Numerical Algorithms 12(1996)309–319

An efficient direct method for fully conforming spectral collocation schemes

Andreas Karageorghis

Department of Mathematics and Statistics, University of Cyprus, Nicosia 1678, Cyprus

Marcin Paprzycki

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

Received 20 December 1994; revised 17 November 1995
Communicated by J. Dongarra

An efficient direct method is presented for the solution of the linear systems arising in the solution of second and fourth order problems by certain multidomain spectral collocation schemes. In it, the block structure of the global matrix is exploited. The performance of the method is examined for problems in two and three dimensions on an RS6000 workstation and a Cray J-916 supercomputer.

Keywords: Domain decomposition, spectral collocation, capacitance techniques, high performance computing.

Subject classification: 65N35, 65F05, 65N22.

1. Introduction

In [1] a fully conforming spectral domain decomposition collocation scheme was presented for second and fourth 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 and C^3 pointwise continuous across the subdomain interfaces for the fourth order problems. The aim of the present study is to suggest an efficient method for solving the resulting systems of linear equations. Unlike the problems examined in [2] and [3] the matrices arising in the present study do not possess an almost block diagonal structure. They are, however, sparse and structured and by applying a capacitance-type matrix technique [4] the existing block structure may be exploited.

2. The method

2.1. *Two-dimensional case*

For the four-element decomposition [1] 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

$$\phi^s(x, y) = \sum_{m=0}^{M_s} \sum_{n=0}^{N_s} x_{mn}^s \hat{T}_m^s(x) \tilde{T}_n^s(y), \qquad s = I, II, III \text{ or } IV, \tag{1}$$

where the polynomials $\hat{T}_m^s(x)$ and $\tilde{T}_n^s(y)$ are appropriately chosen shifted Chebyshev polynomials. The above approximations were used to approximate the solution of either a second order problem or a fourth order problem. The global system (2) has the form given in figure 1.

If we take $k_s = (M_s + 1)(N_s + 1), s = I, II, III$ or IV, then clearly the global matrix has dimension $(k_I + k_{II} + k_{III} + k_{IV})^2$. 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, respectively. The matrices A, B, C, and D (which are dense) have dimensions k_I, k_{II}, k_{III} and k_{IV} respectively and their rows correspond to the satisfaction of the boundary conditions, the governing equation and interface conditions in each region. The matrices $R_1, R_2, S_1, S_2, T_1, T_2, V_1$, and V_2 (which are dense) correspond to the interface conditions $I - II, I - IV, I - II, II - III, II - III, III - IV, I - IV$ and $III - IV$, respectively, and have the following dimensions: R_1 is $m_1 \times k_{II}$, R_2 is $n_1 \times k_{IV}$, S_1 is $m_2 \times k_I$, S_2 is $n_2 \times k_{III}$, T_1 is $m_3 \times k_{II}$, T_2 is $n_3 \times k_{IV}$, V_1 is $m_4 \times k_I$, V_2 is $n_4 \times k_{III}$. For example, in the case of second order problems $m_1 = M_I$, $n_1 = N_I - 1, m_2 = M_{II} - 1, n_2 = N_{II}, m_3 = N_{III} - 1, n_3 = M_{III}, m_4 = N_{IV}$ and $n_4 = M_{IV} - 1$.

$$\begin{bmatrix} A & & & R_1 & & & R_2 \\ S_1 & & & B & & & \\ & & & S_2 & & & \\ & & & T_1 & & & \\ & & & C & & & T_2 \\ & & & & & & V_2 \\ V_1 & & & & & & D \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \underline{x}_3 \\ \underline{x}_4 \end{bmatrix} = \begin{bmatrix} \underline{a} \\ \underline{b} \\ \underline{c} \\ \underline{d} \end{bmatrix} \tag{2}$$

Figure 1. Structure of the global system for the 2-D case.

By denoting $R_1^* = [0/R_1]$, $R_2^* = [R_2/0]$, $S_1^* = [S_1/0]$, $S_2^* = [0/S_2]$, $T_1^* = [T_1/0]$, $T_2^* = [0/T_2]$, $V_1^* = [0/V_1]$, $V_2^* = [V_2/0]$, the global system (2) may be rewritten in component form:

$$A\underline{x}_1 + R_1^*\underline{x}_2 + R_2^*\underline{x}_4 = \underline{a}, \tag{3a}$$

$$S_1^*\underline{x}_1 + B\underline{x}_2 + S_2^*\underline{x}_3 = \underline{b}, \tag{3b}$$

$$T_1^*\underline{x}_2 + C\underline{x}_3 + T_2^*\underline{x}_4 = \underline{c}, \tag{3c}$$

$$V_1^*\underline{x}_1 + V_2^*\underline{x}_3 + D\underline{x}_4 = \underline{d}. \tag{3d}$$

From (3a) and (3c) we express $\underline{x}_1$ and $\underline{x}_3$ in terms of $\underline{x}_2$ and $\underline{x}_4$ (as these equations are independent of each other). Subsequent substitution of these expressions into (3b) and (3d) leads to the Schur complement system for the vectors $\underline{x}_2$ and $\underline{x}_4$. Once $\underline{x}_2$ and $\underline{x}_4$ have been calculated, $\underline{x}_1$ and $\underline{x}_3$ can be easily obtained from (3a) and (3c).

2.2. *Three-dimensional case*

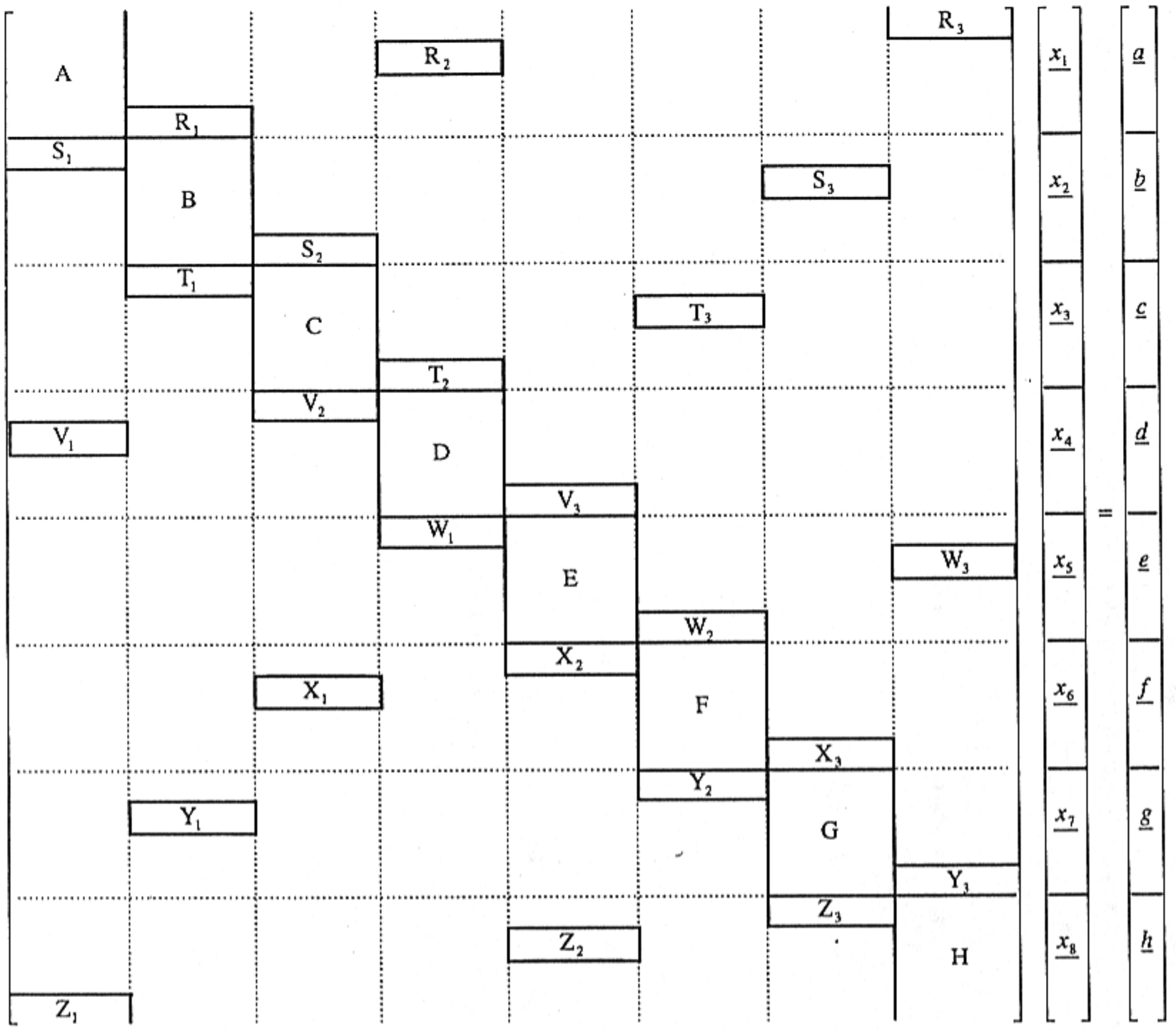
From [1], the cuboid $(\alpha, \beta) \times (a, b) \times (A, B)$ is decomposed into the eight cuboidal subdomains $(\alpha, h_x) \times (a, h_y) \times (h_z, B)$ (Region I), $(\alpha, h_x) \times (a, h_y) \times (A, h_z)$ (Region II), $(h_x, \beta) \times (a, h_y) \times (A, h_z)$ (Region III), $(h_x, \beta) \times (a, h_y) \times (h_z, B)$ (Region IV), $(h_x, \beta) \times (h_y, b) \times (h_z, B)$ (Region V), $(h_x, \beta) \times (h_y, b) \times (A, h_z)$ (Region VI), $(\alpha, h_x) \times (h_y, b) \times (A, h_z)$ (Region VII), $(\alpha, h_x) \times (h_y, b) \times (h_z, B)$, (Region VIII), where $\alpha < h_x < \beta, a < h_y < b, A < h_z < B$. In each, the solution is approximated by

$$\phi^s(x, y, z) = \sum_{m=0}^{M_s} \sum_{n=0}^{N_s} \sum_{l=0}^{L_s} x_{mnl}^s \hat{T}_m^s(x) \tilde{T}_n^s(y) \bar{T}_l^s(z),$$

$$s = I, II, III, IV, V, VI, VII \text{ or } VIII, \tag{4}$$

where the polynomials $\hat{T}_m^s(x)$, $\tilde{T}_n^s(y)$, $\bar{T}_l^s(z)$ are appropriately chosen shifted Chebyshev polynomials. As in the two-dimensional case, the above approximations were used to approximate the solution of either a second or a fourth order problem. The global system (5) is of the form given in figure 2.

As before, we take $k_s = (M_s + 1)(N_s + 1)(L_s + 1)$, $s = I, II, III, IV, V, VI, VII$ or $VIII$. The global matrix has dimension $\sum_{s=I}^{VIII} k_s \times \sum_{s=I}^{VIII} k_s$. The vectors $\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4, \underline{x}_5, \underline{x}_6, \underline{x}_7$ and $\underline{x}_8$ are vectors containing the unknown coefficients in regions $I, II, III, IV, V, VI, VII$ and $VIII$, respectively. The matrices A, B, C, D, E, F, G and H(which are dense) have dimensions k_I, k_{II}, k_{III}, k_{IV}, k_V, k_{VI}, k_{VII} and k_{VIII}, respectively, and their rows correspond to the satisfaction of the boundary conditions, the governing equation and interface conditions in each region.



(5)

Figure 2. Structure of the global system for the 3-D case.

The global system (5) may be rewritten in component form as:

$$A\underline{x}_1 + R_1^*\underline{x}_2 + R_2^*\underline{x}_4 + R_3^*\underline{x}_8 = \underline{a}, \tag{6a}$$

$$S_1^*\underline{x}_1 + B\underline{x}_2 + S_2^*\underline{x}_3 + S_3^*\underline{x}_7 = \underline{b}, \tag{6b}$$

$$T_1^*\underline{x}_2 + C\underline{x}_3 + T_2^*\underline{x}_4 + T_3^*\underline{x}_6 = \underline{c}, \tag{6c}$$

$$V_1^*\underline{x}_1 + V_2^*\underline{x}_3 + D\underline{x}_4 + V_3^*\underline{x}_5 = \underline{d}, \tag{6d}$$

$$W_1^*\underline{x}_4 + E\underline{x}_5 + W_2^*\underline{x}_6 + W_3^*\underline{x}_8 = \underline{e}, \tag{6e}$$

$$X_1^*\underline{x}_3 + X_2^*\underline{x}_5 + F\underline{x}_6 + X_3^*\underline{x}_7 = \underline{f}, \tag{6f}$$

$$Y_1^*\underline{x}_2 + Y_2^*\underline{x}_6 + G\underline{x}_7 + Y_3^*\underline{x}_8 = \underline{g}, \tag{6g}$$

$$Z_1^*\underline{x}_1 + Z_2^*\underline{x}_5 + Z_3^*\underline{x}_7 + H\underline{x}_8 = \underline{h}. \tag{6h}$$

The "starred" matrices in (6a – h) are defined in a way similar to the corresponding "starred" matrices of section 2.1. As in the two-dimensional case, from (6a), (6c), (6e) and (6g) we may express $\underline{x}_1, \underline{x}_3, \underline{x}_5$ and $\underline{x}_7$ in terms of $\underline{x}_2, \underline{x}_4, \underline{x}_6$

and $\underline{x}_8$ (as again these equations are independent of each other). Substitution into (6b), (6d), (6f) and (6h) gives the Schur complement system for the vectors $\underline{x}_2, \underline{x}_4, \underline{x}_6$ and $\underline{x}_8$. The unknown vectors $\underline{x}_1, \underline{x}_3, \underline{x}_5$ and $\underline{x}_7$ may then be obtained from (6a), (6c), (6e) and (6g).

3. Experimental results

3.1. *Implementation details*

In the implementation of the above algorithms all matrix operations were performed using calls to the appropriate level 1, 2 and 3 BLAS subroutines [5] of the NAG Library [6]. In the calculations, advantage was taken of the presence of the zero blocks in the matrices $S_1^*, S_2^*, V_1^*, V_2^*$ in the 2-D case, and matrices $S_1^*, S_2^*, S_3^*, V_1^*, V_2^*, V_3^*, X_1^*, X_2^*, X_3^*, Z_1^*, Z_2^*$ and Z_3^* in the 3-D case.

The experiments were performed on an RS6000–250 workstation and on a Cray J-916 supercomputer. Timings on the RS6000 were obtained on an empty machine using the *time* function. Timings on the Cray (in the one-processor mode) were obtained using the *perftrace* utility. Timings on the Cray multiprocessor were obtained on an empty machine using the *timef* function (as *perftrace* does not produce accurate results in the multiprocessor mode). Each result presented here is an average of multiple runs.

3.2. *Numerical examples*

The performance of the capacitance matrix techniques described in section 2 was tested on two problems from [1].

Example 1: Two dimensions

$$\nabla^2\phi(x,y) = (y^2-1)e^x + (x^2-1)e^y + 2e^x + 2e^y \qquad \text{on } (-1,1)^2$$

subject to Dirichlet boundary conditions which correspond to the exact solution of this problem, $\phi(x,y) = (y^2-1)e^x + (x^2-1)e^y$. For this example we used $h_x = 0.1, h_y = 0.2$ and $M_I = M_{II} = M_{III} = M_{IV} = N_I = N_{II} = N_{III} = N_{IV} = n$. The total number of unknowns is therefore $4(n+1)^2$.

Example 2: Three dimensions

$$\begin{aligned}\nabla^2\phi(x,y,z) = {} & e^x(y^2-1)(x^2-1) + e^y(x^2-1)(z^2-1) + e^z(x^2-1)(y^2-1) \\ & + 2((z^2-1)e^y + (y^2-1)e^z + (z^2-1)e^x + (x^2-1)e^z \\ & + (y^2-1)e^x + (x^2-1)e^y) \text{ on } (-1,1)^3\end{aligned}$$

subject to Dirichlet boundary conditions which correspond to the exact solution of this problem, $\phi(x,y,z) = (y^2-1)(z^2-1)e^x + (x^2-1)(z^2-1)e^y + (x^2-1)$

Table 1
Experimental results for the 2-D case.

n	Matrix size	RS6000			Cray		
		Full system	Capacitance technique	Ratio	Full system	Capacitance technique	Ratio
4	100	0.52	0.41	1.26	0.0739	0.0517	1.42
5	144	0.82	0.52	1.57	0.152	0.0901	1.68
6	196	1.36	0.65	2.06	0.244	0.131	1.86
7	264	3.34	1.08	3.09	0.443	0.193	2.29
8	324	5.65	1.48	3.81	0.592	0.264	2.24
9	400	11.95	2.49	4.79	0.897	0.369	2.43
10	484	20.89	3.83	5.45	1.30	0.433	3.00
11	576	33.99	7.33	4.63	1.97	0.702	2.80
12	676	51.90	11.67	4.44	2.72	0.935	2.90
13	784	80.44	18.71	4.29	3.87	1.28	3.02
14	900	114.88	27.72	4.14	5.30	1.69	3.13

$(y^2 - 1)e^z$. For this example we used $h_x = 0.1, h_y = 0.2, h_z = 0.3$ and $M_s = N_s = L_s = m_s, s = I, II, ..., VIII$ and $m_I = m_{II} = ... = m_{VIII} = m$. The total number of unknowns is therefore $8(m+1)^3$.

3.3. *One processor performance*

In table 1 the timings for the 2-D problem on both computers for $n = 4, 5, \ldots, 14$ are presented (the results are presented in seconds). The ratio of execution time of the full system technique to that of the capacitance method is also shown.

The capacitance technique is clearly superior to the full system technique. The gain from its usage increases as the size of the system increases (the time ratio increases from 1.26 to 5.45 on the workstation and from 1.42 to 3.13 on the Cray). On the Cray, for $n = 14$ the full system technique was running at approximately 110 MFlops whereas the capacitance technique achieved about 89 MFlops (assuming that the practical peak performance of the one-processor Cray J-916 is about 195 MFlops [7], the capacitance technique reaches about 46% efficiency).

Table 2 presents the results of experiments with the 3-D problem on both computers (for $m = 4, 5, 6$ for the workstation and for $m = 4, 5, \ldots, 7$ for the Cray). The results are presented in seconds or fractions thereof (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 the full system technique to that of the capacitance technique is presented.

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 4.08 for the workstation and

Table 2
Experimental results for the 3-D case.

m	Matrix size	RS6000			Cray		
		Full system	Capacitance technique	Ratio	Full system	Capacitance technique	Ratio
4	1000	58.75	17.41	3.37	5.8	1.7	3.41
5	1728	284.61	69.75	4.08	27.9	7.3	3.82
6	2744				94.4	24.2	3.90
7	4096				316.0	83.4	3.79

3.90 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). For the large systems the capacitance technique executed at more than 170 MFlops so it surpassed 87% of the practical peak performance.

3.4. *Sparse solver performance*

We also experimented with the application of a general sparse matrix solver. In particular we experimented with UMFPACK, which represents one of the most recent of such solvers [9]. UMFPACK is based on a multifrontal method based on a sequence of small dense frontal matrices which are factorized using the dense matrix kernels. It is an improved version of the M48 routine from the Harwell Library [10]. Table 3 summarizes the results of the experiments performed on the RS6000 workstation and the Cray J-916 for both the 2-D problem (for $n = 4, 5, \ldots, 14$) and for the 3-D problem (for $m = 4, 5, 6$). The results for the capacitance technique (column Capaci) are provided for comparison purposes and the ratio calculated is between the values obtained by using UMFPACK (column UMF) and the capacitance technique. Empty rows indicate that the system did not fit into the memory of the RS6000 workstation. The results are reported in seconds.

In the 2-D case on the RS6000 the capacitance technique outperforms the UMFPACK based approach. In the 3-D case on the RS6000 for $m = 4$ UMFPACK slightly outperforms the capacitance technique. For $m = 5$, when the sizes of individual blocks increase, the overall matrix size increases, the capacitance technique substantially outperforms UMFPACK solver.

The situation is more interesting on the Cray. Here in all cases the capacitance technique is substantially (in the 2-D case between 2 and 3 times faster; in the 3-D case approximately 2 times faster) outperforms the UMFPACK solver. This can be explained by the fact that Cray provides highly optimized BLAS

Table 3
Performance of the sparse solver.

	2-D							
	RS6000				Cray			
n	UMF	Capaci	Ratio	NAG	UMF	Capaci	Ratio	NAG
4	0.50	0.41	1.21	0.73	0.095	0.051	1.86	0.174
5	0.59	0.52	1.13	0.66	0.152	0.090	1.68	0.491
6	0.90	0.65	1.38	1.22	0.322	0.131	2.45	0.932
7	1.24	1.08	1.14	1.99	0.457	0.193	2.36	2.47
8	2.16	1.48	1.46	3.55	0.670	0.294	2.27	4.05
9	3.65	2.49	1.46	9.04	0.989	0.369	2.68	10.7
10	5.15	3.83	1.34	8.47	1.47	0.433	3.39	13.4
11	9.42	7.33	1.28	24.03	2.00	0.702	2.84	29.8
12	14.27	11.67	1.22	31.55	2.78	0.935	2.97	29.3
13	21.97	18.71	1.17	117.26	3.52	1.28	2.75	75.1
14	30.30	27.72	1.09	61.70	5.12	1.69	3.02	73.1
m				3-D				
4	15.64	17.41	0.89	57.24	3.64	1.76	2.06	44.9
5	93.07	69.75	1.33	371.07	14,00	7.39	1.89	378.0
6					45.80	24.20	1.89	1510.0

kernels and the capacitance technique takes full advantage of them, while in UMFPACK, the multifrontal approach introduces additional overhead and prevents it from taking full advantage of the Cray's hardware. This fact is clearly illustrated by the MFlop rates achieved by both approaches. The capacitance technique reaches more than 80% of the practical peak performance of the Cray (see section 3.3) while UMFPACK reaches 33 MFlops in the 2-D case (approximately 17% of the practical peak) and up to 70 MFlops (approximately 36% of the practical peak) in the 3-D case. While this fact does not play a dominating role in the case of a RISC processor of the RS6000, it clearly makes a difference on the Cray's vector-processor. It should be also noted that in contrast to the capacitance technique, UMFPACK requires additional memory to be used as a temporary space.

We also examined the performance of the F01BRF and F04AXF pair from the NAG Library [6]. These routines are based on the MA28 package from the Harwell Library and attempt to permute the system to the lower triangular form [8]. F01BRF provides a user controllable parameter (PIVOT) which is a relative measure used to control the pivot strategy: the smaller its value the more F01BRF maintains sparsity at the expense of stability. We experimented with $PIVOT = 10^{-1}, 10^{-2}, \ldots, 10^{-6}$ while only the results for $PIVOT = 10^{-1}$ (the default value suggested in the documentation) are presented (table 3, column NAG). It should be noted that $PIVOT = 10^{-6}$ was the last value for which at

least in some cases a numerically correct results have been obtained while in most cases the breakdown in accuracy occurred as early as for $PIVOT = 10^{-4}$. A number of observations can be made. The NAG based solver is outperformed by UMFPACK, the capacitance technique and the full matrix solver (see tables 1 and 2) on both computers. For the decreasing values of *PIVOT* the performance of the NAG solver was improving (while the numerical stability was decreasing) but even for $PIVOT = 10^{-6}$ it was only approximately that of the full matrix solver. These results confirm the superiority of the newer general sparse solvers over the standard algorithm used by NAG.

3.5. *Parallel performance*

Parallelization characteristics of the proposed method were tested on the 16-processor Cray J-916. The parallelization was introduced inside the BLAS kernels. Since most of the cases have rather short execution times, only the results for the largest 3-D cases are reported. Figure 3 presents the speed-up of the capacitance technique for $m = 4, 5 \ldots, 8$ for $P = 1, 2 \ldots, 12$ processors. For small values of m we stopped experimenting when the speed-up curves flattened, indicating that further increase on the number of processors would not produce additional performance gains.

It can be observed that the capacitance technique performs quite well but the efficiency of parallelization depends heavily on the value of m. For small m a speed-up of 2 is observed while for $m = 8$ the speed-up is as high as 4.31. It can be predicted that the efficiency of the proposed method increases with m. At the same time, since parallelization was limited to the BLAS kernels, reasonable parallel performance can be expected only for shared memory computers with a small/medium number of processors. It should be mentioned that the problem in question (the linear system resulting form the discretization) is too small for

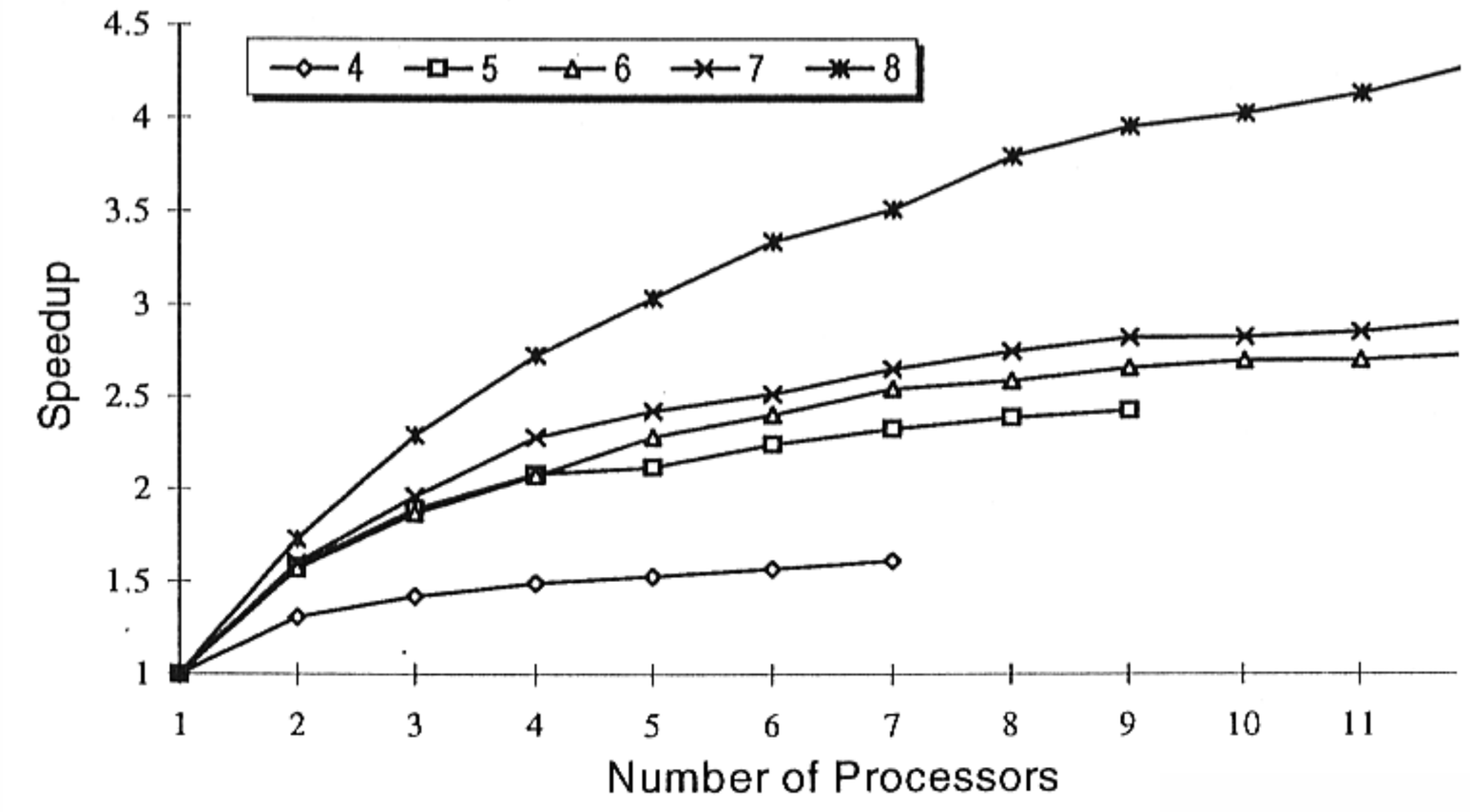


Figure 3. Speed-up for the 3-D problems.

good overall parallel performance. There are two possible ways in which the system size could increase. First, by increasing the sizes of individual blocks in the discretization (see figure 2) which would correspond to introducing a larger number of degrees of freedom in each subdomain. In this case the capacitance algorithm and its BLAS-based implementation proposed here will definitely lead to a very efficient solver for shared memory computers (with the efficiency being limited only by the efficiency of the BLAS kernels). The performance on a distributed memory architecture would still be limited by the amount of data transmission and a scheme similar to that proposed in [11] for large dense systems would need to be designed. Second, the system size could increase by increasing the number of blocks in the system (and possibly keeping their sizes constant). This would correspond to an increase in the number of subdomains. In this approach a different parallelization approach should be sought for both shared and distributed memory architectures. It is possible that a sparse matrix solver based on parallel reordering or a parallel multi-frontal-type method [8,12] could be relatively successful. The increase in the number of subdomains would, however, defeat the purpose of using the spectral domain decomposition method in the first place, as the whole idea of the method is the discretization with a relatively small number of large domains.

4. Concluding remarks

An efficient direct method is presented for the solution of linear systems arising in certain multidomain spectral collocation schemes for boundary value problems in two and three dimensions. In previous spectral multidomain discretizations [2,3] the global matrices possessed an almost block diagonal structure and could be solved using standard software. The present type of decomposition which is particularly useful in problems involving boundary layers and/or boundary singularities (see [13]) leading to linear systems which do not have an almost block diagonal structure, but are sparse and structured. As a result one cannot use standard software for the solution of linear systems as was the case in [2,3]. Furthermore, the systems are not sparse enough for a general sparse matrix solver to be effective.

The algorithm is easy to implement and provides substantial savings in storage and computational cost. In addition, for large systems, it is quite effective on shared-memory multiprocesors. The fact that the structure of the linear system depends only on the decomposition makes the algorithm widely applicable in the sense that it can be applied to any second (or higher) order problem in two and three dimensions. It would be particularly useful for non-linear problems where the non-linear governing equation is linearized using a Newton-type technique and as a consequence one needs to solve a sequence of linear systems possessing the present structure. The technique is also general in terms of the formulation of the method. It is not restricted to a collocation formulation but can also be applied to a Galerkin (weak) formulation, as the two formulations are equivalent

in this sense (see [14, chapter 13]). The reason is that the structure of the matrix depends only on two families of equations: the ones arising from the discretization of the satisfaction of the differential equation and the boundary conditions (blocks A, B, C and D in figure 1) and the ones arising from the discretization of the interface conditions (the matrices R_i,S_i and T_i in figure 1).

Acknowledgements

The computer time grant from the Computation Center of the University of Texas in Austin is kindly acknowledged. The authors would like to thank Richard Brankin for his suggestions and the two anonymous referees for their useful comments.

References

[1] A. Karageorghis, A fully conforming spectral collocation scheme for second and fourth order problems, Comput. Methods Appl. Mech. Engng. 126 (1995) 305–314.
[2] A. Karageorghis and T.N. Phillips, On efficient direct methods for conforming spectral domain decomposition techniques, J. Comput. Appl. Math. 33 (1990) 141–155.
[3] A. Karageorghis, The numerical solution of laminar flow in a re-entrant tube geometry by a Chebyshev spectral element collocation method, Comput. Methods Appl. Mech. Engng. 100 (1992) 339–358.
[4] B.L. Buzbee, F.W. Dorr, J.A. George and G.H. Golub, The direct solution of the discrete Poisson equation on irregular regions, SIAM J. Numer. Anal. 8 (1971) 722–736.
[5] J.J. Dongarra, I.S. Duff, D.C. Sorensen and H.A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM, Philadelphia, 1991.
[6] *Numerical Algorithms Group Library Mark 15*, NAG (UK) Ltd, Wilkinson House, Jordan Hill Road, Oxford, UK.
[7] M. Paprzycki and C. Cyphers, Multiplying matrices on the Cray – Practical considerations, CHPC Newsletter 6 (1991) 77–82.
[8] I.S. Duff, A.M. Erisman and J.K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, 1990.
[9] UMFPACK, available from Netlib and www.cis.ufl.edu/~davis.
[10] T.A. Davis, Users' guide for the unsymmetric-pattern multifrontal package, Technical Report, University of Florida, TR-95–004.
[11] R.A. van de Geijn, LINPACK benchmark on the Intel Touchstone GAMMA and DELTA machines, Preliminary Report, University of Texas (1991).
[12] I.S. Duff and J.A. Scott, The use of multiple fronts in Gaussian elimination, Technical Report, Rutherford Appleton Laboratory, RAL 94–040 (1994).
[13] P. Raad and A. Karageorghis, A Chebyshev spectral collocation method for the solution of the Reynolds equation of lubrication, J. Comput. Phys. 106 (1993) 42–51.
[14] C. Canuto, M.Y. Hussaini, A. Quarteroni and T.A. Zang, *Spectral Methods in Fluid Dynamics*, Springer, Berlin, 1988, Chap. 13.