A shared memory parallel implementation of block-circulant preconditioners *

I. Lirkov† S. Margenov† R. Owens‡ M. Paprzycki§

Abstract

The parallel numerical solution of large scale elliptic boundary value problems is discussed. We analyze the parallel complexity of two block-circulant preconditioners when the conjugate gradient method is used to solve the sparse linear systems arising from such problems. A simple general model of the parallel performance is applied to the considered shared memory parallel architecture. Estimates for the parallel times, the speed-up and the parallel efficiency are derived. The numerical tests have been executed on SGI PC 8 000 and SGI PC 10 000 as well as on Sun Ultra-Enterprise 168 MHz and 250 MHz computers.

Key words: parallel algorithms, PCG method, preconditioner, circulant, performance

AMS subject classifications: 65F10, 65N30

1 Introduction

In this paper we are concerned with algorithms for numerical solution of linear systems arising from the discretization of elliptic boundary value problems. The finite difference method, as well as the finite element method, reduce the continuous problem to a discrete one described by a linear system $Ax = b$, where A is a symmetric positive definite sparse matrix. The projection iterative methods of the conjugate gradient (CG) type are the most cost-efficient way to solve large problems of this class. To accelerate the convergence of the iterative process, the CG methods are almost always used with a preconditioner C. The theory of the preconditioned conjugate gradient (PCG) method leads to the following criteria for constructing preconditioners: to minimize $\kappa(C^{-1}A)$ and to allow efficient computation of the product $C^{-1}v$ for a given vector v.

The present study is focussed on preconditioners based on a block-wise average of the coefficients of A. The first attempts to use this approach have been motivated by the attractive ability

*This research has been initiated by a COBASE grant from the National Research Council. The first two authors were supported in part by Bulgarian NSF Grants MM-417 and I-504. Mr. R. Owens' research was supported by the US NSF through the UT System Alliance for Minority Participation.

†Central Laboratory for Parallel Processing, Bulgarian Academy of Sciences

‡University of Texas of the Permian Basin, Odessa, Texas

§University of Southern Mississippi, Hattiesburg, Mississippi

to exploit the fast inversion of circulant blocks by Fast Fourier Transform (FFT) (see [**?**]). In addition, the recent research on circulant preconditioners for Toeplitz systems [**?**, **?**, **?**] shows promising results for favorable clustering of eigenvalues of the preconditioned system.

We consider two preconditioners that are constructed by circulant approximation of the initial matrix A. The *block-circulant* preconditioner was proposed by R.H.Chan and T.F.Chan in [**?**]. It preserves the block structure of the matrix A and can be written in a tensor product form. The *circulant block-factorization* preconditioner introduced by Lirkov, Margenov and Vassilevski in [**?**] is based on a circulant approximation of the blocks of A (considered in its block-tridiagonal form). The circulant approximation is thus stabilized with respect to varying coefficients along one of the coordinate directions.

The relative condition number of the preconditioned system for a model Poisson problem for both BC and CBF preconditioners is estimated by $O(\sqrt{N})$, where N is the number of unknowns, i.e., the same result as for certain ILU type preconditioners holds. The advantage of the block-circulant preconditioners is their parallel efficiency.

Results of parallel implementation of PCG algorithms with various preconditioners have been already published in a number of papers (see, e.g., in [**?**, **?**, **?**, **?**, **?**]). The implementation and parallel performance characteristics of block circulant preconditioners on distributed memory systems (a network of transputers) was considered in [**?**]. The goal of this paper is to analyze the parallel performance of such preconditioners on shared memory computers. We derive theoretical estimates of the parallel times, the speed-up and the parallel efficiency of the proposed algorithms.

The paper is arranged as follows. The structure of the BC and CBF preconditioners is briefly described in §2. The parallel performance model of the PCG methods with circulant preconditioners is derived in §3 and §4. In §5 we report preliminary experimental results collected on SGI Power Challenge as well as on Sun Ultra-Enterprise shared memory computers. The summary of future research concludes the paper.

2 Circulant preconditioners

As a model problem we consider a 2D second order elliptic PDE in the unit square $\Omega = (0,1) \times (0,1)$ with homogeneous Dirichlet boundary conditions. Let the domain Ω be triangulated by a rectangular uniform grid with n grid points in each coordinate direction. Consider the usual five-point stencil finite difference approximation. This discretization leads to a system of linear algebraic equations $Ax = b$. It is well known that if the grid points are ordered along, e.g., the y-direction first, then the matrix A becomes block-tridiagonal where the diagonal blocks are tridiagonal matrices and the off-diagonal blocks are diagonal matrices. Consequently, the matrix A can be written in the form $A = tridiag(-A_{i,i-1}, A_{i,i}, -A_{i,i+1})$, $i = 1, 2, \ldots, n$.

Let us remind that C is called *circulant* if $(C_{k,j}) = \left(c_{(j-k) \bmod m}\right)$, where C is an $m \times m$ matrix. We denote by $C = (c_0, c_1, \ldots c_{m-1})$ the circulant matrix with first row $(c_0, c_1, \ldots c_{m-1})$. The efficient application of the circulant matrices in the large scale scientific computations is based on the following important property.

For any circulant matrix there exist the following factorization

$$C = F\Lambda F^*. \tag{1}$$

Here Λ is a diagonal matrix containing the eigenvalues of C, and F is the Fourier matrix

$$F = \frac{1}{\sqrt{m}} \left\{ e^{2\pi \frac{jk}{m}\mathbf{i}} \right\}_{0 \le j,k \le m-1},$$

where $\mathbf{i}$ denotes the imaginary unit.

Let us introduce the following notations:

$$\bar{a} = \frac{1}{n^2} \sum_{i=1}^{n-1} \sum_{j=(i-1)n+1}^{in} a_{j,j+n} \qquad \bar{b} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=(i-1)n+1}^{in-1} a_{j,j+1}$$

$$C_0 = (2\bar{a} + 2\bar{b} + 2\rho n^{-2}, -\bar{b}, 0, \ldots 0, -\bar{b}) \qquad C_1 = -\bar{a}I = (-\bar{a}, 0, \ldots 0)$$

where ρ is a positive constant independent of n. Then the *block-circulant preconditioner* C_{BC}, proposed by R. Chan and T. Chan in [**?**], can be written in the form

$$C_{BC} = (C_0, C_1, 0, \ldots 0, C_1). \tag{2}$$

The *circulant block-factorization* preconditioner introduced by Lirkov, Margenov and Vassilevski in [**?**] is defined by

$$C_{CBF} = tridiag(-C_{i,i-1}, C_{i,i}, -C_{i,i+1}) \qquad i = 1, 2, \ldots, n, \tag{3}$$

where $C_{i,j} = Circulant(A_{i,j})$ is a proper circulant approximation of the corresponding block $A_{i,j}$.

3 Model of the computation and communication times

It has been shown that the convergence rate of these preconditioners is asymptotically the same [**?**, **?**]. Based on this fact, to analyze the parallel complexity, we can estimate the parallel execution time for a single PCG iteration only.

To establish the theoretical performance characteristics of the preconditioners in question we apply a simple standard general model for the arithmetic and the communication times [**?**]. We will assume that the computations and communications are not overlapped, and therefore, the parallel execution time is the sum of the computation and communication times. We will also assume that the execution of M arithmetic operations on one processor takes time $T_a = M * t_a$, where t_a is the average unit time to perform one arithmetic operation on one processor (no vectorization). We model the shared memory architecture by assuming that all the processors in the system can be considered as neighbors. Therefore the communication time to transfer M data elements from one processor to another can be approximated by $T_{com} = t_s + M * t_c$, where t_s is the start-up time and t_c is the incremental time necessary for each of M elements to be sent.

The times of the following two global operations will be considered:

- $b(p)$ - *the time for broadcasting a data element from one processor to all others, where there are p processors in the computer system;*
- $g(M,p)$ - *the time for gathering p data packets from all processors into one (there are M/p data elements per packet)*

The estimates of $b(p)$ and $g(M,p)$ represent the capabilities of the parallel architecture. Our assumptions lead to the following simple relations:

$$b(p) = \lceil \log p \rceil (t_s + t_c)$$

$$g(M,p) = (p-1)(t_s + \frac{M}{p} t_c)$$

To achieve a good load balance as well as a low communication cost, we partition the unit square into p rectangles so that each rectangle contains n/p y-grid lines. We map all grid points from a given rectangle onto one processor. In this way, the matrix-vector multiplication requires communication of only n values from one grid line between the corresponding "neighbor processors".

4 Analysis of the parallel complexity

We can now estimate the total execution time T_{PCG} for one PCG iteration for the considered two circulant preconditioners on a shared memory parallel system with p processors. Each iteration consists of one matrix vector multiplication with the matrix A, one multiplication with the inverse of the preconditioner C (solving a system of equation with matrix C), two inner products and three linked triads (a vector updated by a vector multiplied by a scalar). Consequently

$$T_{PCG}(p) = T_{mult} + T_{prec} + 2T_{inn\_prod} + 3T_{triads},$$

where

$$T_{mult} = 9\frac{n^2}{p} t_a + 4(t_s + nt_c),$$

$$T_{inn\_prod} = \left(2\frac{n^2}{p} - 1\right) t_a + \lceil \log p \rceil (t_s + t_c + t_a),$$

$$T_{triads} = 2\frac{n^2}{p} t_a,$$

and where for the BC and CBF preconditioners, respectively

$$T_{prec}^{BC} = 4\frac{n}{p} T_{FFT}(n) + 2\frac{n^2}{p} t_a + 2(p-1)(t_s + \frac{n^2}{p^2} t_c),$$

$$T_{prec}^{CBF} = 2\frac{n}{p} T_{FFT}(n) + 12\frac{n^2}{p} t_a + 2(p-1)(t_s + \frac{n^2}{p^2} t_c),$$

and where $T_{FFT}(n)$ is the time for execution of FFT on a given n-vector on one processor. If the factorization of n in prime numbers is $n = n_1 n_2 \dots n_l$, then $T_{FFT}(n) = 4n(n_1+n_2+\dots+n_l)t_a$. If n is equal to an exact power of two, then we use 2-radix algorithm and then $T_{FFT}(n) = 5n \log n t_a$.

Combining these results we obtain the following estimates of the execution times for the studied preconditioners

$$\begin{aligned} T_{PCG}^{BC}(p) &= 2\left(p + \lceil \log p \rceil + 1\right) t_s + 2\left[(p-1)\frac{n^2}{p^2} + 2n + \lceil \log p \rceil\right] t_c \\ &+ \left(21\frac{n^2}{p} + 2\lceil \log p \rceil - 2\right) t_a + 4\frac{n}{p} T_{FFT}(n) \\ T_{PCG}^{CBF}(p) &= 2\left(p + \lceil \log p \rceil + 1\right) t_s + 2\left[(p-1)\frac{n^2}{p^2} + 2n + \lceil \log p \rceil\right] t_c \\ &+ \left(31\frac{n^2}{p} + 2\lceil \log p \rceil - 2\right) t_a + 2\frac{n}{p} T_{FFT}(n) \end{aligned}$$

For simplicity we assume that the mesh size n is an exact power of two, i.e., $n = 2^l$. Then, the leading terms of the parallel time complexity functions are:

$$T_{PCG}^{BC}(p) \approx 2pt_s + 2(1 - \frac{1}{p})\frac{n^2}{p} t_c + (21 + 20 \log n)\frac{n^2}{p} t_a, \tag{4}$$

$$T_{PCG}^{CBF}(p) \approx 2pt_s + 2(1 - \frac{1}{p})\frac{n^2}{p} t_c + (31 + 10 \log n)\frac{n^2}{p} t_a. \tag{5}$$

It can be observed that for large n $T_{PCG}^{BC}(p) \approx 2T_{PCG}^{CBF}$. Our next goal is to analyze the relative speed-up S_p and the relative efficiency E_p, where

$$S_p = \frac{T(1)}{T(p)} \leq p \quad \text{and} \quad E_p = \frac{S_p}{p} \leq 1.$$

We apply now (**??**–**??**) and obtain:

$$S_p^{BC} \approx \frac{21 + 20 \log n}{2\frac{p^2}{n^2}\frac{t_s}{t_a} + 2(1 - \frac{1}{p})\frac{t_c}{t_a} + 21 + 20 \log n} p, \tag{6}$$

$$S_p^{CBF} \approx \frac{31 + 10 \log n}{2\frac{p^2}{n^2}\frac{t_s}{t_a} + 2(1 - \frac{1}{p})\frac{t_c}{t_a} + 31 + 10 \log n} p. \tag{7}$$

Obviously, for both preconditioners,

$$\lim_{n\to\infty} S_p = p \quad \text{and} \quad \lim_{n\to\infty} E_p = 1,$$

i.e., the algorithms are asymptotically optimal. More precisely, if

$$\log n \gg \frac{p^2}{n^2}\frac{t_s}{t_a} + \frac{t_c}{t_a},$$

then E_p is approximately 1. Unfortunately, the start-up time t_s is usually much larger than t_a, and for relatively small n the first term of the denominators in (**??** - **??**) is significant, in this case the efficiency is much smaller than 1.

5 Experimental results

5.1 Silicon Graphics Power Challenge

In this section we present experimental results collected when running a PVM-based code on SGI Power Challenge 8000 (SGI PC 8000) and SGI Power Challenge 10000 (SGI PC 10000) shared memory computers. Times have been collected using the $mclock$ utility. All data has been collected in benchmarking mode or on lightly loaded systems. In the latter case the best results (out of multiple runs) are reported. The parallel time $T(p)$, the relative speed-up S_p and

Table 1: Parallel time, speed-up and efficiency for the BC preconditioner.

	SGI PC 8 000					SGI PC 10 000				
n	$T(1)$	p	$T(p)$	S_p	E_p	$T(1)$	p	$T(p)$	S_p	E_p
62	0.13	1				0.07	1			
124	0.55	2	0.31	1.77	0.89	0.29	2	0.16	1.81	0.90
186	1.31	3	0.49	2.67	0.89	0.72	3	0.38	1.89	0.63
248	2.29	4	0.67	3.42	0.85	1.21	4	0.51	2.37	0.59
310	3.82	5	1.00	3.82	0.76	2.13	5	0.69	3.09	0.62
372	5.38	6	1.26	4.26	0.71	3.07	6	0.85	3.61	0.60
434	7.97	7	1.62	4.92	0.70	5.30	7	1.33	3.98	0.57
496	9.50	8	1.77	5.37	0.67	5.50	8	1.45	3.79	0.47
558	13.18	9	2.16	6.10	0.68	7.62	9	1.79	4.26	0.47
620	15.88	10	2.41	6.59	0.66	9.22	10	1.92	4.80	0.48
682	21.48	11	2.94	7.31	0.66	12.33	11	2.23	5.53	0.50
744	23.23	12	3.58	6.49	0.54	13.91	12	2.46	5.65	0.47
806	31.53	13	4.41	7.15	0.55	17.93	13	2.69	6.66	0.51
868	36.03	14	4.58	7.87	0.56	22.36	14	2.77	8.07	0.58
930	39.39	15	5.07	7.77	0.52	22.82	15	3.08	7.41	0.49

the relative efficiency E_p for the block-circulant preconditioner BC are given in Table **??**. The similar results for the circulant block-factorization CBF are presented in Table **??**. The mesh-size parameter n and the number of processors p are simultaneously increased following the rule $\frac{n}{p} = 62$. The CBF preconditioner is faster than the BC confirming the theoretical estimates from §4. What is more interesting, the speed-up and the efficiency are also better for the CBF even for these relatively small values of n. The speed-up and the efficiency decrease for the faster machine SGI PC 10000 as a result of its relatively slow communication.

The behavior of the relative efficiency obtained on the SGI PC 8000, varying p for $n = 300$, is shown in Figure **??**. The decrease of E_p with p is clearly visible. Some fluctuations of the reported timings can be observed. It can be explained by the fact that in the reported cases n is not an exact power of two that causes a non-smooth (even non-monotonic) behavior of the computational complexity of the Fourier transforms.

Table 2: Parallel time, speed-up and efficiency for the CBF preconditioner.

	SGI PC 8 000					SGI PC 10 000				
n	$T(1)$	p	$T(p)$	S_p	E_p	$T(1)$	p	$T(p)$	S_p	E_p
62	0.06					0.03				
124	0.28	2	0.16	1.75	0.87	0.15	2	0.09	1.66	0.83
186	0.67	3	0.27	2.48	0.82	0.39	3	0.17	2.29	0.76
248	1.18	4	0.37	3.19	0.78	0.66	4	0.21	3.14	0.78
310	1.98	5	0.49	4.04	0.81	1.16	5	0.30	3.87	0.77
372	2.81	6	0.58	4.84	0.81	1.62	6	0.36	4.50	0.75
434	4.11	7	0.75	5.48	0.78	2.41	7	0.63	3.82	0.55
496	5.00	8	0.89	5.62	0.70	2.78	8	0.74	3.75	0.47
558	6.86	9	1.00	6.86	0.76	4.00	9	0.89	4.49	0.50
620	8.46	10	1.49	5.67	0.57	4.74	10	1.01	4.69	0.43
682	11.22	11	1.50	7.41	0.68	6.36	11	1.16	5.71	0.50
744	11.96	12	1.35	8.86	0.74	7.23	12	1.20	6.02	0.50
806	16.25	13	1.85	8.78	0.68	6.75	13	1.11	6.08	0.47
868	17.02	14	1.74	9.78	0.70	9.17	14	1.40	6.55	0.47
930	19.90	15	1.93	10.31	0.72	11.59	15	1.85	6.26	0.42

5.2 SUN Ultra-Enterprise Symmetric Multiprocessor

In this section we report the results of the experiments executed on SUN Ultra-Enterprise Symmetric Multiprocessor. The code has been implemented using the MPI (Message-Passing Interface) library. Based on the results from the previous section (and similar results presented in [**?**]), indicating the superior performance of the Circulant Block-Factorization, only the results for this preconditioner are reported. The parallel time, the relative parallel speed-up and the relative efficiency obtained on systems consisting of 168 MHz and 250 MHz processors for the mesh-size $n \in \{128, 256, 384, 420\}$ are reported in Table **??**. The general behavior is in a good agreement with the theoretical estimates. We can observe that, similarly to the results from the previous subsection, the efficiency is higher on the computer with the slower processors. In the case of the faster processor the speed of the communication network is clearly inadequate. Comparing the results obtained using PVM and MPI as the communication libraries it can be conjectured that the MPI environment provides a better performance for the global communication operations.

6 Concluding remarks

The parallel implementation of two block-circulant preconditioners on a shared memory parallel system was studied. Even though the convergence of the BC and the CBF preconditioners is of the same order the results presented in [**?**, **?**] indicate that, for problems with varying coefficients or anisotropy, as well as for y-periodic problems, the CBF preconditioner has considerable ad-

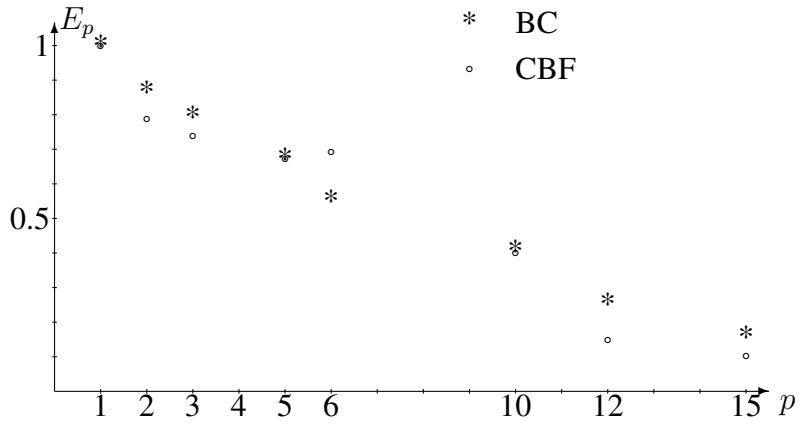

Figure 1: Parallel relative efficiency obtained on SGI PC 8000 for $n = 300$.

vantages. In addition, the results presented in this paper indicate that the parallel performance characteristics are also better for the CBF preconditioner.

The experimental data collected so far is only preliminary. In the near future we plan to expand our investigations in the following directions:

- Study the performance characteristics of the proposed preconditioners on the dynamic shared memory machines (e.g. SGI Origin and Convex Exemplar).
- For the shared and dynamic shared memory machines compare the performance of the message passing and the shared memory approaches to parallelization.
- Comparison of the PVM and MPI based implementations of the proposed preconditioners on the shared, distributed and dynamic shared memory machines.
- Develop a more realistic theoretical models of parallel performance.
- Extend the study into 3D elliptic problems.

Acknowledgments

The computer time grants from the NCSA in Urbana and the University of Nijmegen are kindly acknowledged.

References

[1] O. Axelsson, V.A. Barker, *Finite Element Solution of Boundary Value Problems: Theory and Computation* (Academic Press, Orlando, Fl., 1983).

Table 3: Parallel time, speed-up and efficiency for the CBF preconditioner on a SUN Ultra-Enterprise symmetric multiprocessor.

		168 MHz			250 MHz		
n	p	$T(p)$	S_p	E_p	$T(p)$	S_p	E_p
128	1	0.086			0.081		
	2	0.047	1.84	0.92	0.047	1.71	0.86
	4	0.028	3.04	0.76	0.029	2.77	0.69
	8	0.021	4.13	0.52	0.096	0.84	0.10
256	1	0.389			0.392		
	2	0.207	1.88	0.94	0.208	1.88	0.94
	4	0.109	3.56	0.89	0.127	3.09	0.77
	8	0.065	6.02	0.75	0.138	2.83	0.35
384	1	1.460			1.498		
	2	0.759	1.92	0.96	0.783	1.91	0.96
	3	0.523	2.79	0.93	0.533	2.81	0.94
	4	0.394	3.71	0.93	0.473	3.17	0.79
	6	0.269	5.43	0.90	0.780	1.92	0.32
	8	0.338	4.32	0.54	1.122	1.33	0.17
420	1	3.718			2.651		
	2	1.922	1.93	0.97	1.378	1.92	0.96
	3	1.313	2.83	0.94	0.937	2.83	0.94
	4	0.990	3.75	0.94	0.714	3.71	0.93
	5	0.817	4.55	0.91	1.005	2.64	0.53
	6	0.679	5.48	0.91	1.233	2.15	0.36
	7	0.595	6.25	0.89	1.314	2.02	0.29

[2] R.H. Chan, T.F. Chan, Circulant preconditioners for elliptic problems, *J. Numerical Lin.Alg.Appl.* **1** (Mar. 1992) 77–101.

[3] R. Chan, G. Strang, Toeplitz equations by conjugate gradients with circulant preconditioner, *SIAM J.Sci.Stat.Comp.*, **10** (1989) 104–119.

[4] P.J. Davis, *Circulant matrices*, John Wiley, New York (1979).

[5] C. Douglas, Parallel multilevel and multigrid methods, *Research report RC 17781 (#78306) 3/9/92, IBM Research Division*, T.J. Watson Research Center, Yorktown Heights, NY 10598.

[6] G.H. Golub, C.F. Van Loan, *Matrix computations*, 2nd edition (John Hopkins Univ.Press, Baltimore, 1989).

[7] T. Huckle, Circulant and skewcirculant matrices for solving Toeplitz matrix problems, *SIAM J.Matr.Anal.Appl.*, **13** (1992), 767–777.

[8] T. Huckle, Some aspects of circulant preconditioners, *SIAM J.Sci. Comput.*, **14** (1993), 531–541.

[9] I. Lirkov, S. Margenov, Parallel complexity of conjugate gradient method with circulant preconditioners, *in Proceedings of the PARCELLA'96*, (R. Vollmar, W. Erhard, V. Jossifov, eds.), Akademie Verlag, Berlin, (1996), 279–286.

[10] I. Lirkov, S. Margenov, P.S. Vassilevski, Circulant block-factorization preconditioners for elliptic problems, *Computing*, **53** 1 (1994), 59–74.

[11] I. Lirkov, S. Margenov, L. Zikatanov, Circulant Block-Factorization Preconditioning of Anisotropic Elliptic Problems, UCLA CAM Report **95–39**, 1995.

[12] Charles Van Loan, *Computational frameworks for Fast Fourier Transform*, (SIAM, Philadelphia, 1992).

[13] M. Neytcheva, *Arithmetic and communication complexity of preconditioning methods*, Ph.D. Thesis, KUN, The Netherlands, 1995.

[14] M. Pester, S. Rjasanow, A Parallel Version of the Preconditioned Conjugate Gradient Method for Boundary Element Equations, *Numer. Lin. Alg. with Appl.*, **2** 1 (1995), 1–16.

[15] Y. Saad, M.H. Schultz, Data Communication in Parallel Architectures, *Parallel Comput.*, **11** (1989), 131–150.

[16] B.F. Smith, A parallel implementation of an iterative substructuring algorithm for problems in three dimensions, *SIAM J.Sci.Stat.Comp.*, **14** (1993), 406–423.

[17] Ch. Tong, The preconditioned conjugate gradient method on the connection machine, *Int. J. High Speed Comp.*, **2** (1989), 263–288.