COMPACT MULTIGRID

VICTOR PAN† AND JOHN REIF‡

Abstract. The bit-complexity is a realistic complexity measure for computations on parallel computers such as the CONNECTION MACHINE (CM1) and the MASPAR. For a large class of linear PDEs satisfying some routine assumptions of the multigrid methods, the N point discretization of their solution is compressed to a constant number of bits per discretization point with no loss of information and without introducing errors beyond the order of the discretization error. Namely, it is shown that the bit-complexity of the compressed solution is $O(N)$ for the storage space and, if the PDE has (piecewise) constant coefficients, then also for the total number of bit-parallel operations. The compressed solution is also computed by using time $O(\log N)$ and $N/\log N$ bit-serial processors. The known bounds on the bit-complexity (for both sequential time and storage space) were at least $N \log N$; moreover, the order of $N \log N$ bit-serial processors was required to support the $O(\log N)$ parallel time in the known algorithms. It is believed that this is the first time when the solution to a linear system has been provably compressed (i.e., the bit-complexity of storage of the compressed solution is less than the solution size) and also the first case where the use of data compression provably speeds up the time to solve the system (in the compressed form).

Key words. multigrid, data compression, partial differential equations, low precision computing, algorithms, complexity

AMS(MOS) subject classifications. 65P05, 65F10

1. Introduction.

1.1. Motivation and limitations. Approximate solution of partial differential equations (PDEs) is usually obtained by means of their discrete approximation and by solving a sparse linear algebraic system of equations. Of course, the amount of the discretization affects the accuracy of the approximation. If N discrete approximation points have been chosen over a regular (two- or three-dimensional) grid, then for a very large and important class of linear PDEs, which we will call *weakly smooth*, the solution to the linear algebraic systems approximates to the solution of the PDE with an error of the order of N^{-g}, for some constant $g > 0$. Such discretization errors have been well studied by numerical analysts from the 1920s [CFL], [BR], [BS], [FW], [SF].

Thus the approximation gives us, at each discretization point, the first $O(\log N)$ bits of the actual value of the solution to the PDE, and the high accuracy solution of two- and three-dimensional PDEs requires in particular that the number of discretization points N grows very large. The computation is often more limited by the storage constraints than by the time constraints (particularly if it is desired to store the solution in the primary storage memory without the use of the much slower secondary storage of the conventional machines). It is therefore important to investigate methods that substantially reduce this storage by compressing the data in the solution. As we will see, this is indeed possible in some important cases and is a surprisingly fundamental

† Computer Science Department, State University of New York, Albany, New York 12222 and Mathematics and Computer Science Department, Lehman College, City University of New York, Bronx, New York 10468. The work of this author was supported by National Science Foundation grant CCR-8805782 and by PSC-CUNY awards 661340, 668541, and 669290.

‡ Department of Computer Science, Duke University, Durham, North Carolina 27706. The work of this author was supported by Air Force contract AFOSR-87-0386, Office of Naval Research contract N00014-87-K-0310, Defense Advanced Research Projects Agency contract N00014-88-K-0458, Office of Naval Research contract DAAL03-88-K-0195, and National Aeronautics and Space Administration/CESDIS subcontract 550-63NAS5-30428URSA.

and also has a relatively large amount of primary storage. For such a machine, the arithmetic complexity model has generally been considered to be appropriate. However, a machine such as the Cray is capable of performing a bit-vector operation in one parallel step (for example, it may perform AND, OR or NOT operations on hundreds of bits), so that the parallel bit-complexity of such machines can also be of interest.

On the other hand, fine grained massively parallel machines (such as the Connection Machine) have been designed with large numbers of bit-serial processors (requiring a relatively long time to execute an arithmetic operation) with a very limited memory, which is generally accessed bit-serially. A complexity model for parallel algorithms must take into account both the bit-serial nature of the processors and the limited memory constraints; in particular, we feel that the parallel complexity is most reasonably measured by the bit-complexity. In this model, we assume that each memory cell holds only one bit, and each processor can do a single bit-operation per step.

1.3. Previous solutions of PDEs

The solution of the linear algebraic systems approximating PDEs can be computed by means of a number of well-known and now classical algorithms. For example, for a large class of PDEs, we may apply the standard linearly convergent iterative solution algorithms, such as Gauss–Siedel's, and find the solutions to the auxiliary linear systems up to the maximum accuracy of $O(\log N)$ bits at N points in $O(\log N)$ iterations, using N processors. However, each such an iteration generally requires an arithmetic operation over the $O(\log N)$-bit numbers and hence requires at least $O(\log N)$ bit-operations per point. Thus the total work is $O(N \log N)$ arithmetic operations or $O(N \log N)$ bit-operations. Various multigrid methods were first proposed by Fedorenko and Bakhvalov in the 1960s, and then by Astrakhantzev and Brandt in the early 1970s for the solution of these linear systems approximating PDEs (see Refs [3–8]). In Brandt [7], it was claimed that these multigrid methods required only $O(N)$ arithmetic operations; this was rigorously proved in Bank and Dupont [9] (also, see Refs [10–14]). (Actually, the multigrid methods are effective even in many cases where the classical iterative algorithms converge too slowly.) The works of Refs [15–19] all describe parallel algorithms that take $O(\log N)$ arithmetic steps using N processors, and thus use the order of $N \log N$ arithmetic operations, which is off by the factor of $\log N$ from the optimum. It follows that the known multigrid methods require a total of $O(N \log N)$ bit-operations, and at least the order of $N \log N$ bit-serial processors to support the order of $\log N$ time. The bit-operation bound appears to be optimal since the binary representation of the solution occupies a total of the order of $N \log N$ bits.

1.4. Our results

Our main goal is a rigorous study of the bit-complexity of these linear algebraic systems approximating linear PDEs. In spite of the lack of theoretical investigation into this area, we feel that the problems are fundamental in nature. In this paper we will show some surprising properties of the linear algebraic systems approximating to weakly and strongly pseudo regular linear PDEs (see more on pseudo regularity below and see the formal definitions in Section 2):

(1) For weakly pseudo regular PDEs, their solutions can be significantly compressed to $O(1)$ bits per solution point (which, by the factor of $\log N$, improves the previous storage requirements) by a data structure that we call the *compact multigrid data structure*. (We do not know of any previous provable results for data compression of the solutions of any type of linear systems or of any other algebraic systems.)

(2) For strongly pseudo regular PDEs, their compressed solutions can be very efficiently computed (both sequentially and in parallel) by an algorithm which we also call *compact multigrid*. The bit-complexity of our compressed solution algorithms is $O(\log N)$ time, $N/\log N$ bit-serial processors and a total of $O(N)$ bit-operations, which is optimum since the size of the compressed solution is of the order of N. This is by the factor of $\log N$ improvement of the bounds on both sequential time and storage space of the known algorithms and by the factor of a $\log^2 N$ decrease of the number of bit-serial processors supporting $O(\log N)$ time.

Note that already the $\log N$ factor is significant for even relatively small problems; for example, this factor is 10 or more for problems of size $N > 1000$, such as the three-dimensional grid of size $10 \times 10 \times 10$.

A bit-serial data communication required by our compact multigrid algorithm happens to be what is known as a pyramid network, consisting of a sequence of grids $L_0, L_1, \ldots, L_k$, where the grid L_i has 2^{di} points and where each node of the ith grid is connected to its $2d$ neighbors in the current grid and also to the corresponding nodes of the $(i+1)$th and $(i-1)$th grids. It is well-known that such a pyramid network can be compactly mapped to a hypercube network with the same number of nodes (within the factor of 2) such that each edge of the pyramid has a corresponding edge of a hypercube; therefore, our compact multigrid algorithm can be efficiently implemented on a hypercube network with the same complexity (within a factor of 2).

The weak pseudo regularity assumption suffices for property (1) to hold, and this assumption is just the very mild and customary bound $O(1/N^c)$ on the discretization errors (see bound (3) below), but even the strong pseudo regularity assumptions, required for property (2) to hold, are still satisfied for a large class of linear PDEs. Specifically, the strong pseudo regularity extends the weak pseudo regularity assumption essentially just by adding the requirement that an iterative algorithm (such as multigrid or SSOR) for solving the auxiliary linear systems over all the grids uses a constant number of steps on each grid in order to decrease the approximation error norms linearly with the same rate for all the grids. What we call strong pseudo regularity is in fact *a routine assumption* of the multigrid methods [20].

Given an N point uncompressed solution, the compression can be done in $O(\log N)$ time using N bit-serial processors, for a total of $O(N \log N)$ amount of work, which is optimal since the input solution is of size $O(N \log N)$. A very simple decompression algorithm requires only $O(\log N)$ sequential bit-operations to access the full precision [of $O(\log N)$ bits] solution value at any discretization point.

The compressed solution can be stored, and it can be decompressed only when its values need to be output. In many practical applications, the solutions need not be decompressed. For example, in the solution of the time dependent PDEs, the most customary solution methods perform at a discrete sequence of, say, T time steps. In each time step, a PDE is approximately solved, using an N point discretization of the PDE fixed at that time value and using the approximate solution obtained (in the compressed form) at the previous time step as an initial approximation to the current solution. Thus the solutions at these time steps need not be decompressed, except for the solution at the final time step. The total bit-complexity estimate for this computation, including decompression of the final solution and T calls for compact multigrid, would be $O(N(T + \log N))$ bit-operations [requiring $O(T \log N + \log^2 N)$ time and using $N/\log N$ bit-serial processors]. Here we need the strong pseudo regularity and, in particular, the linear convergence assumption; if it holds initially, we shall preserve it by using sufficiently small time steps.

1.5. Organization of the paper

We will specify our compression techniques for PDEs in Sections 2–5. We will simplify our presentation, assuming, in particular, the simple lattice grids, although our results hold for more general discretization sets. In Section 6 we indicate some further extensions of our results, in particular, to more general discretization sets. In the appendix we add a method for saving the storage space in the parallel solution of a general well-conditioned algebraic linear system.

2. SOME DEFINITIONS AND ASSUMPTIONS

To simplify our presentation, we will study the linear PDEs on the unit d-dimensional cube, discretized over a family of d-dimensional lattices $L_0, L_1, \ldots, L_k$, where each point of L_j lies at the distance $h_j = 2^{-j}$ from its $2d$ nearest neighbors, so that there are exactly $|L_j| = N_j = 2^{dj}$ points in L_j, for $j = 0, 1, \ldots, k$, and the overall number of points equals $N_k = N = 2^{dk}$, where $k = (\log_2 N)/d$, provided that we identify the boundary points whose coordinates only differ by 0 or 1 from each other. (On the actual discretization grids for PDEs, all the boundary points are distinct, and so the grids contain slightly more than N_j points.)

Let $u(x)$ denote a function in the d-dimensional vector, let $u(x)$ be the solution to the PDE, and let $u_j(x)$ for a fixed $x \in L_j$ denote the respective component of the N_j-dimensional vector u_j representing the solution to the linear system,

$$D_j u_j = b_j, \tag{1}$$

of the difference equations generated by the discretization of the PDE over the lattice L_j, so that $\Delta_j(x) = u(x) - u_j(x)$, for $x \in L_j$, denotes the discretization error function on L_j for $j = 1, \ldots, k$.

Surely, discretization of the linear PDE gives matrices D_j with $O(1)$ nonzero coefficients per row as $j \to \infty$. Let $u_0(x) = 0$ for $x \in L_0$, and let $\hat{u}_{j-1}(x)$, for $j = 1, 2, \ldots, k$, denote the prolongation of $u_{j-1}(x)$ from L_{j-1} to L_j, obtained by the interpolation (which usually means just the averaging) of the values of $u_{j-1}(x)$ at the appropriate points of L_{j-1} lying near x. Then

$$u_j(x) = \hat{u}_{j-1}(x) + e_j(x), \quad x \in L_j, \quad j = 1, \ldots, k, \tag{2}$$

where $e_j(x)$ denotes the interpolation error on L_j.

We will assume that the discretization and interpolation errors satisfy the two following bounds, which we will call the assumptions of *the weak pseudo regularity* of the PDE:

$$|\Delta_j(x)| \leqslant 2^{c-\alpha j}, \tag{3}$$

$$|e_j(x)| \leqslant 2^{c-\alpha j}, \tag{4}$$

for all $x \in L_j$, $j = 1, \ldots, k$, and for fixed $c \geqslant 0$ and $\alpha \geqslant 1$. Assumption (3) holds for a wide class of PDEs (see, for instance, Refs [21, p. 29; 22]), including the well-posed linear PDEs, as well as many nonlinear PDEs. Such an assumption is routinely made in the analysis of the multigrid methods (e.g. Refs [6–9]); in particular, the auxiliary grid problems are said to be "solved to the level of truncation" defined by bound (3) (see McCormick [20, p. 26]).

As a part of the weak pseudo regularity assumption, let us further assume that $u(x)$ has been scaled so that $|u_j(x)| \leqslant 1$ for $x \in L_j$ and for all j and that every $e_j(x)$ is represented with (that is, rounded-off to) α binary bits (digits).

Let us show that bound (3) implies bound (4) assuming that $\hat{u}_{j-1}(x_j) = u_{j-1}(x_j)$ for $x_j \in L_{j-1}$ and that $\hat{u}_{j-1}(x_j)$ has been defined on $L_j - L_{j-1}$ as the average of $u_{j-1}(x)$ at all the $2d$ points x of L_{j-1} such that $|x - x_j| = h_j$. Then $|e_j(x_j)| \leqslant |u_j(x_j) - u_{j-1}(x)|$ for at least one of these points. First rewrite bound (3) as follows:

$$|\Delta_j(x_j)| \leqslant a h_j^{\gamma}, \quad x_j \in L_j, \tag{5}$$

where a and γ are positive constants, and h_j is the length of a side of the mesh L_j, so that $h_{j-1} = 2h_j$ for the lattices L_j that we have chosen. Let $\gamma < 1$, $x_{j-1} \in L_{j-1} \subset L_j$, $x_j \in L_j$, $|x_j - x_{j-1}| = h_j$, and $|e_j(x_j)| \leqslant |u_j(x_j) - u_{j-1}(x_{j-1})|$. Then we deduce from condition (5) that for some Θ, $0 \leqslant \Theta \leqslant 1$,

$$\begin{aligned}|e_j(x_j)| &\leqslant |u_j(x_j) - u_{j-1}(x_{j-1})| \leqslant |u(x_j) - u_j(x_j)| + |u(x_{j-1}) - u_{j-1}(x_{j-1})| \\ &\quad + |u(x_j) - u(x_{j-1})| \leqslant a h_j^{\gamma} + a h_{j-1}^{\gamma} + |u'(x_j + \Theta h_j)| h_j \leqslant a^* h_j^{\gamma},\end{aligned}$$

that is,

$$|e_j(x_j)| \leqslant a^* h_j^{\gamma}, \tag{6}$$

where $a^* = a(1 + (h_{j-1}/h_j)^{\gamma}) + |u'(x_j + \Theta h_j)| h_j^{1-\gamma} \leqslant 3a + \max|u'(x)| h_j^{1-\gamma}$. Bounds (5) and (6) turn into bounds (3) and (4) for $c = \log_2 a^*$, $\alpha = -\gamma(\log_2 h_j)/j$ and $x = x_j$. If $x_j \in L_{j-1}$ then we just replace x_{j-1} by x_j above and cancel the term $|u(x_j) - u(x_{j-1})|$.

Remark

We may replace bounds (3), (4) and $|u_j(x)| \leqslant 1$ by the bounds

$$\|\Delta_j\| \leqslant 2^{c-\alpha j} \|u_j\|,$$

$$\|e_j\| \leqslant {}^{c-\alpha j} \|u_j\|,$$

for a fixed vector norm, provided that Δ_j, e_j and u_j are considered as N_j-dimensional vectors with the components $\Delta_j(x)$, $e_j(x)$ and $u_j(x)$ for $x \in L_j$. This modification would not change our resulting estimates for the complexity of the compact multigrid.

In Section 4, we will assume *pseudo regularity*, which means the weak pseudo regularity together with the assumption that the prolongation from L_{j-1} to L_j only requires $O(1)$ time using N_j bit-serial processors (which is surely the case for the interpolation by averaging).

In Section 5, we will assume *strong pseudo regularity*, that is, in addition to the pseudo regularity, we will assume that:

(1) a fixed iterative algorithm (such as Jacobi, Gauss–Seidel, SSOR or a multigrid algorithm) for linear systems with matrices D_j uses $O(1)$ multiplications of submatrices of D_j by vectors for every j, in order to linearly decrease, by the factor independent of j and N, the norm of the error of the approximation to the solution $u_j(x)$ of system (1);

(2) the entries of the matrices D_j for all j, as well as the components of b_j, are integers having magnitudes $O(1)$ or turn into such integers after the scaling and truncation of the entries of system (1).

3. COMPRESSION OF THE OUTPUT DATA

Now assume the weak pseudo regularity relations and compress approximations to all the N values of $u_k(x)$ on L_k within absolute errors of at most $2^{c-\alpha k}$, so as to decrease the storage space required.

For the straightforward fixed point binary representation of these values of $u_k(x)$, we generally need $N\lceil \alpha k - c \rceil$ binary bits.

As an alternative, let us store $u_k(x)$ on L_k in the compressed form by recursively approximating within $2^{c-\alpha j-\alpha}$ to the fixed point binary values $e_j(x)$ for $x \in L_j$, $j = 1, \ldots, k$. The storage space of $2^d\lceil \alpha - c \rceil + \alpha(N_2 + N_3 + \cdots + N_k) < 2^d\lceil \alpha - c \rceil + 2\alpha N = O(N)$ binary bits suffices for this compressed information, which means saving roughly the factor of $k = \log_2 N$ binary bits against the straightforward representation.

It is convenient to assume the fixed point binary representation in order to estimate and to compare the numbers of binary bits used in both representations of the output, but shifting to the floating point representation would not actually require to increase these estimates.

4. RECOVERY OF THE OUTPUT FROM THE COMPRESSED DATA

In this section, we will assume the pseudo regularity. To recover $u_k(x)$ on L_k from the compressed information given by $e_j(x)$ on L_j, for $j = 1, \ldots, k$, we start with $u_0(x) = 0$ for $x \in L_0$ and recursively, for $j = 1, \ldots, k$, compute the values

(a) $\hat{u}_{j-1}(x)$ on L_j, by prolongation of $u_{j-1}(x)$ from L_{j-1} to L_j, and then

(b) $u_j(x)$ on L_j, by applying equations (2).

Both stages (a) and (b) are performed within the precision $2^{c-\alpha j-\alpha}$, so that the stage (b) amounts to appending α binary bits of $e_j(x)$ to the available string of binary bits in the fixed point binary representation of $\hat{u}_{j-1}(x)$ for each $x \in L_j$, and the stage (a) amounts to scanning the values of $u_{j-1}(x)$ on L_{j-1} and to the summation of few β-bit binary numbers [where, say, $\beta = O(\alpha)$] defined by the β least significant binary bits in the representation of $u_{j-1}(x)$ for appropriate x from L_{j-1}. Since

$$\sum_j N_j = O(N),$$

the computational complexity estimates for stages (a) and (b) stay within the bounds stated in the introduction.

5. COMPUTING THE COMPRESSED SOLUTION BY COMPACT MULTIGRID

In this section, we will assume the strong pseudo regularity of the PDE. The time complexity of computing the compressed data structure is dominated by the time required to obtain the solution vectors e_j for the linear systems of equations over L_j for $j = 1, \ldots, k$:

$$D_j e_j = r_j, \tag{7}$$

where

$$r_j = b_j - D_j \hat{u}_{j-1}, \tag{8}$$

the matrices D_j and the vector b_j are from linear systems (1), and the vectors e_j and $\hat{u}_{j-1}$ have components $e_j(x)$ and $\hat{u}_{j-1}(x)$ for $x \in L_j$, defined by systems (1) and (2), respectively.

We will follow the routine of the V-cycle multigrid methods (cf. Refs [16, 20]) and will evaluate the vectors e_j recursively for $j = 1, \ldots, k$. Initially, we will let $u_0(x) = 0$ for $x \in L_0$, and at stage j, we will successively compute for all $x \in L_j$:

(a) $\hat{u}_{j-1}(x)$ [by prolongation of $u_{j-1}(x)$ from L_{j-1} to L_j],

(b) $r_j(x)$ [by using equations (8)],

(c) $e_j(x)$ [by solving linear system (7) "to the level of truncaton"],

(d) $u_j(x)$ [by using equations (2), as in Section 4].

We may then restrict $u_{j+1}(x)$ to $u_j(x)$ for $j = k-1, k-2, \ldots, 1$ (this stage contains no smoothing iterations, unlike some customary multigrid algorithms) and then recursively repeat such a V-cycle.

Part (1) of the strong pseudo regularity assumption means that the errors of the approximations to $u_j(x)$ decrease by a constant factor independent of j and N when stages (a)–(d) are repeated once for all j, even if only $O(1)$ iteration steps are used at stage (c) for solving linear systems (7) for every j. Such convergence results have been proven for the customary multigrid algorithms applied to a wide class of PDEs [9–14, 23].

Let us estimate the time complexity of these computations, dominated by the time needed for solving linear systems (7).

The size $|L_j| = 2^{dj}$ of linear system (7) increases by 2^d times as j grows by 1. Even if we assume that the solution time for system (7) is linear in $|L_j|$, the overall solution time for all the k such systems in terms of the number of arithmetic operations involved is less than $1/(1 - 2^{-d})$ times the solution time for single system (1) for $j = k$, which gives us the uncompressed output values $u_k(x)$ for $x \in L_k$. The bit-operation count is even more favorable to the solution of systems (7) for all j, as opposed to single system (1) for $j = k$, because the output values $e_j(x)$, satisfying systems (7), are sought with the lower precision of α binary bits.

Furthermore, we solve linear systems (7) by iterative methods where each step is essentially reduced to a constant number (say, one or two) multiplications of a matrix D_j or its submatrices by vectors. Due to the linear convergence assumption we made, a constant number of iterations suffices at each step j in order to compute the α desired binary bits of $e_j(x)$.

The computational cost of multiplication of D_j by a vector is $O(N_j)$ arithmetic operations for a sparse and structured discretization matrix D_j [having $O(1)$ nonzero entries in each row]. Moreover, parallel acceleration to the parallel time bound $O(1)$ is possible using N_j processors [for we deal with a matrix-by-vector multiplications, and the matrix has $O(1)$ nonzero entries per row]. Thus we arrive at Proposition 1, whose processor bound follows similarly to the proof of Proposition 2 below.

Proposition 1

$O(\log N)$ parallel arithmetic steps and $N/\log N$ processors suffice to compute the vectors e_j for all j, that is, to compute the smooth compressed solution to a strongly regular PDE discretized over the lattice L_k.

Furthermore, we only need $O(1)$ binary bits in order to represent $e_j(x)$ for every $x \in L_j$ and every j. Since D_j has only $O(1)$ nonzero entries per row and since these entries are integers having magnitudes $O(1)$, it suffices to use $O(1)$ bits to represent $r_j(x)$. (These bits may occupy not the same positions as the nonzero bits of the corresponding components of b_j, whose most significant binary digits may be canceled in subtracting $D_j\hat{u}_{j-1}$.) Thus, we will perform all the arithmetic operations with $O(1)$-bit operands and will arrive at Proposition 2.

Proposition 2

$O(\log N)$ steps, $N/\log N$ bit-serial processors and $O(N)$ storage space under the Boolean model of computation suffice in order to compute the compressed solution to a strongly pseudo regular PDE by using the compact multigrid algorithm.

Proof. As described above, our algorithm has stages $j = 1, \ldots, k = (\log N)/d$, where at stage j we require $O(1)$ time for each of the $N_j = 2^{dj}$ bit-serial processors. Thus our parallel algorithm (if naïvely implemented) appears to take $O(\log N)$ time using N bit-serial processors. However, the first $(\log N)/d - \log\log N$ stages only require $N/\log N$ bit-serial processors. Thus, at each of the last $\log\log N$ stages j, $j = (\log N)/d - \log\log N + 1, \ldots, (\log N)/d$, we will slow down the computation to the time

$$O\left(\frac{2^{dj}\log N}{N}\right),$$

using only $N/\log N$ bit-serial processors. The overall time of our resulting parallel algorithm is then still $O(\log N)$.

6. EXTENSIONS OF THE RESULTS

Our results can be immediately extended to the case of more general sequences of the sets $S_0, S_1, \ldots, S_k$ of the discretization of the PDEs, provided that each set S_j consists of $c_j\sigma^j$ points where $0 < c < c_j < c^*$, $\sigma > 1$, c, c^* and σ are constants (this includes the grids with step sizes that vary depending on the direction of the steps), and that the pseudo regularity assumptions are respectively extended to the case of the sets S_j. We also need to assume a constant degree bound $2d$ for all the discretization points, that is, each of them is supposed to have at most $2d$ neighbors: this will imply that each equation of the associated linear algebraic system has at most $O(d)$ nonzero coefficients.

Finally, the presented approach can be further extended to some nonlinear PDEs, as long as our assumptions [such as systems (3) and (4)] hold and as long as dealing with nonlinear systems of difference equations replacing linear systems (1) remains relatively inexpensive.

Acknowledgements—The work of the first author (V.P.) was supported by NSF Grant CCR-8805782 and by PSC–CUNY Awards Nos 668541 and 669290. The work by the second author (J.R.) was supported by Air Force Contract AFOSR-87-0386, ONR Contract N00014-87-K-0310, DARPA/ISTO Contract N00014-88-K-0458, ONR Contract DAAL 03-088-K-0195 and NASA/CESDIS Subcontract 550-63 NAS 5-30428 URSA.

REFERENCES

1. R. Courant, K. O. Friedrichs and H. Lewy, Uber die partiellen Differenzengleich-ungen der mathematischen Physik. *Math. Ann.* **100,** 32–74 (1928).
2. G. Strang and G. Fix, *An Analysis of the Finite Element Method.* Prentice-Hall, Englewood Cliffs, N.J. (1973).
3. R. P. Fedorenko, The speed of convergence of one iteration process (in Russian). *Zh vȳchisl. Mat. mat. Fiz.* **4,** 559–663 (1964).
4. N. S. Bakhvalov, On the convergence of a relaxation method under natural constraints on an elliptic operator (in Russian). *Zh vȳchisl. Mat. mat. Fiz.* **6,** 861–883 (1966).
5. G. P. Astrakhantzev, An iterative method of solving elliptic net problem (in Russian). *Zh vȳchisl. Mat. mat. Fiz.* **11,** 439–448 (1971).
6. A. Brandt, Multilevel adaptive technique (MLAT) for fast numerical solutions to boundary value problems. In *Proc. 3rd Int. Conf. Numerical Methods in Fluid Mechanics*, Paris (1972); and *Lecture Notes in Physics*, Vol. 18, pp. 82–89. Springer, Berlin (1984).
7. A. Brandt, Multi-level adaptive solutions to boundary value problems. *Math. Comput.* **31,** 333–390 (1977).
8. A. Brandt, Multigrid Techniques: 1984 Guide, with Applications to Fluid Dynamics. Available as GMD Studien Nr. 85, GMD-AIW, Postfach 1240, D-5205, St Augustin 1, G.D.R. (1984).
9. R. Bank and T. Dupont, An optimal order process for solving finite element equations. *Math. Comput.* **36,** 35–51 (1981).
10. W. Hackbusch, On the convergence of multi-grid iteration applied to finite element equations. Report 77-8, Universität zu Köln, July (1977).
11. W. Hackbusch and U. Trottenberg (Eds), *Multigrid Methods, Lecture Notes in Mathematics*, Vol. 960. Springer, Berlin (1982).
12. W. Hackbusch, *Multigrid Methods and Applications.* Springer, Berlin (1985).
13. S. McCormick (Ed.), *Proc. 2nd Copper Mountain Multigrid Conf.* (Special issue). *Appl. Math. Comput.* **19,** 1–372 (1986).
14. S. McCormick and U. Trottenberg (Eds), *Multigrid Methods* (Special issue). *Appl. Math. Comput.* **13,** 213–474 (1983).
15. A. Brandt, Multi-grid solvers on parallel computers. ICASE Technical Report 80-23, NASA Langley Research Center, Hampton, Va (1980).

16. P. O. Frederickson and O. A. McBryan, Parallel superconvergent multigrid. *Multigrid Methods: Theory, Applications and Supercomputing* (Ed. S. McCormick), *Lecture Notes in Pure and Applied Math.*, Vol. 100, pp. 195–210. Dekker, New York (1988).
17. T. F. Chan, Y. Saad and M. H. Schultz, Solving elliptic partial differential equations on hypercubes. *Hypercube Multiprocessors 1986.* SIAM, Philadelphia, Pa (1986).
18. O. McBryan and E. Van de Velde, Parallel algorithms for elliptic equations. *Communs pure appl. Math.* **38,** 769–795 (1985).
19. O. McBryan and E. Van de Velde, The multigrid method on parallel processors, in *Multigrid Methods II* (Eds W. Hackbusch and U. Trottenberg). *Lecture Notes in Mathematics*, Vol. 1228. Springer, Berlin (1986).
20. S. McCormick (Ed.) *Multigrid Methods*, Vol. 3, SIAM Frontiers Series. SIAM, Philadelphia, Pa (1987).
21. W. F. Ames, *Numerical Methods for Partial Differential Equations.* Academic Press, New York (1977).
22. L. Lapidus and G. F. Pinder, *Numerical Solution of Partial Differential Equations in Science and Engineering.* Wiley, New York (1982).
23. P. O. Frederickson and O. A. McBryan, Superconvergent multigrid methods. Cornell Theory Center. Preprint, May (1987).

APPENDIX

Space Efficient Parallel Solution of a Well-conditioned Linear Algebraic System of Equations

In this appendix, we will give a space efficient methodology (but not a data compression technique), which we also suggest for reducing local storage in a parallel solution of a general well-conditioned linear algebraic system of equations. The idea is to subdivide the original problem into the problem of parallel solution of several linear systems, with a substantial decrease of temporary storage space for the transition to each of the new linear systems.

Formally, we will proceed as follows: given a nonsingular linear system of equations

$$Ax = b, \tag{A.1}$$

where the components of the vector b are numbers between -1 and 1 having αg binary bits in their fixed point representation, we will subdivide each such a component into g segments of α-bit binary numbers and represent b as the

$$b = \sum_{i=1}^{g} b_i$$

where component j of the vector b_i for every j is defined by the ith segment of the respective component j of b. Denote

$$x_i = A^{-1} b_i, \tag{A.2}$$

and let x_i^* be the (hi)-bit approximation to x_i within absolute error 2^{-hi-1} in each component.

It follows that

$$\left\| x - \sum_{i=1}^{g} x_i^* \right\| \leq 2^{-h}.$$

Thus, we reduced the solution of linear system (A.1) within the error norm 2^{-h} to the solution of g linear systems (A.2) within 2^{-hi} for $i = 1, \ldots, g$. The main advantage of this reduction is that for system (A.2) for each i we only need to store the α binary bits of each component of the input vector b_i and h binary bits of each component of the output vector x_i^*, whereas the storage space for the input and output of the original system (A.1) is by g times greater than that.