$, in which case (3.1) reduces to

$$B^k = Q^k B^{k-1} P^k, \tag{3.1'}$$

which may be compactly represented by the commutative diagram

$$\begin{array}{ccc} \mathcal{X}^k & \xleftarrow{B^k} & \mathcal{Y}^k \\ \uparrow Q^k & & \downarrow P^k \\ \mathcal{X}^{k-1} & \xleftarrow{B^{k-1}} & \mathcal{Y}^{k-1} \end{array} \tag{3.2}$$

In other cases either P^k or Q^k is the identity operator, further simplifying the computation. The recursive definition of B^k is closely related to the recursive definition of the operator A^k at level k using the Galerkin projection $A^{k-1} = P^k A^k Q^k$, illustrated in the commutative diagram

$$\begin{array}{ccc} \mathcal{X}^k & \xrightarrow{A^k} & \mathcal{Y}^k \\ \uparrow Q^k & & \downarrow P^k \\ \mathcal{X}^{k-1} & \xrightarrow{A^{k-1}} & \mathcal{Y}^{k-1} \end{array} \tag{3.3}$$

This definition of the operator A^k is used in most versions of the algorithm TPMA, although we will see exceptions.

4: The algorithm PSMG

The algorithm PSMG (Parallel Superconvergent Multigrid) is our primary example of a totally parallel multilevel algorithm. For clarity we consider first a rather simple problem, namely the two dimensional Laplacian on a rectangle with periodic boundary conditions. The fact that this problem is singular will cause us no difficulty. In this case the operators commute, being convolution operators, and we are able to afford a significant saving in storage by taking

$$P^k = I, \quad 0 \leq k \leq l. \tag{4.1}$$

As a result of this simplification we do not need to store the projections of the problem data, but always use the data at the highest level. Moreover, we are able to write u^k over u^{k-1}, so the solution process requires only two arrays in all (in addition to the temporary variables introduced by the compiler). The savings is almost a factor of $log(n)$, which allows us to solve much larger problems.

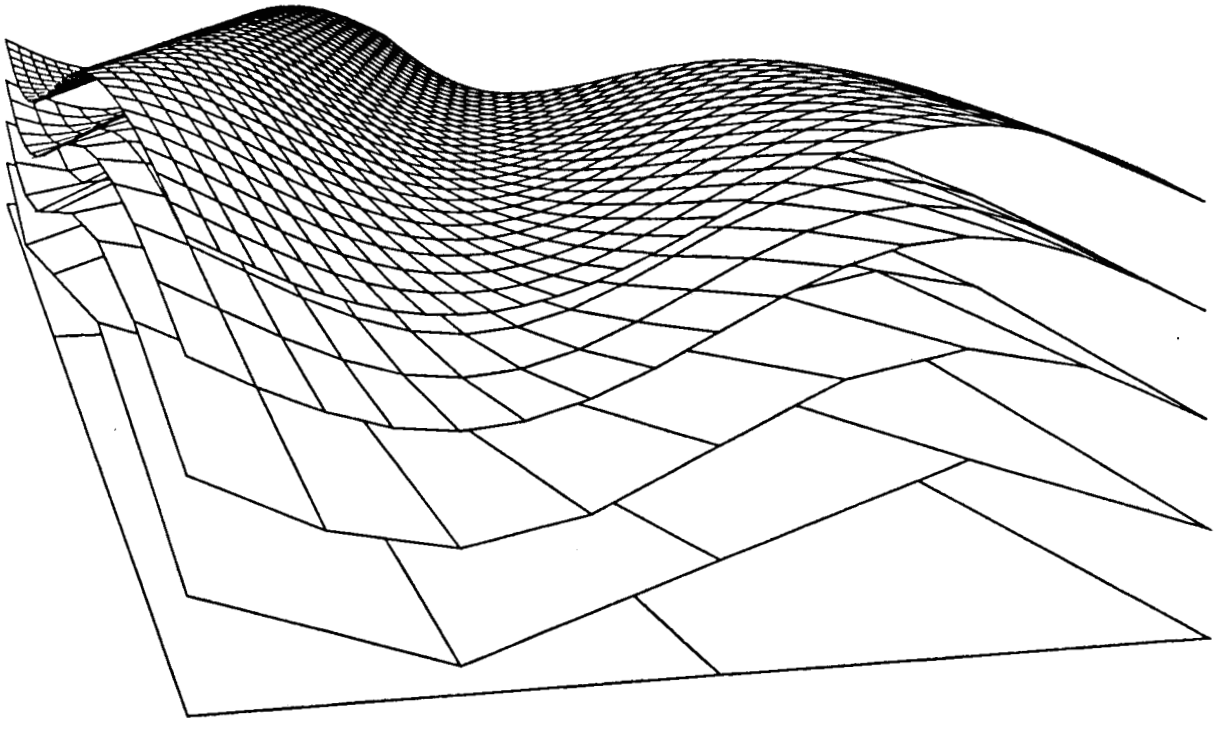

Figure 1. A multilevel approximation of a scalar function on a rectangle.

We have found that there is considerable advantage to using the mehrstellen discretization of the negative Laplacian defined by

$$A^k = \frac{1}{6}\begin{bmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{bmatrix} \tag{4.2}$$

rather than the bilinear finite element discretization

$$A^k = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{4.3}$$

which is invariant under the Galerkin projection (3.3), or the always popular five-point discretization

$$A^k = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{4.4}$$

In our implementation on the Connection Machine we have chosen a data structure for the arrays r^k and u^k which leaves them superimposed on the arrays for the higher and lower levels. This means that the operators at level k are scaled, when $k < l$, to operate on grid points a distance 2^j apart, where $j = l - k$. The coefficients are invariant from one level to the next, in this simple case, but the scales on which they operate are not.

We have found that there is an advantage to a larger operator Q, which now takes on the role of P as well. We have obtained very good results with

$$Q = \begin{bmatrix} -.005464 & .013850 & -.017536 & .013850 & -.005464 \\ .013850 & .062500 & .097300 & .062500 & .013850 \\ -.017536 & .097300 & .341997 & .097300 & -.017536 \\ .013850 & .062500 & .097300 & .062500 & .013850 \\ -.005464 & .013850 & -.017536 & .013850 & -.005464 \end{bmatrix} \tag{4.5}$$

and the smoothing operator Z given by

$$Z = \begin{bmatrix} .01566 & .04649 & .01566 \\ .04649 & .3059 & .04649 \\ .01566 & .04649 & .01566 \end{bmatrix} \tag{4.5}$$

The algorithm PSMG was developed by Oliver McBryan and the author [3,4] after McBryan had implemented a standard multigrid algorithm on the Connection Machine and observed good convergence but inefficient operation: most of the processors were idle most of the time. We asked ourselves how these idle processors could be used most efficiently in furthering the computation. For further motivation behind the multigrid method see the early paper of Brandt [10]. The recent book of Hackbusch [11] provides an excellent review of current multigrid algorithms, and puts their convergence analysis in a firm theoretical framework, while a variety of parallel implementations are discussed in references [12-16].

5: Hackbusch's Robust Multi-Grid algorithm RMG

In the development of PSMG it was assumed that the approximate problems at the next lower level would be nearly identical, differing only in their interwoven relationship to the finer grid. Hackbusch observes [4] that there are advantages, particularly when the given problem is highly variable in space, to using *different projection operators* for each of

these lower level approximations. Since the algorithm has a wider domain of convergence (even though it may not converge as rapidly for nice problems) he refers to it as RMG, for Robust Multi-Grid.

Hackbusch specifies the four projection operators $P_{0,0}$, $P_{0,1}$, $P_{1,0}$ and $P_{1,1}$ in a two-dimensional example of PSMG (he uses the notation $r_{i,j}$) to be the duals of the four interpolation operators $Q_{0,0}$, $Q_{0,1}$, $Q_{1,0}$ and $Q_{1,1}$ (for which his notation is $p_{i,j}$) given by the equations

$$Q_{0,0} = \begin{bmatrix} .25 & .50 & .25 \\ .50 & 1.0 & .50 \\ .25 & .50 & .25 \end{bmatrix} \quad Q_{1,0} = \begin{bmatrix} -.25 & .50 & -.25 \\ -.50 & 1.0 & -.50 \\ -.25 & .50 & -.25 \end{bmatrix}$$

$$Q_{0,1} = \begin{bmatrix} -.25 & -.50 & -.25 \\ .50 & 1.0 & .50 \\ -.25 & -.50 & -.25 \end{bmatrix} \quad Q_{1,1} = \begin{bmatrix} .25 & -.50 & .25 \\ -.50 & 1.0 & -.50 \\ .25 & -.50 & .25 \end{bmatrix} \tag{5.1}$$

It is clear that this may be implemented with essentially the same software that we have used in implementing PSMG, with a short preprocessing step in which the signs of some of the coefficients are changed.

6: Spectral Methods based on the FFT

The classical spectral algorithm for the solution of Poisson's problem on a square or cube has three steps: take the FFT of the problem data v, divide this by the transform of the Laplacian (or some other constant coefficient differential operator), and transform back. We may, however, equally well describe it as an example of the algorithm TPMA in which the operator P is given, in two dimensions, by

$$P_{0,0} = \frac{1}{2}\begin{bmatrix} .00 & .00 & .00 \\ .00 & 1.0 & {\omega_k}^{i_0} \\ .00 & {\omega_k}^{i_1} & {\omega_k}^{i_0+i_1} \end{bmatrix} \quad P_{0,1} = \frac{1}{2}\begin{bmatrix} .00 & .00 & .00 \\ 1.0 & -{\omega_k}^{i_0} & .00 \\ {\omega_k}^{i_1} & -{\omega_k}^{i_0+i_1} & .00 \end{bmatrix}$$

$$P_{1,0} = \frac{1}{2}\begin{bmatrix} .00 & 1.0 & {\omega_k}^{i_0} \\ .00 & -{\omega_k}^{i_1} & -{\omega_k}^{i_0+i_1} \\ .00 & .00 & .00 \end{bmatrix} \quad P_{1,1} = \frac{1}{2}\begin{bmatrix} 1.0 & -{\omega_k}^{i_0} & .00 \\ -{\omega_k}^{i_1} & {\omega_k}^{i_0+i_1} & .00 \\ .00 & .00 & .00 \end{bmatrix} \tag{6.1}$$

It is clear that this may be implemented with essentially the same TPMA software that we have used in implementing PSMG or RML, which helps to clarify the strong theoretical relationship between these fast algorithms. It is often advantageous, however, to implement these operators one dimension at a time, in effect first doing

$$P_0 = \frac{1}{\sqrt{2}}(.00 \quad 1.0 \quad {\omega_k}^{i_0}) \qquad P_1 = \frac{1}{\sqrt{2}}(1.0 \quad -{\omega_k}^{i_0} \quad .00) \tag{6.2}$$

followed by pair of operators transverse to these. Actually, there may be a strong computational advantage in not multiplying by the indicated zero, for on the connection machine this means that this data does not need to be communicated. As we will see later, it is possible to arrange the communication on a hypercube, and the CM2 in particular, in such a way that the communication step takes just half as long. There may also be an advantage to keeping the factor $\frac{1}{\sqrt{2}}$ on the cuff until the end of the computation, and divide once by 2^l.

An important observation is that the Fourier multipliers ${\omega_k}^i$ need be computed only once, on the highest level. Those coefficients needed at the next lower level are a Hamming distance at most two away, and are moved in at the start of the computation at that level.

7: Parallel Cyclic Reduction Algorithms.

There are important applications which require the solution of a separable elliptic equation on a rectangle, or the image of a rectangle under a separable coordinate transformation. For many years the algorithm of choice, when the circumstances are right, has been one of the generalizations of the Cyclic Reduction algorithm of Hockney [17] and Buneman [18]. Notable among these generalizations are the Fast Poisson Solver of Buzbee, Golub and Nielsen [19], the algorithm FACR of Hockney[20], and the FISHPAK routines of Swartztrauber and Sweet [21-22]. Jespersen and Levit [23] have recently implemented a Navier-Stokes solver on the connection machine at Ames Research Center in which they use a parallel algorithm developed by Levit and based on the the work of Hockney and Jessope [7]. For further discussions of parallel generalizations see Johnsson [24] and Ortega [25].

Consider first the classic cyclic reduction algorithm. It is easily described as Gaussian elimination in an unusual order: the even numbered rows are eliminated first, then half the remaining rows, and so on until $log(n)$ steps later only one block remains. (Thus the alternate name *odd even elimination* for the algorithm.) The parallel cyclic reduction algorithm of Hockney and Jessope [7] stems from the observation that when one has sufficiently many processors it is efficient to apply this transformation in parallel on all the rows for all $log(n)$ steps of the reduction process at which point one need only do one (parallel) solve. When the parallel cyclic reduction algorithm is applied to a block tridiagonal system in which the off-diagonal blocks are identity operators it leads to a sequence of block tri-diagonal systems of the form

$$(A^k u^k)_i \;=\; u^k{}_{i-2^j} \;+\; a^k\, u^k{}_i \;+\; u^k{}_{i+2^j} \;=\; v^k{}_i \tag{7.1}$$

with the right hand sides defined recursively by

$$v^{k-1}{}_i \;=\; P^k\, v^k{}_i \;=\; v^k{}_{i-2^j} \;-\; a^k\, v^k{}_i \;+\; v^k{}_{i+2^j}, \tag{7.2}$$

and in which the diagonal blocks a^k are given by

$$a^{k-1} \;=\; 2I \;-\; a^k a^k. \tag{7.3}$$

Observe that the operators A^k defined by (7.1) and (7.2) and the projection operators P^k defined by (7.2) and (7.3) are related by

$$A^{k-1} \;=\; P^k A^k, \tag{7.4}$$

which is (3.3) with $Q^k \;=\; I$. This is another way of saying that the interpolation operators are not needed in this version of the algorithm TPMA.

8: Information Exchange on the CM2

All multilevel algorithms require long distance communication, for it is by spreading problem information and solution information over long distances that they obtain their computational advantage. All of the totally parallel multilevel algorithms that we have discussed here have one further characteristic in common: the distances over which they communicate is always a power of two. This gives parallel computers with a hypercube architecture a considerable advantage, for on hypercubes it is possible to send a message a grid distance 2^k in only two hops, provided the grid is laid out according to the binary reflected gray code BRGC (or at least a gray code which is a bit-permutation of BRGC).

Moreover, if the computer is built to send information on two lines simultaneously, then there is a communication scheme with the property that a distance 2^k exchange takes no longer, in principle, than a distance one exchange. Moreover, if information can be sent in both directions simultaneously on any line an exchange will be twice as fast as use of either get or send. We note that the CM2 satisfies the first property, and timing experiments indicate that the second property is close to true.

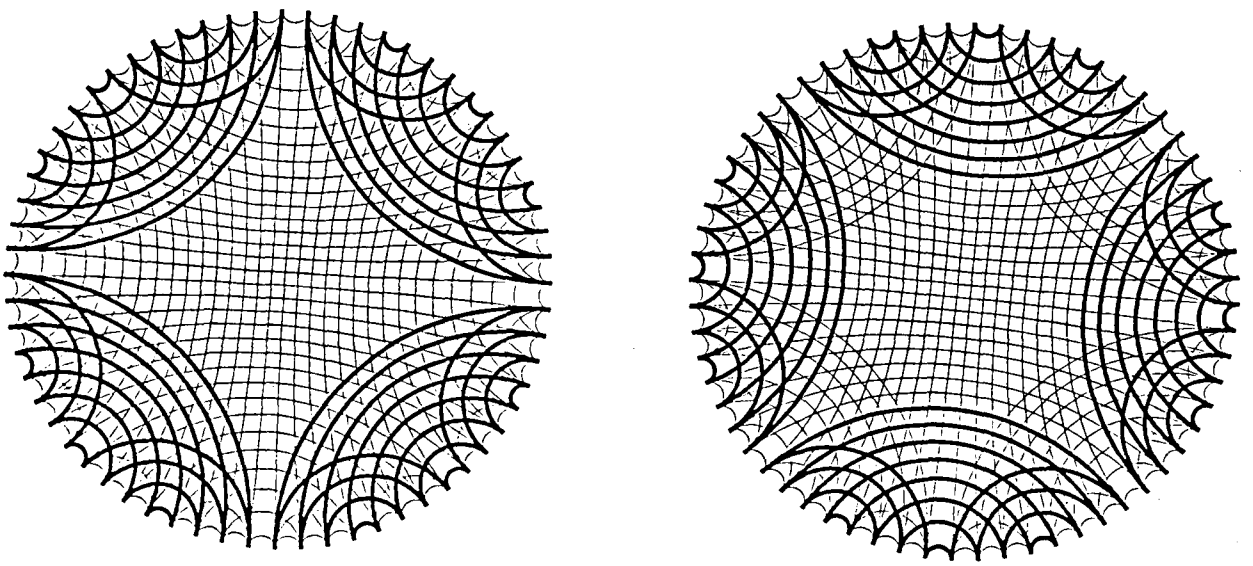

Figure 2. Communication lines active during phase 0 of a distance 16 exchange are shown in (a), and those active during phase 1 of the exchange are shown in (b).

The exchange of information proceeds in two phases, during each of which half the processors are communicating with distant neighbors to their left (or below them) in grid ordering and half are communicating with neighbors to their right. As is clear from Fig. 3, one communication line is active during both phases, and is therefore a bottleneck which prevents a faster communication scheme. This is communication line $k-1$ for distance 2^k communication. (When a higher dimensional grid is in use we consider only one dimension at a time, and refer to the lowest order line dedicated to that dimension as line 0.)

During phase 0 of the exchange line k is also active on all processors, transferring information upwards for half of the processors and downward for the other half. Although it is not clear from the figure, the set of communication lines active during phase 1 is not the same on all processors. If processor i is exchanging with processor j during this phase, the other line in use is the index of the highest nonzero bit in i **XOR** j. The three communication lines active during this exchange are, therefore, $k-1$, k and $\lfloor lg\ (\ i\ XOR\ j\ )\ \rfloor$. At this point we should observe that the communication required by the FFT is provided by phase 0 alone, and thus requires only half the time if it is arranged in this manner, as the communication required by the other versions of TPMA we have discussed.

Acknowledgements

Most of the work reported here is the result of a long and fruitful collaboration with Oliver McBryan. The research benefited greatly from discussions with Walter Tichy, Creon Levit and Eric Raible on hypercube communication schemes, with Roland Sweet, Tom Jordan and Youcef Saad on cyclic reduction algorithms, and with David Bailey and Paul Swartztrauber on parallel implementations of the FFT algorithm. Finally, the work of Craig Douglas and Willard Miranker [26] was a motivation for the present research, even though it moves in a somewhat different direction.

REFERENCES

[1] W. Daniel Hillis, *The Connection Machine*, MIT Press, Cambridge, Massachusetts, 1985.

[2] Paul O. Frederickson and Oliver A. McBryan, *Parallel superconvergent multigrid*, in *Multigrid Methods*, S. F. McCormick (editor), Marcel Dekker, New York and Basel, 1988.

[3] Paul O. Frederickson and Oliver A. McBryan, *Intrinsically Parallel Multiscale Algorithms for Hypercubes*, in *The Third Conference on Hypercube Concurrent Computers and Applications*, Geoffry Fox (editor), ACM Press, New York, 1988.

[4] Wolfgang Hackbusch *A new approach to robust multi-grid methods* Invited Lecture at ICIAM, Paris, June 29, 1987, and Bericht 8708, Christian-Albrechts-Universitaet Kiel, 1987 to appear in *Proceedings of the ICIAM'87 Conference*, SIAM books, Philadelphia, 1988.

[5] David Gottlieb and Steven A. Orszag. *Numerical Analysis of Spectral Methods, NSF-CBMS Monograph, No. 26*, SIAM, Philadelphia, 1977.

[6] Parviz Moin, John Kim and Robert Moser. *Turbulence statistics in fully developed channel flow at low reynolds number*, J. Fluid Mech. 177 (1987) 133–166.

[7] Roger Hockney and Chris Jessope, *Parallel Computers: Architecture, Programming and Algorithms*, Adam Hilger, Bristol, 1981.

[8] Paul N. Swartztrauber and Roland A. Sweet, *Vector and parallel methods for the direct solution of Poisson's equation*, to appear.

[9] Roland A. Sweet, *A parallel and vector variant of the cyclic reduction algorithm*, SIAM J. Sci. Stat. Comput. 9 (1988) 761–765.

[10] Achi Brandt, *Multi-level adaptive solutions of boundary-value problems*, Math. Comp. 31 (1977) 333–390.

[11] Wolfgang Hackbusch *Multi-Grid Methods and Applications* Springer-Verlag, Berlin Heidelberg New York Tokyo, 1985.

[12] Maurice W. Benson and Paul O. Frederickson, *Fast Pseudo-Inverse Algorithms on Hypercubes*, in *Multigrid Methods*, S. F. McCormick (editor), Marcel Dekker, New York and Basel, 1988.

[13] Tony F. Chan and Youcef Saad, *Multigrid Algorithms on the Hypercube Multiprocessor*, Research Report 368, Dept Computer Science, Yale University, 1985

[14] Tony F. Chan and Rob Schreiber, *Parallel Networks for Multigrid Algorithms: Architecture and Complexity*, SIAM J. Sci. Stat. Comput. 6 (1985) 698-711.

[15] Tony F. Chan and Ray S. Tuminaro, *A Survey of Parallel Multigrid Algorithms*, in *Parallel Computations and their Impact on Mechanics*, ASME, 1988.

[16] Oliver A. McBryan and Eric Van de Velde, *The multigrid method on parallel processors*, in *Multigrid Methods II*, ed. W. Hackbusch and U. Trottenberg, Lecture Notes in Mathematics, vol. 1228, Springer-Verlag, Berlin, Heidelberg, 1986.

[17] Roger Hockney, *A fast direct solution of Poisson's equation using Fourier analysis*, J. Assoc. Comput. Mach. 12 (1965) 95–113.

[18] O. Buneman, *A compact non-iterative Poisson solver*, Report 294, Stanford University Institute for Plasma Research, Stanford, CA, 1969.

[19] Bill Buzbee, Gene Golub and Clair Nielsen, *On direct methods for solving Poisson's equations*, SIAM J. Numer. Anal. 7 (1970) 627–656.

[20] Roger Hockney, *The potential calculation and some applications*, Meth. Comp. Physics. 9 (1970) 135–211.

[21] Paul N. Swartztrauber, *A direct method for the discrete solution of separable elliptic equations*, SIAM J. Numer. Anal. 11 (1974) 1136–1150.

[22] Paul N. Swartztrauber and Roland A. Sweet, *Efficient FORTRAN subprograms for the solution of elliptic partial differential equations*, ACM Trans. Math. Soft. 5 (1979) 352–364.

[23] Creon Levit and Dennis Jespersen, *Explicit and implicit solution of the Navier-Stokes equations on a massively parallel computer*, to appear in *Symposium on Advances and Trends in Computational Structural Mechanics and Fluid Dynamics*, Pergammon Press.

[24] S. Lennart Johnsson, *Solving tridiagonal systems on ensemble architectures*, SIAM J. Sci. Stat. Comput. 8 (1987) 354–392.

[25] James M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York and London, 1988.

[26] Craig C. Douglas and Willard L. Miranker, *Some nontelescoping parallel algorithms Based on Serial Multigrid / Aggregation / Disaggregation Techniques*, in *Multigrid Methods*, S. F. McCormick (editor), Marcel Dekker, New York and Basel, 1988.