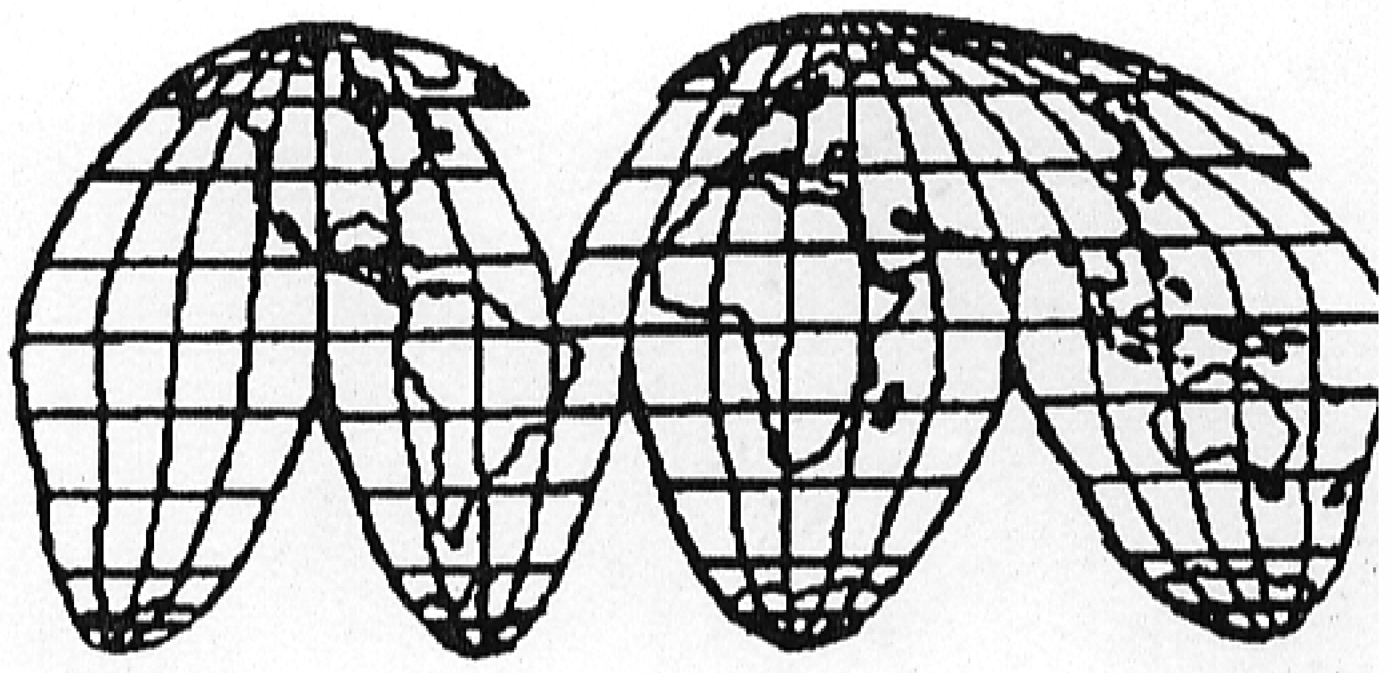
INVITED LECTURES and SHORT COMMUNICATIONS

DELIVERED AT THE SIXTH INTERNATIONAL COLLOQUIUM ON DIFFERENTIAL EQUATIONS

August 18-23, 1995, Plovdiv, Bulgaria



Volume II

Editor: Angel Dishliev

Impulse - *M*

HIGH PERFORMANCE SOLUTION OF LINEAR SYSTEMS ARISING IN SPECTRAL DOMAIN DECOMPOSITION TECHNIQUES

MARCIN PAPRZYCKI
CLIFF CYPHERS
Department of Mathematics and Computer Science
University of Texas of the Permian Basin
Odessa, TX 79762
USA

ANDREAS KARAGEORGHIS
Department of Mathematics and Statistics
University of Cyprus
Nicosia
CYPRUS

ABSTRACT
When spectral collocation methods are applied to problems in rectangularly decomposable domains they lead to the solution of a structured linear system. Since the linear system needs to be solved at each step of a Newton-type iterative process, an efficient method to do so needs to be used. Two level 3 BLAS based algorithms are presented for the solution of linear systems arising from discretizations of a re-entrant tube flow problem and Poisson problems in two and three dimensions. The efficiency of the proposed algorithms is studied.

KEYWORDS: Domain Decomposition, Spectral Methods, Collocation, Structured Linear Systems, Capacitance Matrix Techniques

1. INTRODUCTION

Spectral collocation methods are often applied to the solution of partial differential equations on rectangular domains [1]. These methods require the repeated solution of structured linear systems, which becomes the most costly part of the solution process. These linear systems are either almost block diagonal (ABD) or augmented ABD where a number of additional blocks appear regularly, adding complexity to the system (see e.g. [1, 2]). In all cases the linear systems are composed of relatively few large blocks. As was shown in [3] ABD systems arising from discretizations of two-point boundary value problems can be solved efficiently using level 3 BLAS [4] kernels. The aim of this paper is two-fold. First, we will show that this method can also be applied efficiently to the solution of a channel flow problem discretized using a Chebyshev spectral collocation. The results of experiments performed on a Cray J916 will be presented and discussed. Second, we will extend the BLAS based approach to certain domain decompositions of rectangular (and cuboidal) domains for Poisson problems. Here a capacitance technique described in [5] will be applied. The performance on an RS6000 workstation and a Cray Y-MP will be presented and discussed. We will also compare the performance of the proposed solver with that of a general sparse solver from NAG.

2. THE RE-ENTRANT TUBE FLOW PROBLEM

We consider the flow of an incompressible fluid through a re-entrant tube (for details, see [1]). The flow is assumed to be steady and governed by the stream formulation of the Navier-Stokes equations. As in [1] the flow region is divided into four elements and in region i the solution is approximated by

(2.1)
$$\psi^i(x,y) = g^i(y) + \sum_{m=2}^{M_i}\sum_{n=2}^{N_i} a^i_{mn}\,\bar{T}^i_m(x)\,\hat{T}^i_n(y)$$

where $\bar{T}_m$ and $\hat{T}_n$ are linear combinations of Chebyshev polynomials chosen so that appropriate boundary conditions are satisfied identically on the boundary. The functions $g^i(y)$ are Poiseuille stream functions corresponding to each element. Careful selection of the number of collocation points ensures that the approximation is C^1 continuous everywhere on these interfaces [1]. The unknown coefficients a^i_{mn} in (2.1) may be found at each iteration by solving the linear system which is of the form shown in Figure 1.

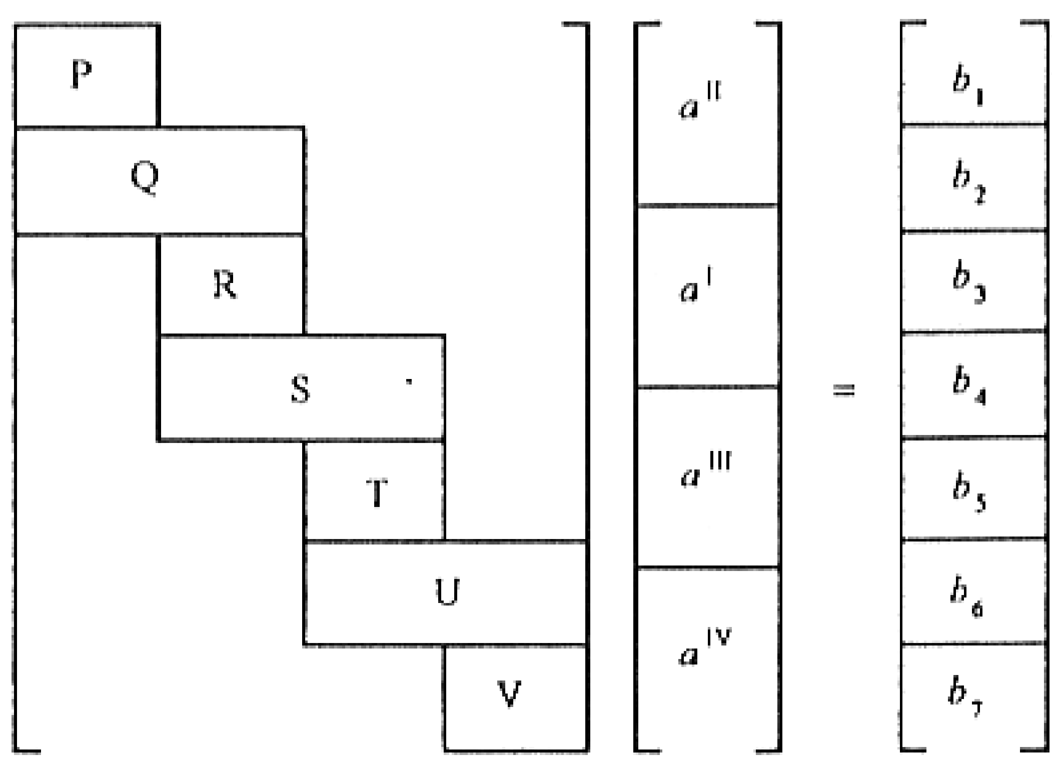


Figure 1. The linear system resulting from the discretization.

The blocks P, R, T and V result from the collocation of the governing equation and boundary conditions whereas Q, S and U result from the imposition of the interelement continuity conditions. The structure of the system is independent of the type of boundary conditions of the problem. The transpose of this system has a generalized ABD form and can be solved using a block form of the alternate row and column elimination algorithm of Varah [6] as implemented using level 3 BLAS in [7]. Each step of the algorithm consists of two phases based on an extended (9-block) block Gaussian elimination. First a rectangular block (in case of the linear system in Figure 1 a row block in P^T) is factorized using Gaussian elimination with row pivoting. If the size of the block to be factorized is larger than the optimal blocksize (of a given computer), the decomposition is performed in a blocked fashion [8]. After this factorization the appropriate parts of the linear system (inside blocks P^T and Q^T) are updated. In the second phase, the next block (in case of the system in Figure 1 a column block of Q^T) is factorized using Gaussian elimination with column pivoting and the appropriate parts Q^T are updated. This process is then repeated. The factorization is performed using the LAPACK provided routine SGETRF (see [8] for more details). The update steps consist of calls to the level 3 BLAS routines STRSM and SGEMM.

3. NUMERICAL EXPERIMENTS

In the 4-block ABD system described in Section 2 the size of the first block depends on d (the size of the re-entrant lip). We experimented with four different values of d = 0.2, 0.5, 1.0 and 1.5 leading to first block of sizes 380×414, 475×509 646×680 and 684×718 respectively. The total system size varies between 1547x1547 and 1851x1851. The experiments were performed using the ABD solver implementation described in [7] and its performance was compared to the F01LHF/F04LHF (decomposition/back substitution) pair from NAG [9]. We performed experiments on a Cray J916 and the Cray provided optimized BLAS kernels were used in both the new code and the NAG routines. The *perftrace* utility was used to measure the performance. Each result presented here is an average of multiple runs. Table 1 presents the results for Re = 5 and Re = 400 (Re is the Reynolds number), for varying d (the optimal blocksize 128 was used inside the decomposition step).

It can be observed that the value of Re has a minimal effect for the MFlop rate of the solvers. The performance gains from the use of the new code are 16-35 MFlops (between 14% and 25%) and grow slightly as the size of the system increases. Since the practical peak performance of a single processor of the Cray J916 is approximately 194 MFlops [10] for the large system the performance is approximately 69% of the practical peak.

Table 1. Solution of the channel problem; one processor; results in MFlops.

	Re = 5		*Re* = 400	
d	New code	NAG	New code	NAG
0.2	116.0	92.6	116.2	100.9
0.5	117.1	91.2	118.4	101.8
1.0	134.3	102.2	134.1	113.2
1.5	135.1	101.7	143.7	116.0

4. FULLY CONFORMING DECOMPOSITIONS

In [2] a fully conforming spectral domain decomposition collocation scheme was presented for second order problems in two and three dimensions. This scheme ensures that the approximations are C^1 pointwise continuous across the subdomain interfaces for second order problems. Unlike the problem examined in Section 3 the matrices arising in this case do not possess a simple ABD structure. They are, however, sparse and structured and a capacitance-type matrix technique [5] can be applied to exploit the block structure.

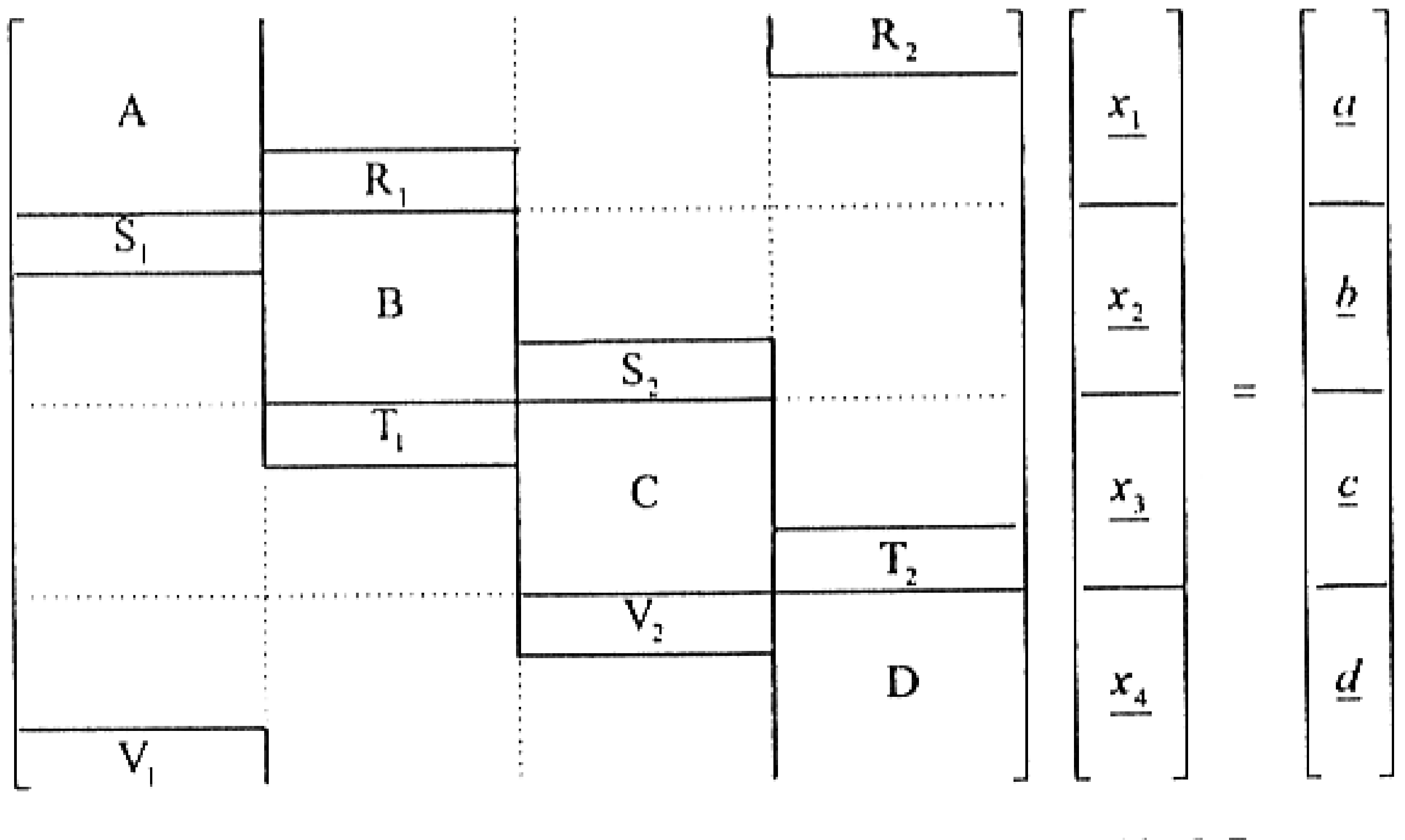


Figure 2. The linear system of the fully conforming method in 2-D.

For the four element decomposition of [2] of the rectangle $(\alpha,\beta)\times(a,b)$ into the subdomains $(\alpha, h_x)\times(h_y, b)$ (Region I), $(\alpha, h_x)\times(a, h_y)$ (Region II), $(h_x, \beta)\times(a, h_y)$ (Region III) and $(h_x, \beta)\times(h_y, b)$ (Region IV), $\alpha < h_x < \beta$, $a < h_y < b$, we used the spectral approximations

(4.1) $$\phi^s(x,y) = \sum_{m=0}^{M_s}\sum_{n=0}^{N_s} x^s_{mn}\overline{T}^s_m(y)\hat{T}^s_n(x) \qquad s = \text{I, II, III or IV}$$

where the polynomials $\overline{T}^s_m(y)$ and $\hat{T}^s_n(x)$ are appropriately chosen shifted Chebyshev polynomials. The linear system (4.2) has the form of Figure 2.

The matrices *A*, *B*, *C* and *D* represent the boundary conditions, the governing equations and the interface conditions in each region. The vectors $\underline{x}_1$, $\underline{x}_2$, $\underline{x}_3$ and $\underline{x}_4$ contain the unknown coefficients in regions I, II, III and IV. The matrices R_1, R_2, S_1, S_2, T_1, T_2, V_1 and V_2 correspond to the interface conditions between respective regions. We can rewrite the global system (4.2) as:

(4.3.a) $$A\,\underline{x}_1 + R_1^*\,\underline{x}_2 + R_2^*\,\underline{x}_4 = \underline{a}$$

(4.3.b) $$S_1^*\,\underline{x}_1 + B\,\underline{x}_2 + S_2^*\,\underline{x}_3 = \underline{b}$$

(4.3.c) $$T_1^*\,\underline{x}_2 + C\,\underline{x}_3 + T_2^*\,\underline{x}_4 = \underline{c}$$

(4.3.d) $$V_1^*\,\underline{x}_1 + V_2^*\,\underline{x}_3 + D\,\underline{x}_4 = \underline{d}$$

Where the star represents the fact that a matrix was augmented by an appropriate block of zeros (e.g. $R_1^* = \begin{bmatrix} 0 \\ R_1 \end{bmatrix}$). From (4.3.a) and (4.3.d) we can represent $\underline{x}_1$ *and* $\underline{x}_3$ in terms of $\underline{x}_2$ and $\underline{x}_4$

(4.4.a) $$\underline{x}_1 = A^{-1}\,\underline{a} - A^{-1}\,R_1^*\,\underline{x}_2 - A^{-1}\,R_2^*\,\underline{x}_4$$

(4.4.b) $$\underline{x}_3 = C^{-1}\,\underline{c} - C^{-1}\,T_1^*\,\underline{x}_2 - C^{-1}\,T_2^*\,\underline{x}_4$$

After substitution into (4.3.b) and (4.3.d), a smaller system can be solved for $\underline{x}_2$ and $\underline{x}_4$ and then the vectors for $\underline{x}_1$ and $\underline{x}_3$ are calculated from (4.4.a) and (4.4.b). In the case of the 3-D problem the linear systems have a more complicated structure, but a similar solution technique can be applied (for the details see [2]). The proposed solution method is rich in matrix-matrix operations, thus naturally leading to a BLAS based implementation.

5. EXPERIMENTAL RESULTS

The experiments were performed on a RS 6000-530 workstation and on a Cray Y-MP supercomputer. Timings on the workstation were obtained on an empty machine using the *time* function. Timings on the Cray were obtained using the *perftrace* utility. The manufacturer provided optimized BLAS kernels were used. Each result is an average of multiple runs. In two dimensions we solved the Poisson equation on a unit square

(5.1) $$\nabla^2 \phi(x, y) = (y^2 - 1)\, e^x + (x^2 - 1)\, e^y + 2e^x + 2e^y$$

subject to Dirichlet boundary conditions. We took $h_x = 0.1$ and $h_y = 0.2$. In three dimensions we solved the Poisson equation on a unit cube

(5.2) $$\nabla^2 \phi(x, y, z) = (y^2 - 1)(x^2 - 1)\, e^x + (x^2 - 1)(z^2 - 1)e^y + (x^2 - 1)(y^2 - 1)e^z + 2((z^2 - 1)e^y + (y^2 - 1)e^z + (z^2 - 1)e^x + (x^2 - 1)e^z + (y^2 - 1)e^x + (x^2 - 1)e^y)$$

subject to Dirichlet boundary conditions. We took $h_x = 0.1$, $h_y = 0.2$ and $h_z = 0.3$ (corresponding to h_x and h_y in three dimensions).

5.1. Performance of the proposed algorithm

In Table 2 the timings for the 2-D problem on both computers for $n = 4, 5, \ldots, 14$ are presented (where n is the order of the highest Chebyshev polynomial used in the approximation in each direction in each subdomain). The ratio of execution time of the solution of the full system to that of the capacitance method is also shown.

Table 2. Fully conforming method; 2-D performance, results in seconds.

n	Matrix size	RS6000			Cray Y-MP		
		Full system	Capacitance technique	Ratio	Full system	Capacitance technique	Ratio
4	100	0.24	0.12	2.00	0.0348	0.0227	1.53
5	144	0.48	0.22	2.18	0.0687	0.0431	1.59
6	196	1.09	0.38	2.86	0.145	0.0539	2.69
7	264	1.72	0.52	3.30	0.242	0.0860	2.81
8	324	2.93	0.84	3.48	0.365	0.115	3.17
9	400	4.83	1.36	3.55	0.564	0.198	2.84
10	484	7.76	2.18	3.55	0.824	0.230	3.58
11	576	11.33	3.03	3.73	1.27	0.420	3.02
12	676	16.18	4.71	3.43	1.73	0.531	3.25
13	784	22.54	6.65	3.38	2.49	0.736	3.38
14	900	38.34	10.60	3.61	3.38	0.986	3.42

The capacitance technique is clearly superior to the solution of the full system. The gain from its usage increases as the size of the system increases (the time ratio increases from 2 to 3.61 on the workstation and from 1.53 to 3.42 on the Cray). On the Cray, for n=14 the full system technique was running at approximately 170 MFlops whereas the capacitance technique achieved about 150 MFlops (assuming that the practical peak performance of the one-processor Cray Y-MP is about 315 MFlops [10], the capacitance technique reaches about 48% efficiency).

Table 3 presents the results of experiments with the 3-D problem on both computers, for m = 4, 5, 6 (where m corresponds to n in 3-D) for the workstation and for m = 4, 5, ... , 8 for the Cray (lack of results means that the system of a given size did not fit into the computer's memory). In addition the ratio of execution times of solving the full system to that of the capacitance technique is presented.

Table 3. Fully conforming method; 3D performance, results in seconds.

		RS6000			Cray Y-MP		
m	Matrix size	Full system	Capacitance technique	Ratio	Full system	Capacitance technique	Ratio
4	1000	29.1	7.7	3.76	3.8	1.2	3.15
5	1728	145.6	31.6	4.60	18.2	4.8	3.81
6	2744		101.1		60.5	15.0	4.03
7	4096				188.0	47.1	3.99
8	5832					120.0	

The gains from using the capacitance technique are even more apparent for the three dimensional case where the sizes of the dense systems are much larger and grow faster as m increases. The time ratio reaches 4.6 for the workstation and 3.99 for the Cray. The new algorithm is not only superior as far as the execution time is concerned, but also its memory requirements are much smaller. For m=7 the memory requirements when the capacitance technique was used were approximately 19.0 MW of Cray's memory. At the same time the full system did not fit into the 32.0 MW memory system (here the matrix size is 4096×4096). To perform experiments with the largest systems reported, a 64.0 MW memory Cray was used. It should be noted that the full matrix technique is characterized by almost the same MFlop rate as the capacitance technique (e.g. for m=7 the dense solver reaches 253 MFlops whereas the capacitance technique attains 246 MFlops). This can be explained by the fact that even though the dense solver uses longer vectors and a smaller number of subroutine calls, the sizes of the individual blocks (varying from 64×512 to 512×512) are large enough for the level 3 BLAS kernels to reach high efficiency. For m=8 the capacitance technique is executing at about 273 MFlops so it reaches about 86% of the practical peak performance.

5.2 Sparse solver performance

We also experimented with the application of a general sparse matrix solver. For this purpose we used the F01BRF and F04AXF pair from the NAG Library [9]. These routines are based on the MA28 package from the Harwell Library and attempt to permute the system to the lower triangular form [11]. F01BRF provides a user controllable parameter (PIVOT), a relative measure used to control the pivot strategy: the smaller its value the more F01BRF maintains sparsity at the expense of stability. Table 4 summarizes the results of the experiments performed on the RS6000 workstation and the Cray Y-MP for both the 2-D problem (for n = 4, 5, ..., 14) and for the 3-D problem (for m = 4, 5). We experimented with $PIVOT = 10^{-1}, 10^{-2}, ..., 10^{-6}$ while only the results for $PIVOT = 10^{1\cdot}$ (the default value suggested in the documentation) and $PIVOT = 10^{-6}$ (the last value for which at least some results remained meaningful) are presented. It should be noted that in most cases the breakdown in accuracy occurred as early as for $PIVOT = 10^{-4}$. The results are reported in seconds. The results for the dense solver are provided for comparison purposes and the ratio calculated is between the values obtained for $PIVOT = 10^{-1}$ and the dense solver.

Table 4. Comparison with the general sparse solver; 2-D and 3-D, results in seconds.

	2–D							
	RS6000				Cray Y-MP			
n	$U = 10^{-1}$	$U = 10^{-6}$	Dense Solver	Ratio	$U = 10^{-1}$	$U = 10^{-6}$	Dense Solver	Ratio
4	0.06	0.12	0.24	0.50	0.09	0.09	0.03	3.00
5	0.13	0.12	0.48	1.08	0.25	0.24	0.06	4.17
6	0.34	0.37	1.09	0.31	0.47	0.45	0.14	3.35
7	0.75	0.67	1.72	0.44	1.19	1.06	0.24	4.95
8	2.44	1.60	2.93	0.86	1.95	1.82	0.36	5.41
9	16.02	5.99	4.83	3.31	5.06	3.78	0.56	9.03
10	18.80	9.31	7.76	2.02	6.21	6.82	0.82	7.53
11	46.83	20.68	11.33	4.13	13.80	10.60	1.27	10.86
12	20.00	22.90	16.28	1.78	13.40	13.60	1.73	7.75
13	161.10	55.72	22.54	6.70	35.10	25.00	2.49	14.09
14	100.90	63.32	38.34	2.87	33.60	33.70	3.38	9.94
	3–D							
4	65.09	28.82	29.10	2.24	21.90	17.00	3.80	5.76
5	1009.56	228.70	145.60	7.35	181.00	115.00	18.20	9.94

A number of observations can be made. In the 2-D case on the RS6000 workstation, the general sparse solver outperforms the capacitance technique only for very small systems (up to n=5) and the dense solver only for minimally larger systems (up to n=7). There is a number of factors that can explain this behavior. For small *n* the block sizes are small enough for the structured sparse matrix to resemble a general sparse matrix. The whole system is small enough for the permutations to be very efficient and at the same time there is no special advantage from using BLAS-based algorithms. In the remaining 2-D cases as well as all the 3-D cases the general sparse solver is up to seven times slower. In this case the sparse solver is not able to successfully cope with the special structure of the linear system. Furthermore, the matrices are large enough for the BLAS-based algorithms to take full advantage of the underlying architecture.

The situation is even clearer on the Cray where the sparse solver neither outperforms the capacitance technique nor does it outperform the dense solver. In this case all the advantages of the Cray vector processor are lost. The processing speed drops to 2-10 MFlops for the 2-D system and to 8-14 MFlops for the 3-D system. For the 2-D problem, the solution times for the Cray are comparable to (or worse than) these obtained on the RS6000.

One should also observe that the matrices are relatively dense. In the 2-D case the total number of non-zero elements is approximately 25%. In the 3-D case the total number of non-zero elements is approximately 12%. As a general rule the sparse solver will only start to become competitive with dense solvers when there are less than 5% non-zero elements [12]. The number of non-zero elements is independent of the values of *n* or *m*, and as a result increasing these values does not lead to a sparser system. The oscillation in the computational time of the sparse solver can be related to the number of permutations performed and thus the size of the fill-in generated. The same oscillation pattern occurs for both the RS6000 and the Cray.

Finally, the reduction in the value of *PIVOT* clearly reduces the time required by the sparse solver but, as previously mentioned, it affects greatly the accuracy of results. For large enough matrices the decrease in solution time was a linear function of *PIVOT*. At the same time, only for the 2-D case on the RS6000 for relatively small values of *n* the reduction in *PIVOT* led to the performance improvement significant enough for the sparse solver to outperform the dense solver.

ACKNOWLEDGMENTS

The computer time grant from the Center for High Performance Computing in Austin is kindly acknowledged.

REFERENCES

1. A. Karageorghis, Comp. Meth. in Appl. Mech. and Eng., 100, pp. 339-358 (1992).

2. A. Karageorghis, Technical Report R94007, Laboratoire d'Analyse Numerique, Universite Pierre et Marie Curie, Paris (1994).

3. M. Paprzycki and I. Gladwell, Proceedings of The Fifth SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, (1992) pp. 52-62.

4. J. Dongarra, J. Du Croz and S. Hammarling, Technical Report ANL-MCS-TM57, Argonne National Laboratory (1988).

5. B. Buzbee, F. Dorr, J. George and G. Golub, SIAM J. Numer. Anal., 8, pp. 722-736

6. J. Varah, SIAM J. Numer. Anal., 13, pp. 71-75 (1976).

7. C. Cyphers, M. Paprzycki and I. Gladwell, Software Report 92-3, Southern Methodist University (1992).

8. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen, LAPACK Users' Guide, SIAM, Philadelphia (1993).

9. Numerical Algorithms Group Library (Mark 15), NAG Ltd., Oxford (1988).

10. M. Paprzycki and C. Cyphers, CHPC Newsletter, 6, pp. 77-82 (1991).

11. I. Duff, A. Erisman, and J. Reid, Direct Methods for Sparse Matrices, Clarendon Press, Oxford, (1990).

12. R. Brankin, NAG Ltd., Private communication.