ELSEVIER

Journal of Computational and Applied Mathematics 69 (1996) 71–80

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

High performance solution of partial differential equations discretized using a Chebyshev spectral collocation method

Cliff Cyphers[a], Marcin Paprzycki[a,*], Andreas Karageorghis[b]

[a]*Department of Mathematics and Computer Science, University of Texas of the Permian Basin, Odessa, TX 79762, USA*
[b]*Department of Mathematics and Statistics, University of Cyprus, Nicosia 1678, Cyprus*

Received 16 March 1994; revised 15 November 1994

Abstract

When a Chebyshev spectral collocation method is applied to a flow problem in a rectangularly decomposable domain it leads to the solution of a structured linear system. Since the linear system is solved at each step of a Newton-type iterative process an efficient method to solve it needs to be provided. A level 3 BLAS-based algorithm is presented and its efficiency is studied.

Keywords: Almost block diagonal systems; Collocation; Domain decomposition; High performance solution; Spectral methods

1. Introduction

Chebyshev spectral collocation methods are often applied to the solution of partial differential equations (PDE) on rectangular domains [9–11]. These methods require the repeated solution of a structured linear system, which becomes the most costly part of the solution process. This system has a generalized almost block diagonal (ABD) form (for a precise description of such a system, see e.g. [3, 20]). There exists a variety of approaches to the solution of ABD systems. The parallel solution was studied by Wright [21, 22], Paprzycki and Gladwell [18], Ascher and Chan [2]. All these authors assume that an ABD system is characterized by a large number of relatively small blocks. In [8, 19] a different algorithm was proposed, which achieved relatively good performance for a small number of large blocks. This algorithm was a level 3 BLAS [5]-based extension of the alternate row and column elimination algorithm proposed by Varah [20] and later modified by Diaz et al. [4]. Recently, a generalized version of this algorithm was implemented by Cyphers et al. [3]. The aim of this paper is to study the performance characteristics of the newly developed code

* Corresponding author. E-mail: pbak614@hpcf.cc.utexas.edu.

when applied to a laminar flow problem in a re-entrant tube geometry. Section 2 describes the mathematical problem. Section 3 outlines the proposed algorithm. Section 4 contains the results of experiments performed on an IBM 6000 workstation and on an 8-processor Cray Y-MP.

2. The problem and the numerical method

We consider the flow of an incompressible fluid through a re-entrant tube. Only a short description of the problem and the method is presented here (for more details, see [9]). Because the problem is symmetrical we only consider the upper half of the flow region. The flow is assumed to be steady and laminar and is governed by the Navier–Stokes equations which in the stream function formulation become

$$\nabla^4\psi - Re\left[\frac{\partial\psi}{\partial y}\frac{\partial}{\partial x}(\nabla^2\psi) - \frac{\partial\psi}{\partial x}\frac{\partial}{\partial y}(\nabla^2\psi)\right] = 0, \tag{2.1}$$

where Re is the Reynolds number. The flow region and the boundary conditions are depicted in Fig. 1.

Eq. (2.1) is linearized using a Newton-type method [9]. If ψ^* is the solution at the kth iterative step, the solution at step $k+1$ is given by ψ, where

$$\begin{aligned}\nabla^4\psi - Re&\left[\frac{\partial\psi^*}{\partial y}\frac{\partial}{\partial x}(\nabla^2\psi) + \frac{\partial\psi}{\partial y}\frac{\partial}{\partial x}(\nabla^2\psi^*) - \frac{\partial\psi}{\partial x}\frac{\partial}{\partial y}(\nabla^2\psi^*) - \frac{\partial\psi^*}{\partial x}\frac{\partial}{\partial y}(\nabla^2\psi)\right]\\ &= Re\left[\frac{\partial\psi^*}{\partial x}\frac{\partial}{\partial y}(\nabla^2\psi^*) - \frac{\partial\psi^*}{\partial y}\frac{\partial}{\partial x}(\nabla^2\psi^*)\right].\end{aligned} \tag{2.2}$$

As in [9] the flow region is divided into four elements (see Fig. 1) and in region i, i = I, II, III or IV, the solution is approximated by

$$\psi^i(x,y) = g^i(y) + \sum_{m=2}^{M_i}\sum_{n=2}^{N_i} a^i_{mn}\,\bar{T}^i_m(x)\,\hat{T}^i_n(y), \tag{2.3}$$

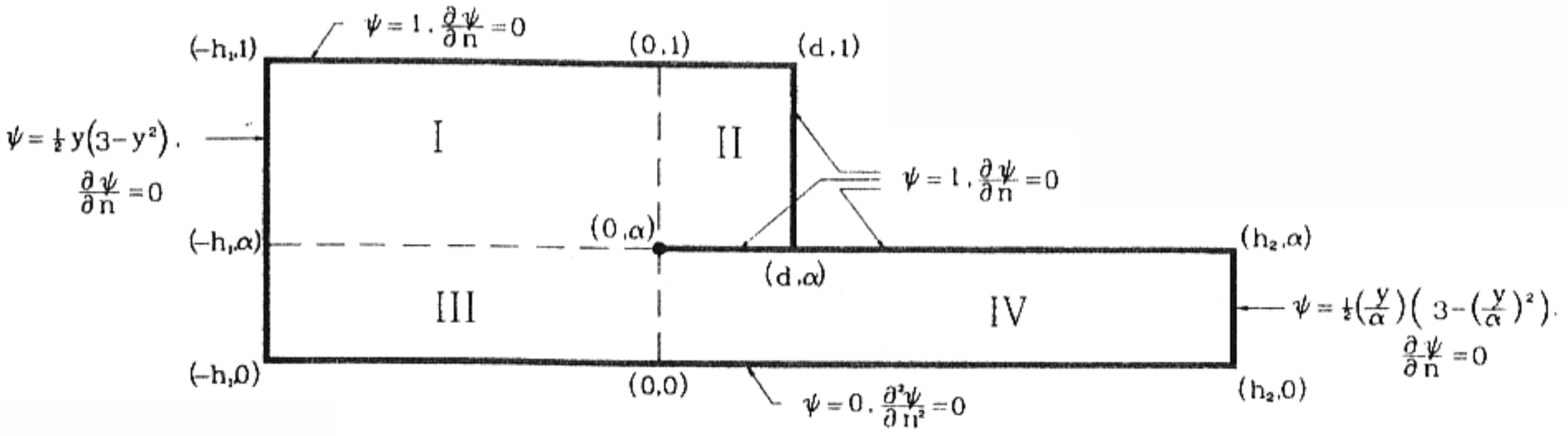

Fig. 1. The flow region and the boundary conditions.

where $\bar{T}_m$ and $\hat{T}_n$ are linear combinations of Chebyshev polynomials chosen so that some of the boundary conditions are satisfied identically on the boundary. The functions $g^i(y)$ are Poiseuille stream functions corresponding to each element. Interelement continuity is achieved by matching the approximation to ψ and its normal derivatives at a finite number of collocation points on the element interfaces. Careful selection of the number of collocation points ensures that the approximation is C^1 continuous everywhere on these interfaces [9]. The unknown coefficients a^i_{mn} in (2.3) may be found at each iteration by solving the linear system resulting from the satisfaction of (2.2) and the boundary and interface conditions, which is of the form shown in Fig. 2. In general, not all domain decompositions lead to ABD systems. For a decomposition to lead to such a system it must have the following properties: (a) Exactly two elements must have only one common interface with another element and (b) All other elements must have two common interfaces with other elements. If these criteria are not met the global matrix does not possess an ABD structure and the algorithm is not applicable (see also [11]).

Matrices 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.

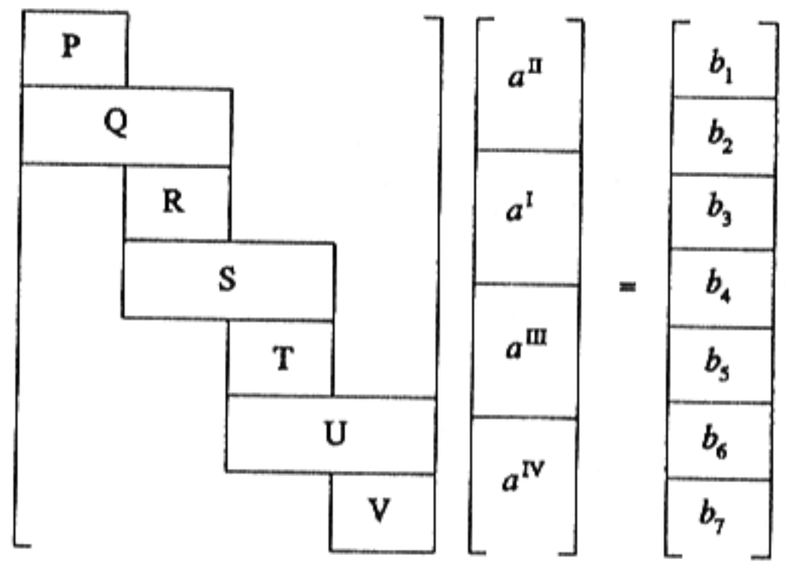

Fig. 2. The linear system resulting from the discretization.

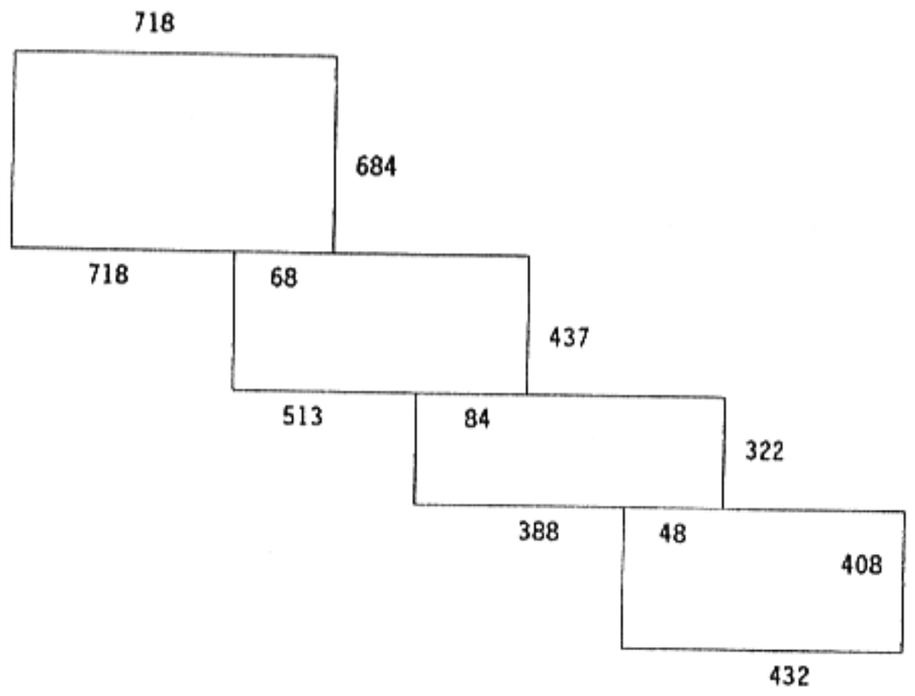

Fig. 3. Example of an almost block diagonal system to be solved in each step of the algorithm.

Clearly, the structure of the system is independent of the type of boundary conditions of the problem. It should be observed that the transpose of this system has a generalized ABD form (Fig. 3) and can solved using standard ABD solvers (the original matrix is characterized by too much overlap between the nonadjacent blocks).

3. Algorithm description

The proposed algorithm is a level 3 BLAS-based extension of the work of Diaz et al. [4]. We will describe it only briefly as a more detailed description can be found in [3, 8, 19]. For our purposes we will rewrite the ABD system as

$$\begin{pmatrix}
A_{1,1} & A_{1,2} & A_{1,3} & & & & & \\
A_{2,1} & A_{2,2} & A_{2,3} & & & & & \\
 & A_{3,2} & A_{3,3} & A_{3,4} & A_{3,5} & & & \\
 & A_{4,2} & A_{4,3} & A_{4,4} & A_{4,5} & & & \\
 & & & & & & & \\
 & & & A_{n-1,n-3} & A_{n-1,n-2} & A_{n-1,n-1} & A_{n-1,n} \\
 & & & A_{n,n-3} & A_{n,n-2} & A_{n,n-1} & A_{n,n}
\end{pmatrix},$$

where $A_{i,i}$ are square and $A_{i,j}$ are rectangular blocks of varying sizes.

The ith step of the algorithm consists of two phases. In phase I the rectangular block

$$\begin{pmatrix} A_{2i-1,2i-1} \\ A_{2i,2i-1} \end{pmatrix}$$

is decomposed using Gaussian elimination with partial pivoting and row interchanges into

$$P\begin{pmatrix} L_{2i-1,2i-1} \\ L_{2i,2i-1} \end{pmatrix} U_{2i-1,2i-1},$$

where P is the permutation matrix. After this factorization, block

$$\begin{pmatrix} A_{2i-1,2i} & A_{2i-1,2i+1} \\ A_{2i,2i} & A_{2i,2i+1} \end{pmatrix}$$

will be updated by the inverse of

$$\begin{pmatrix} L_{2i-1,2i-1} & 0 \\ L_{2i,2i-1} & I \end{pmatrix}.$$

In phase II, the block

$$(A_{2i,2i} \;\; A_{2i,2i+1})$$

will be decomposed using Gaussian elimination with partial pivoting and column interchanges into

$$L_{2i,2i}(U_{2i,2i}\ U_{2i,2i+1})Q,$$

where Q is the permutation matrix. After this factorization, block

$$\begin{pmatrix} A_{2i+1,2i} & A_{2i+1,2i+1} \\ A_{2i+2,2i} & A_{2i+2,2i+1} \end{pmatrix}$$

will be updated by the inverse of

$$\begin{pmatrix} U_{2i,2i} & U_{2i,2i+1} \\ 0 & I \end{pmatrix}.$$

The decomposition will be performed in a block fashion using the LAPACK [1] provided routine SGETRF (which uses appropriate level 1 and 2 BLAS kernels [6, 12]). The update steps consist of calls to the level 3 BLAS routines STRSM and SGEMM.

4. Numerical experiments

The system described above is a 4-block ABD system where the size of the first block depends on d. We experimented with four different values of $d = 0.2, 0.5, 1.0$ and 1.5. Table 1 summarizes the sizes of the first block corresponding to different values of d (the sizes of the remaining blocks are as presented in Fig. 3); OVRL represents the overlap between the first and the second block; the total system size varies between 1547×1547 and 1851×1851.

The experiments were performed using the level 3 BLAS-based code proposed in [3] and its performance was compared to the best existing code F01LHF/F04LHF (decomposition/back substitution) pair from NAG [13]. All results presented here are averages of multiple (minimum of three) runs.

4.1. Experiments on an IBM RS/6000 workstation

We performed experiments on an 24 ns IBM RS/6000 station model 550. All experiments were performed on an empty machine. The IBM provided optimized BLAS kernels were used in both the proposed code and NAG routines. Timings were obtained using the system timer (routine *times*(3)).

Table 1
Sizes of the first block of an ABD system depending on the value of d

	$d = 0.2$	$d = 0.5$	$d = 1.0$	$d = 1.5$
Rows	380	475	646	684
Columns	414	509	680	718
OVRL	68	68	68	68

The first series of experiments was executed for the solution of a linear system of a given size. In the decomposition step we experimented with a variety of blocksizes and we found that the blocksizes 32 and 64 lead to the best performance (which is consistent with [7]). The results for blocksize 32 are summarized in Table 2.

The performance of the new code in the decomposition is approximately 46 MFlops and about 30 MFlops for back substitution. Since the practical peak performance for this model of RS/6000 station is about 75 MFlops [7], the new code was running at approximately 61% of this practical peak in the decomposition and approximately 40% in the back substitution steps.

The second series of experiments was performed with the solution of the whole problem. The results reported in Table 3 are for the (the most efficient) blocksize 32, for the largest system ($d = 1.5$), for varying Re; ITER specifies how many iterations (solutions of the linear system) were performed before an acceptable solution was reached (for $Re > 100$ the solution was found in steps of 100 using the converged solution at the end of each step as a starting point for the next step).

Assuming that most of the work is done in the solution of the linear system (which is true especially for the largest problems with multiple iterations), the whole application is running at 40 MFlops, which constitutes approximately 53% of the practical peak performance. It can be also observed that the time reduction is approximately 56% and is independent of the value of Re.

4.2. Experiments on a Cray Y-MP

The second series of experiments was performed on an 8-processor Cray Y-MP 8/864. The experiments were performed in both single processor and parallel modes. Cray Assembly Language

Table 2
Solution of the linear system; time in seconds

d	Decomposition		Back substitution	
	New code	F01LHF	New code	F04LHF
0.2	3.470	8.100	0.040	0.065
0.5	4.200	9.785	0.050	0.070
1.0	6.335	14.775	0.055	0.090
1.5	7.030	17.730	0.065	0.100

Table 3
Solution of the whole problem; time in seconds

Re	ITER	New code	NAG
5	4	34.95	80.89
100	8	65.56	153.00
200	13	99.00	229.50
300	21	139.47	325.48
400	22	180.58	420.75

coded BLAS kernels were used. For the multi-processor runs, the parallelization was introduced within the BLAS kernels. For the single-processor experiments the code efficiency was measured using the Cray provided routine *perftrace*. The multi-processor runs were performed on an empty machine and the *timef* routine was used to measure execution time.

The first series of experiments was performed in one-processor mode for the solution of a linear system of a given size only. After experimenting with a variety of blocksizes we found that when only one processor is used the blocksize 128 leads to the best performance (which is consistent with the results presented in [14]). The results for this blocksize are summarized in Table 4.

As Table 4 clearly shows, the new code outperforms the old one by 12–15 MFlops (approximately 5% performance gain) in the decomposition step. If we assume that the practical peak performance of a single processor of the Cray Y-MP is approximately 315 MFlops [15, 17] then the new code obtains approximately 91% of this peak. The performance of the back substitution routine must be much lower, since each time only one right-hand side is considered. This effectively reduces the performance from what we would expect for a level 3 BLAS-based code to the level 2 BLAS (matrix–vector operations instead of matrix–matrix operations). The back substitution results of the NAG routine represent a known problem with the transpose version of the F04LHF solver [19].

Table 4
Solution of the linear system; one-processor performance in MFlops

d	Decomposition		Back substitution	
	New code	F01LHF	New code	F04LHF
0.2	279.0	266.7	80.0	11.4
0.5	281.9	269.0	83.6	11.0
1.0	285.5	268.3	91.6	10.3
1.5	288.7	272.8	93.9	10.2

Table 5
Solution of the whole problem; one processor; results in MFlops

d	$Re = 5$		$Re = 400$	
	New code	F01LHF	New code	F04LHF
0.2	227.7	199.3	227.4	198.6
0.5	229.4	200.0	228.6	198.2
1.0	258.5	222.5	257.1	220.1
1.5	260.8	225.5	259.7	222.5

Secondly, we experimented with a one-processor solution of the whole problem. Table 5 presents the results for $Re = 5$ and $Re = 400$, for varying d (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 solver(s). The performance gains from the use of the new code are 18–35 MFlops (between 12% and 14%) and grow slightly as the size of the system increases. For the large system the performance is approximately 82% of the practical peak.

The last series of experiments was performed using the 8-processor Cray Y-MP as a parallel computer. The results in Table 6 present the times for a varying number of processors for the solution of the whole problem for $Re = 400$, for $d = 0.5$ and $d = 1.5$. In this case, it took 28 iterations to solve the small problem and 22 iterations to solve the large one. As the number of processors increases the optimal blocksize increases as well (which is consistent with results reported in [16]). A number of blocksizes were tested and the best 8-processor performance of the new code was obtained for the blocksize 256; these results are reported in Table 6.

The performance of the new code is significantly better than that of NAG for any number of processors. The performance gains for the large number of processors are 16% for the smaller and 23% for the larger system. It should be mentioned that even though NAG routines are not designed with parallel computations in mind, the parallelization is introduced by calls to level 1 and level 2 BLAS kernels.

Finally, in Fig. 4 the speed-up for the solution of the whole problem by the new code is presented for the largest system size ($d = 1.5$) and for varying values of Re. The result for NAG is presented for $Re = 400$.

The speed-up for the largest problem on 8 processors is approx. 2.7. This result is much worse than the speed-up of 4.5 reported in [8] for an ABD system with smaller blocks. This can be explained by the fact that although the individual blocks of the system are large enough, the overlaps between them are much too small. The solution process in the overlaps constitutes the serial bottleneck for this problem.

Table 6
Solution of the whole problem; for varying number of processors

No.	$d = 0.5$		$d = 1.5$	
	New code	NAG	New code	NAG
1	31.10	35.08	35.40	40.59
2	22.22	25.55	22.81	27.09
3	18.81	22.20	18.35	22.51
4	17.07	20.47	16.05	20.22
5	16.23	19.71	14.93	19.05
6	15.78	18.95	14.04	18.22
7	15.32	18.61	13.54	17.67
8	15.03	18.25	13.15	17.35

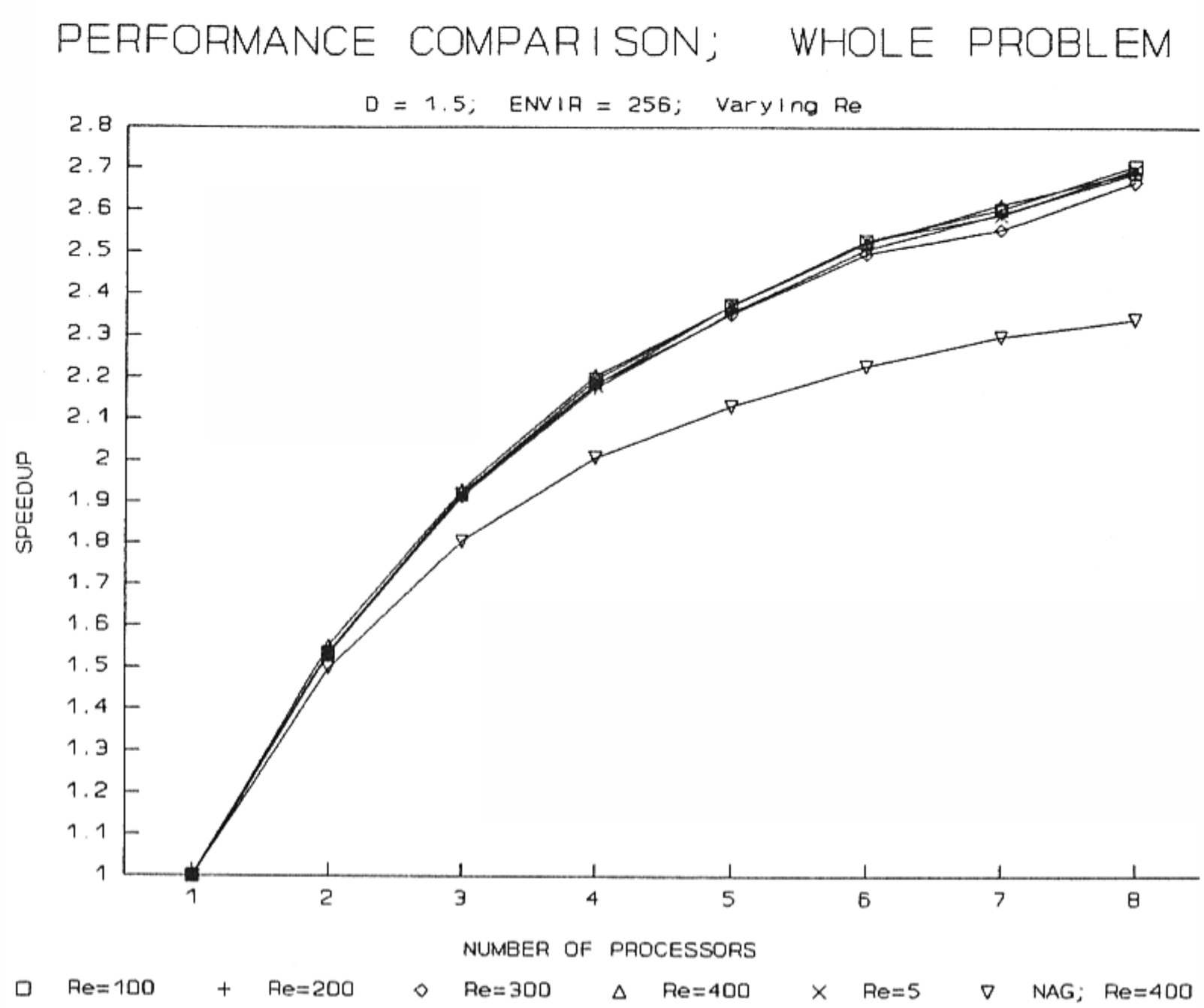

Fig. 4. Speed-ups for the solution of the whole problem.

5. Conclusions

A new code for the solution of ABD systems arising from the application of a Chebyshev spectral collocation method to a flow problem in a rectangularly decomposable domain is presented. This problem is an excellent representative of the class of problems leading to ABD systems. It represents a complex physical phenomenon the numerical solution of which reveals features of the flow apparently not previously observed. The natural decomposition of the domain leads to a global system with an ABD structure. The example is general enough as the algorithm proposed in this study depends on the structure of the global matrix and not on the number of elements. The performance of the new code is compared with the best existing code. It is shown that application of level 3 BLAS leads to considerable performance gains on high end workstations as well as on a vector computer. The relatively small number of elements hampers the parallel performance of the proposed code. Since there are only four blocks in the discretization the problem will also be "too small" for other parallelizing techniques based on divide and conquer strategies. The relatively small number of elements is, however, inevitable, because of the nature of the numerical method.

Acknowledgements

The computer time grants from the Center for High Performance Computing in Austin and Cray Research Inc., are kindly acknowledged. The authors would like to express their deep gratitude to Chris Hempel from Cray Research Inc., and to Robert Harkness from CHPC for their advice in running the experiments and optimizing the code performance.

References

[1] 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).
[2] U.M. Ascher and S.Y.P. Chan, On parallel methods for boundary value ODEs, *Computing* **46** (1991) 1–17.
[3] C. Cyphers, M. Paprzycki and I. Gladwell, A level 3 BLAS based solver for almost block diagonal systems, SMU Software Report 92-3, Southern Methodist Univ., 1992.
[4] J.C. Diaz, G. Fairweather and P. Keast, FORTRAN packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination, *ACM Trans. Math. Software* **9** (1983) 359–375.
[5] J.J. Dongarra, J. Du Croz and S. Hammarling, A set of level 3 basic linear algebra subprograms, Tech. Report ANL-MCS-TM57, Argonne National Laboratory, 1988.
[6] J.J. Dongarra, J. Du Croz, S. Hammarling and R.J. Hanson, An extended set of FORTRAN basic linear algebra subprograms, *ACM Trans. Math. Software* **14** (1988) 1–17.
[7] J.J. Dongarra, P. Mayes and G. Radicati, The IBM RISC System/6000 and linear algebra operations, *Super Computer* **8** (1991) 15–30.
[8] I. Gladwell and M. Paprzycki, Parallel solution of almost block diagonal systems using level 3 BLAS, *J. Comp. Appl. Math.* **45** (1993) 181–189.
[9] A. Karageorghis, The numerical solution of laminar flow in a re-entrant tube geometry by a Chebyshev spectral element collocation method, *Comp. Meth. Appl. Mech. Eng.* **100** (1992) 339–358.
[10] A. Karageorghis and T.N. Phillips, Conforming Chebyshev spectral collocation methods for the solution of laminar flow in a constricted channel, *IMA J. Numer. Anal.* **11** (1991) 33–55.
[11] J.A. Kolodziej and G. Musielak, Domain decomposition in boundary collocation method, in: W. Hackbush, Ed., *Parallel Algorithms for Partial Differential Equations; Proc. 6th GAMM-Seminar*, Kiel, January 19–21 (1990) (Vieweg, 1991).
[12] C.L. Lawson, R.J. Hanson, D.R. Kincaid and F.T. Krogh, Basic linear algebra subprograms for FORTRAN usage, *ACM Trans. Math. Software* **5** (1979) 306–323.
[13] *Numerical Algorithms Group Library; Mark 15* (NAG Ltd., Oxford, 1988).
[14] M. Paprzycki, Comparison of Gaussian Elimination algorithms on a Cray Y-MP, *Lin. Alg. Appl.* **172** (1992) 57–69.
[15] M. Paprzycki, Parallel matrix multiplication — can we learn anything new?, *CHPC Newsletter* **7** (1992) 55–59.
[16] M. Paprzycki, Comparisons of parallel Gaussian Elimination algorithms on an 8-processor Cray Y-MP, *Informatica* **19** (in press).
[17] M. Paprzycki and C. Cyphers, Multiplying matrices on the Cray — practical considerations, *CHPC Newsletter* **6** (1991) 77–82.
[18] M. Paprzycki and I. Gladwell, Solving almost block diagonal systems on parallel computers, *Parallel Comp.* **17** (1991) 133–153.
[19] M. Paprzycki and I. Gladwell, Solving almost block diagonal systems using level 3 BLAS, *Proc. 5th SIAM Conf. on Parallel Processing for Scientific Computing* (SIAM, Philadelphia, 1992) 52–62.
[20] J.M. Varah, Alternate row and column elimination for solving certain linear systems, *SIAM J. Numer. Anal.* **13** (1976) 71–75.
[21] S.J. Wright, Stable parallel algorithms for two-point boundary value problems, *SIAM J. Sci. Stat. Comput.* **13** (1992) 742–746.
[22] S.J. Wright, Stable parallel elimination for boundary value ODE's, *Numer. Math.* **67** (1994) 521–536.