CESA'98

Under the High Patronage of His Excellency the President of the Republic of Tunisia /
Sous le Haut Patronage de son Excellence Monsieur le Président de la République Tunisienne

Proceedings

Edited by

Pierre Borne
Mekki Ksouri
Abdelkader El Kamel

COMPUTATIONAL ENGINEERING IN SYSTEMS APPLICATIONS

IMACS Multiconference

Nabeul-Hammamet, Tunisia • April 1-4, 1998

CESA'98 Conference Secretariat

UCIS - Ecole Centrale de Lille
BP 48 - 59651 Villeneuve d'Ascq Cedex - France
Fax : (33/0) 3 20 33 54 99
E-mail : cesa98@ec-lille.fr
URL : http://www.ec-lille.fr/~cesa98

Solving Block-Structured Matrix Problems on RISC-Based Supercomputers

M. Paprzycki
Department of Computer Science and Statistics
University of Southern Mississippi
Hattiesburg, MS 39406, USA

P. Yalamov
Center of Applied Mathematics and Informatics
University of Rousse
7017 Rousse, Bulgaria

N.E. Mastorakis
Military Institutions of University Education
Chair of Computer Science
Hellenic Naval Academy
18539 Piraeus, Greece

ABSTRACT

Block-structured matrices appear in computational practice as results of discretizations of a number of mathematical problems. Usually, operations on these matrices become the most expensive part of the solution of the original problem. To provide an efficient tool to operate on such matrices, a level 3 BLAS based library of subroutines is proposed and its performance studied on the RISC-based SGI Power Challenge 8000 and Power Challenge 10000 supercomputers.

1. INTRODUCTION

A number of mathematical problems give rise to block bidiagonal (*BBD*), block tridiagonal (*BTD*), almost block diagonal *(ABD)* and other block-structured matrices [1,4,6,8,9,11]. In computational practice there exist a number of situations when efficiency of operations on such matrices becomes particularly important: first, when the matrix size is very large and thus operating on it is very time consuming; second, when a block-structured matrix appears in a non-linear problem which is solved using an iterative scheme and thus it has to be factorized in almost every step of the algorithm; third, when operations on a very large block-structured matrix are performed on a parallel computer. In the last case a tearing-type strategy is typically applied to achieve parallelization [1,2,5]. In all three cases the efficiency of the matrix operations becomes the bottleneck of the solution process.

In this note, we study the efficiency of a level 3 BLAS based [3] library of subroutines designed to perform three basic operations on block structured matrices: matrix multiplication, LU factorization with partial pivoting and back substitution. Obviously, when the solution process calls for operating on block structured matrices (of the three types described above) there are at least two possibilities of representing them. First, they can be represented directly in the blocked form. For instance, in Fortran they would be represented as a sequence of block-rows stored consecutively in a long vector. Each row-block would be stored as a sequence of columns. Second, they can be represented as a banded matrix encompassing the blocked structure. In this case the main diagonal, as well as each super- and sub-diagonal will be stored as a vector. In this case a penalty of storing a number of zeroes (residing outside of the block structure) is paid. At the same time, it can be argued that some simplicity of programming can be gained. In addition, the results gathered earlier on the Cray J-9x suggest that (on a vector computer) when the block size (n) is relatively small the banded approach based on long vectors has a significant advantage over the blocked representation [10]. Here we concentrate our attention on the RISC-based supercomputers reporting results collected on the SGI Power Challenge 8000 (PC 8000) and Power Challenge 10000 (PC 10000) supercomputers. For these two computers we will try to answer a basic question: should the block-structured matrices be represented in the blocked form, or should they be represented as banded matrices. We will also use the collected benchmarking data to compare the performance of the two machines.

2. MATRIX-VECTOR MULTIPLICATION

Multiplication of a block-structured matrix by a vector or by a square dense matrix of any size can be easily realized by a sequence of calls to the level 3 BLAS matrix multiplication routine *_GEMM*. In each step of the algorithm a complete block-row is multiplied by an appropriate part of the vector (matrix). Following the design principles of the *LAPACK* library, both the straightforward matrix multiplication and the multiplication by the transposed block-matrix are available in the same driver subroutine. The latter is achieved by appropriately applying the *transpose* option of the *_GEMM* subroutine.

Let us assume that A is a *BTD* matrix consisting of k row-blocks of size n. Here, the first and last row-block are of size $n \times 2n$ while the remaining row blocks are of size $n \times 3n$. When this matrix is stored as a banded matrix it is stored as a collection of vectors representing the main diagonal, the $2n-1$ sub-diagonals

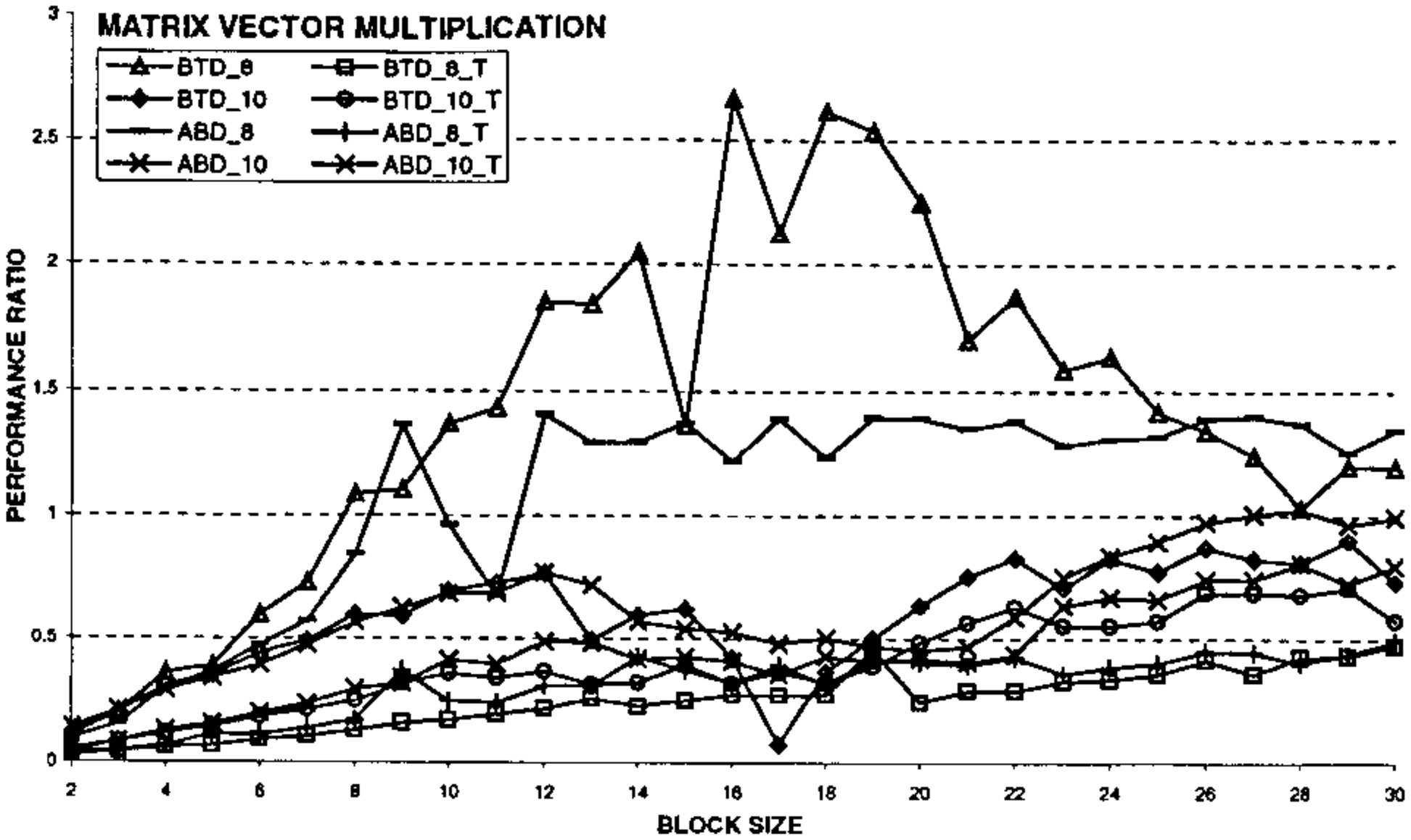

Figure 1. Matrix-vector multiplication banded vs. blocked approach; 'BTD' denotes block tridiagonal matrix; 'ABD' denotes almost block diagonal matrix; '8' denotes PC 8000; '10' denotes PC 10000; 'T' denotes transpose.

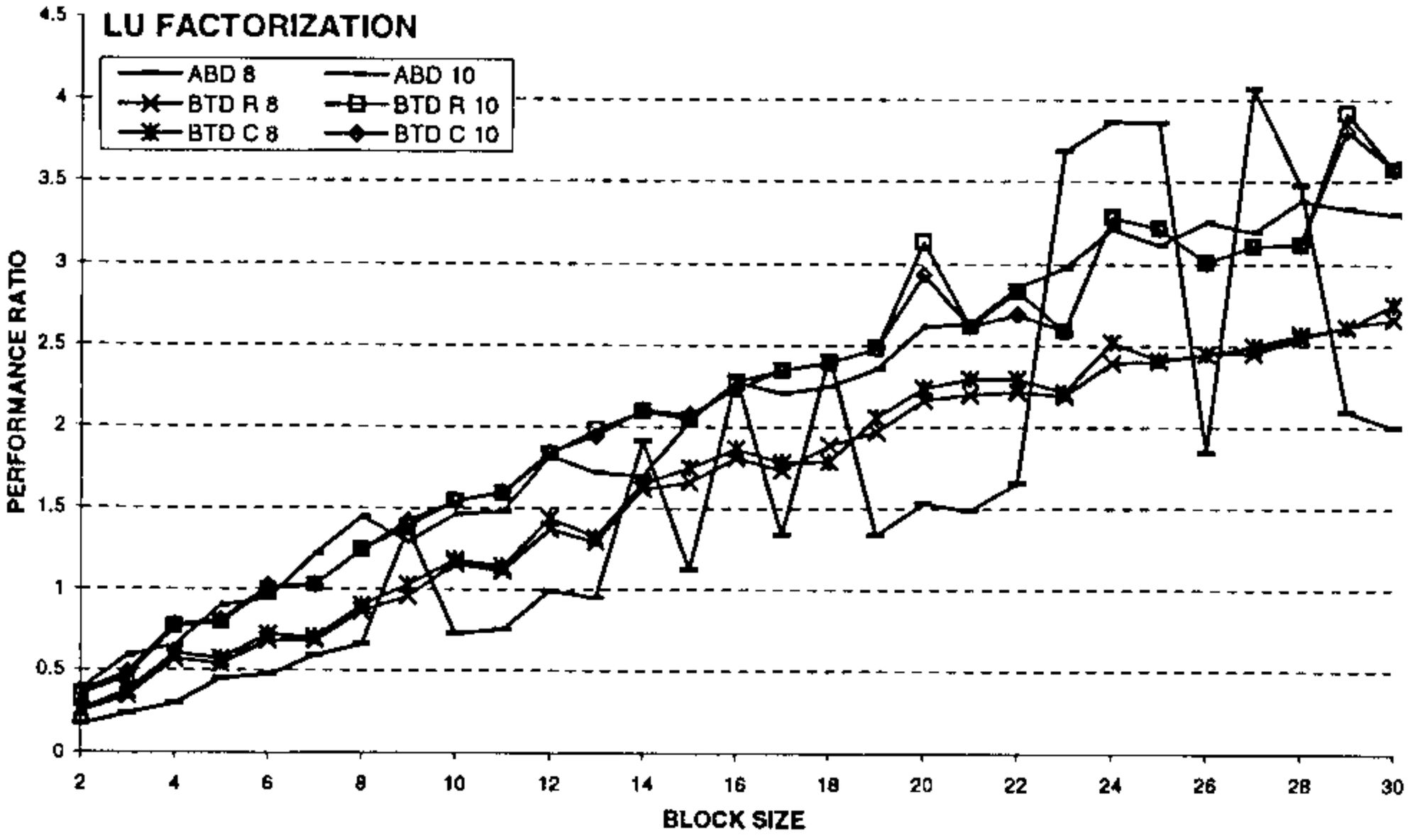

Figure 2. Matrix factorization banded vs. blocked approach; 'ABD' denotes almost block diagonal matrix; 'BTD' denotes block tridiagonal matrix; '8' denotes PC 8000; '10' denotes PC 10000; 'R' denotes row pivoting, 'C' denotes column pivoting.

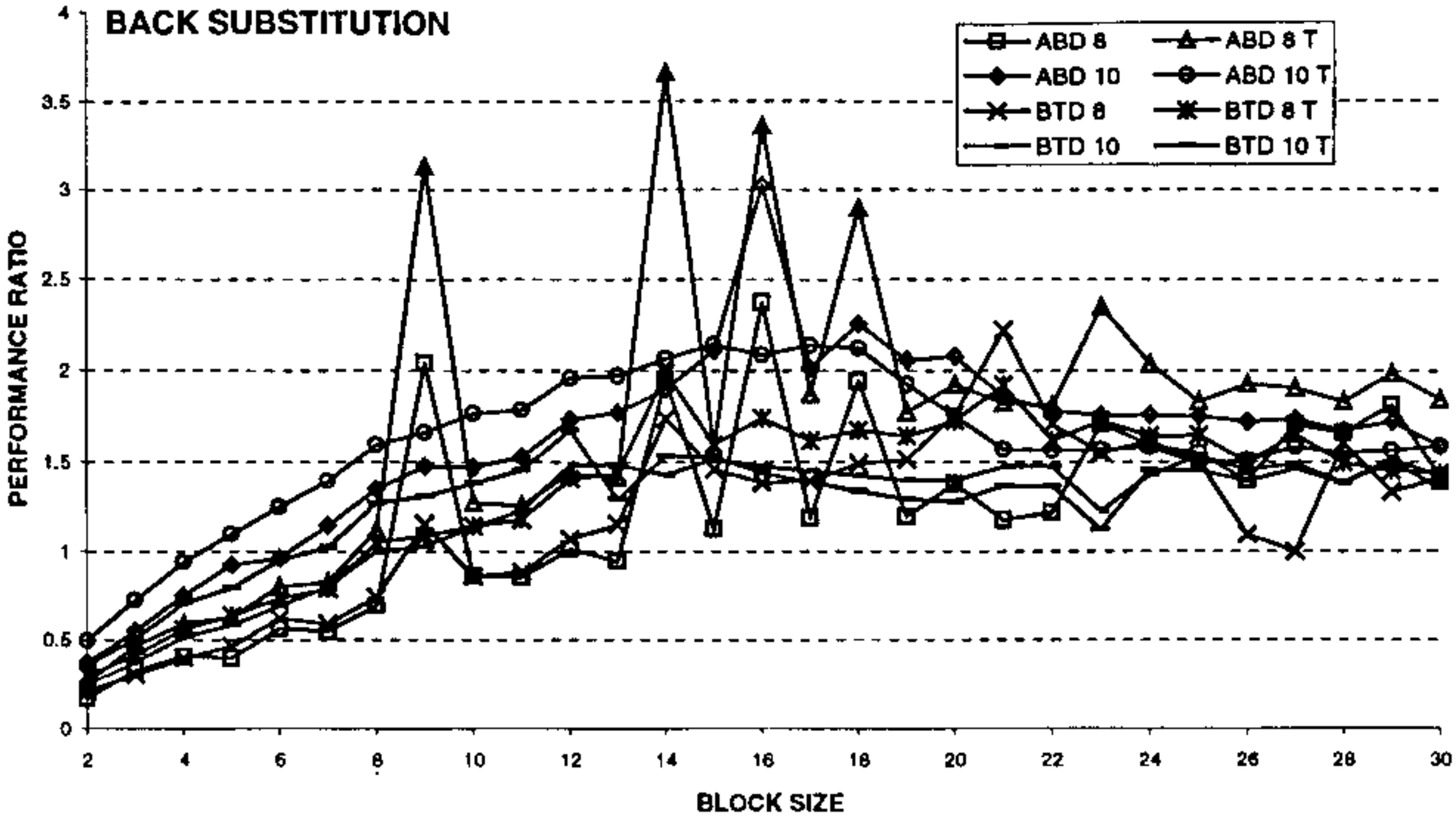

Figure 3. Back substitution; banded vs. blocked approach; 'ABD' denotes almost block diagonal matrix; 'BTD' denotes block tridiagonal matrix; '8' denotes PC 8000; '10' denotes PC 10000; 'T' denotes transpose.

and the $2n-1$ super-diagonals. In case when A is an almost block diagonal matrix, the situation is slightly more complicated. Such matrices arise typically when boundary value problems for ordinary differential equations are discretized. The sizes of particular blocks (and especially the top and the bottom block) depend on the choice of the discretization scheme and the distribution of boundary conditions. For the purpose of this paper we restrict our attention to the simplest case of matrices that arise when a system of n first order differential equations with separated boundary conditions is discretized using a finite difference approach. In this case each internal row-block is of size $n \times 2n$. To further simplify the picture let us assume that the boundary conditions are equally distributed on both sides of the interval ($q = \lfloor n/2 \rfloor$). In this case the first block is of size $q \times n$ and the last block is of size $(n-q) \times n$. This *ABD* matrix can be represented as a banded matrix with $2n-q-1$ super-diagonals and as many sub-diagonals.

In the first series of experiments the performance of the block-oriented matrix-vector multiplication was compared with the performance of the level 2 BLAS routine *_GBMV* from the *LAPACK* library. This routine performs a matrix-vector multiplication for a banded matrix stored as described above. In Figure 1 we present the ratio of time of the banded and block-oriented routines for the increasing block size n for *BTD* and *ABD* matrices and their transposes (denoted by T) on both PC 8000 and PC 10000. The results for the *BBD* matrices have been omitted to preserve the readability of the figure, but they were similar to the *BTD* matrices. In all experiments the CPU times have been collected using the *dtime* Unix timer and each result represents an average of multiple runs. In this, and the remaining figures, the results are reported for $k = 400$ blocks.

We have also performed experiments varying the number of blocks and have found not effects on performance.

The results are rather discouraging. Only for the straightforward matrix-vector multiplication of the *BTD* matrices on the PC 8000 the blocked approach becomes a feasible alternative to the banded one. This result is similar to what was detected on the Cray and shows a high level of optimization of the *LAPACK* routines. In can be noticed that, as the size of the blocks increases, the performance of the blocked routine slowly reaches the level of the banded one. This suggests that for very large blocks the blocked approach would become a viable alternative.

3. MATRIX FACTORZATION

The second series of experiments was designed to compare the performance of the LU factorization routines. Here the question of fairness of comparison needs to be addressed. The block oriented algorithms perform the LU factorization utilizing special pivoting strategies designed to avoid fill-in. In case of *ABD* matrices, an alternate row and column pivoting strategy is used (for more details see Varah [12]). The numerical stability of this approach has been proven to be equivalent to the stability of LU factorization with partial pivoting [12]. For the *BTD* and *BBD* matrices a restricted pivoting is applied (for more details see Varah [11]). The pivot element is sought only in the diagonal block. (In this case, restricted row or column pivoting can be applied so both methods have been implemented for both *BBD* and *BTD* matrices.) In [11] Varah has proven that this approach is stable if the matrices are block diagonally dominant. In the *LAPACK* library the LU factorization for banded matrices

is performed by the *_GBTRF* routine which uses standard column pivoting with row interchanges. This means that a fill-in of the size of the upper bandwidth is generated (and thus additional arithmetical operations performed). We believe that from the point of view of numerical stability the block-oriented and banded algorithms are similar enough to make a straightforward performance comparison between the two approaches. It is assumed that the additional cost of the banded solver is the price that one has to pay for using a black-box library software. In Figure 2 we present the performance ratio of the banded and block-oriented LU factorization routines for the increasing block size n.

The results are much more favorable for the blocked approaches. On both computers for approximately $n>9$ the block-oriented routines outperform the banded one. In addition, the performance gain continues to increase as the block size n increases. Clearly, in this case the long vectors used in the banded representation provide no performance advantage. For the *ABD* matrices on the PC 8000 a relatively strange variance of results can be observed (which may be also responsible for the strange behavior of the matrix-vector multiplication, see Figure 1). This phenomenon does not have an immediate explanation but (most likely) can be linked to the implementation details of the BLAS kernels on the PC 8000. We plan to investigate it further in the near future. The performance gain is slightly larger for the *BTD* than for the *ABD* matrices. This can be explained by the width of the banded matrix that encompasses both block structures. In case of the *BTD* matrices the banded system is wider and while larger fill-in is generated also a more substantial number of arithmetical operations is performed. In case of *BTD* (and, as our other experiments show, also *BBD* matrices) there is almost no performance difference between the column and row pivoting strategies. Additionally, no numerical difference between these strategies has been observed.

4. BACK SUBSTITUTION

The third series of experiments compared the performance of the block oriented and banded back solvers applied to the problem with a single right hand side. In Figure 3 we present the ratio of times used by the level 2 BLAS based banded solver *_GBTRS* from the *LAPACK* library and the block-oriented back-solvers, for the increasing block size n. Since, for the *BTD* matrices, both the row and column pivoting based factorizations have been implemented the library contains four back solvers: row pivoting, row pivoting for a transposed original matrix, and similar two versions of column pivoting oriented solvers. However, since there was no particular difference in performance between them, only the row pivoting based back solvers are reported. They are compared with the banded and banded transposed solvers as appropriate.

As in the case of LU factorization, starting from a relatively small block size ($n>13$) the block-oriented solvers outperform the banded approach (similar results were observed for the *BBD* matrices). Again, a significant variance in performance occurs on the PC 8000 for the *ABD* matrices. No substantial difference between the transposed and the non-transposed solvers can be observed. It should be noted that the behavior of the back solvers differs from that of the matrix-vector multiplication and the LU factorization. In the earlier cases, as the block size n increased the performance gains from using the blocked approach increased. For the back solvers, the performance ratio is relatively flat (centered around 1.5 times faster) and seems to decrease as the value of n increases. Still, it can be predicted that for large values of n the difference will start to increase as the banded solver performs additional arithmetical operations.

5. COMPUTER COMPARISON

The collected data allows us also to attempt to compare the two RISC based supercomputers from SGI. The Power Challenge 8000 is an older machine and its processor (MIPS 8000) does not belong to the main product line of the MIPS processors as it was designed primarily for the floating point performance. The Power Challenge 10000 is a newer architecture and it continues the main, integer performance oriented, MIPS product line (for more details about MIPS processors see [7]). Thus the performance gains of the 10000 model over the 8000 model should manifest themselves primarily in the integer arithmetic, with only about 25% gain in the floating point performance. In the next three figures we present the ratio of times obtained on the PC 8000 and the PC 10000 for matrix vector multiplication, LU factorization and back substitution. In our comparisons we report the performance of both, blocked and banded, algorithms.

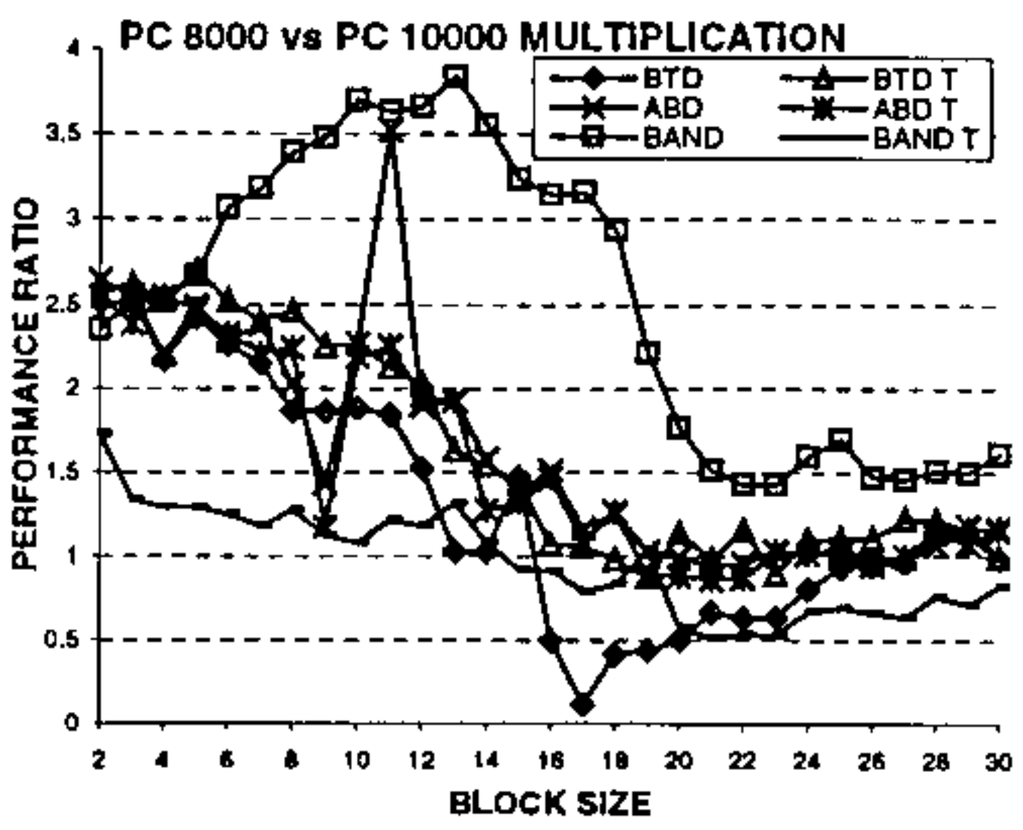

Figure 4. Matrix-vector multiplication. (Notation as above.)

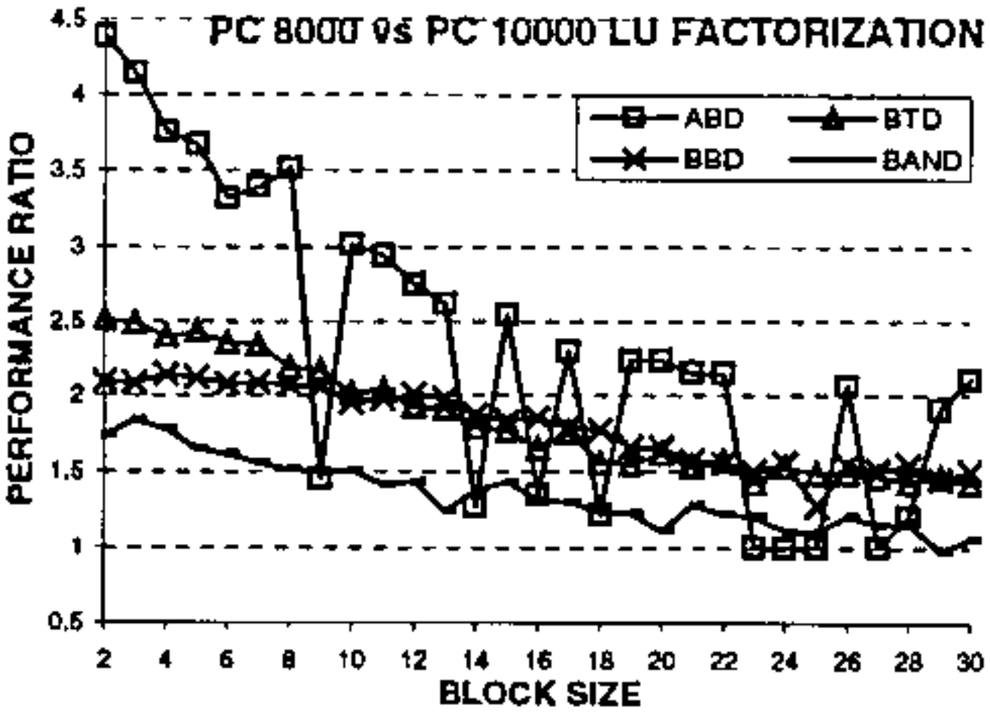

Figure 5. LU Factorization. (Notation as above.)

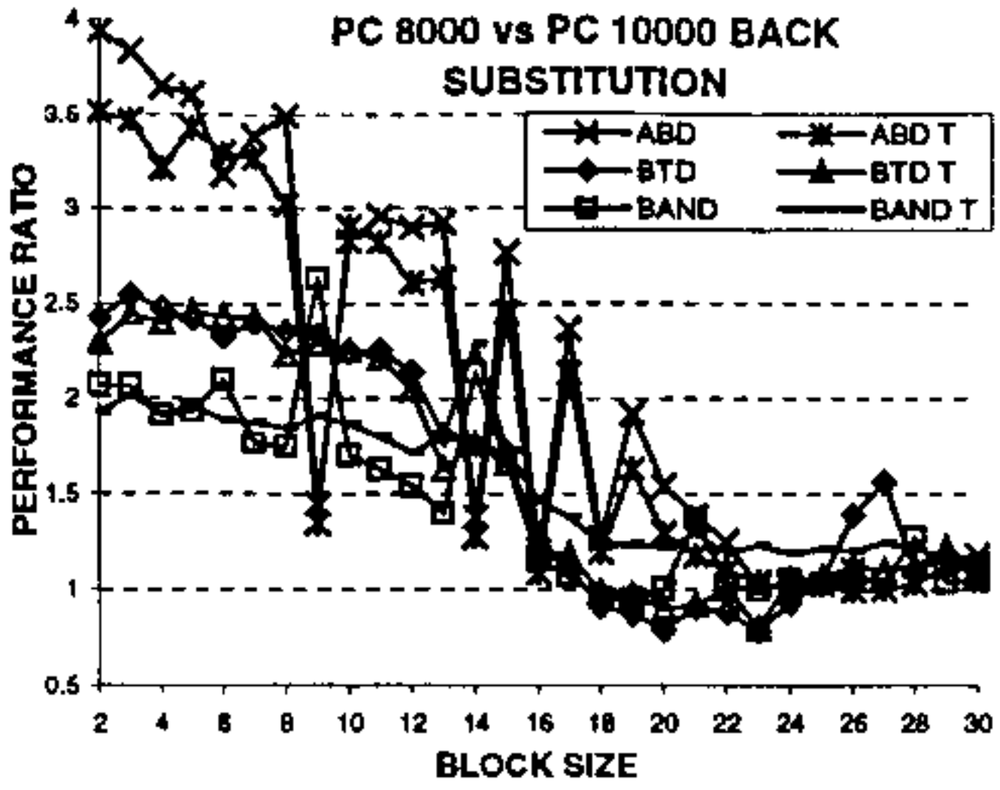

Figure 6. Back substitution. (Notation as above.)

We can observe that in all cases the performance advantage of the PC 10000 decreases as the problem size increases. This result holds for both the proposed *BLAS* based blocked codes and the *LAPACK* library based banded matrix oriented routines. The performance difference is particularly small for the matrix-vector multiplication and back substitution. Again, the performance jumps of the ABD routines can be observed, but the general pattern remains unchanged. These results are rather surprising as they indicate that already for blocks of size approximately $n = 15$ the floating point arithmetic dominates the solution and acts as the performance equalizer.

6. CONCLUDING REMARKS

We have compared the performance of banded and block-oriented approaches to the basic operations (multiplication, factorization and back-substitution) on block-structured matrices. We have found that on the SGI Power Challenge 8000 and Power Challenge 10000 RISC-based supercomputers the block-oriented approach outperforms the banded method for the LU factorization and back substitution for matrices with blocks of size $n>10$. In case of matrix vector multiplication the banded approach is a clear winner for all block sizes studied. These results resemble quite closely those observed on the Cray vector computer [10]. The results present an interesting challenge to the designers of codes similar to COLNEW [4] that rely heavily on block structured linear algebra. Finally, the results also show that for floating point arithmetic based applications there is almost no difference in performance between the PC 8000 and the PC 10000 RISC based supercomputers.

7. ACKNOWLEDGEMENTS

Computer time grant from NCSA at Urbana is kindly acknowledged. The research has been initiated by a COBASE grant from the National Research Council. The second author was supported by Grant I-702/97 from the National Research Fund of the Bulgarian Ministry of Education and Science.

REFERENCES

[1] P. Amodio, T. Politi, M. Paprzycki, "Survey of Parallel Algorithms for Block Bidiagonal Linear Systems," *Journal of Computers in Mathematics Applications*, Vol. 31, No. 7, 1996, pp. 111-127

[2] P. Amodio, T. Politi, M. Paprzycki, "Solving Block Bidiagonal Linear Systems on Distributed Memory Compters," *Proceedings of the Seventh International Conference on Parallel and Distributed Computing Systems*, ISCA, Raleigh, NC, 1994, pp. 812-815

[3] 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

[4] U. Ascher, J. Christiansen, R. D. Russell, "*Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*," Prentice-Hall, New York, 1988

[5] M. Berry, A. Sameh, "Multiprocessor Schemes for Solving Block Tridigonal Linear Systems," *The International Journal of Supercomputing Applications*, Vol. 2, 1988, pp. 37-57

[6] C. Cyphers, M. Paprzycki, A. Karageorghis, "High Performance Solution of Partial Differential Equations Discretized Using a Chebyshev Spectral Collocation Method," *Journal of Computational and Applied Mathematics*, Vol. 66, No. 1, 1996, pp. 71-80

[7] J. Foster, "Evolution of MIPS RISC Microprocessor Architecture," *Journal of Computing in Small Colleges*, Vol. 12, No. 4, 1997, pp. 215-229

[8] M. Paprzycki, C. Cyphers, "Level 3 BLAS Based Library for Block Tridiagonal Matrices," in: S. Elaydi, et. al. (eds.), *Advances in Difference Equations*, Gordon and Breach, Amsterdam, 1997, pp. 491-498

[9] M. Paprzycki, I. Gladwell, "Solving Almost Block Diagonal Systems Using Level 3 BLAS," *Proceedings of The Fifth SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, Philadelphia, 1992, pp. 52-62

[10] M. Paprzycki, A. Karageorghis, C. Cyphers, "Solving Structured Matrix Problems on a Cray Vector-Computer," Technical Report, University of Cyprus, *Mathematica Balcanica*, to appear.

[11] J. Varah, "On the Solution of Block-Tridiagonal Systems Arising from Certain Finite-Difference Equations," *Mathematics of Computation*, Vol. 26, 1972, pp. 859-868

[12] J. Varah, "Alternate row and column elimination for solving certain linear systems," SIAM J. Numer. Anal. Vol. 13, 1976, pp. 71-75