Advances in Difference Equations

Proceedings of the
Second International Conference
on Difference Equations

Veszprém, Hungary
August 7–11, 1995

Edited by

S. Elaydi
Trinity University, San Antonio, Texas, USA

I. Győri
University of Veszprém, Hungary

and

G. Ladas
University of Rhode Island, Kingston, USA

Gordon and Breach Science Publishers
Australia • Canada • China • France • Germany • India • Japan
Luxembourg • Malaysia • The Netherlands • Russia • Singapore
Switzerland • Thailand • United Kingdom

LEVEL 3 BLAS BASED LIBRARY FOR BLOCK TRIDIAGONAL MATRICES

M. PAPRZYCKI and C. CYPHERS
Department of Mathematics and Computer Science
University of Texas of the Permian Basin
Odessa, TX 79762 USA

Abstract Block tridiagonal linear systems arise from discretizations of a number of mathematical problems. Since tlfey can be solved in each step of an iterative process it is very important to solve them efficiently. A level 3 *BLAS* based library of subroutines performing basic operations (multiplication, decomposition, back substitution) is presented. The efficiency of individual subroutines is studied.

1. INTRODUCTION

Block tridiagonal linear systems (Figure 1) arise from the discretization of a number of mathematical problems [1,2,3]. In case of non-linear problems such systems are solved in each step of an iterative process. Since these systems are large and sparse a number of tearing-type strategies have been designed to solve them on parallel computers [2,3,4]. Typically, each processor solves a smaller problem of the same structure as the original problem. It is thus extremely important to have an efficient solver to perform factorizations on individual processors as this is the only way to achieve overall high efficiency of parallelization. In the case of a Newton-type iteration the linear system solution is the most costly part of the process, so any performance gain in this step will substantially reduce the total solution time.

$$M = \begin{pmatrix} A_1 & C_1 & & & & & \\ B_2 & A_2 & C_2 & & & & \\ & B_3 & A_3 & C_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & B_{n-1} & A_{n-1} & C_{n-1} \\ & & & & & B_n & A_n \end{pmatrix}$$

FIGURE 1 Block tridiagonal matrix; each block of size $k \times k$

In this paper, we will introduce a level 3 *BLAS* [5] based library of algorithms designed to perform basic numerical operations on block tridiagonal linear matrices. It consists of a matrix-matrix multiplication routine (for normal and transposed matrices), a Gaussian elimination based decomposer (where special care has been taken to eliminate the fill-in) and a back solver (for the normal and transposed system). The library has a unified, *LAPACK* [6] based interface. The performance of the proposed libraries is illustrated on a Cray J-916 supercomputer. Optimized *BLAS* kernels provided by the Cray were used. Timings were obtained using the *perftrace* utility. All results presented are averages of multiple runs.

2. BLOCK TRIDIAGONAL MATRIX LIBRARY

2.1. Matrix Multiplication

Matrix multiplication is performed as a series of calls to the level 3 *BLAS* matrix multiplication routine *_GEMM* [5]. Since the matrix is represented as rows of blocks of columns (similar to the representation of the almost block diagonal systems adopted in [7]) a total of n calls to *_GEMM* is required. For the transposed matrix the *transpose* option of the *_GEMM* is used. The first series of experiments was performed for the matrix-vector multiplication. We have observed the effects of increasing block size k on the obtained performance. The results for $n = 20$ row blocks and $k = 2, 3, \ldots, 37$ are summarized in Figure 2.

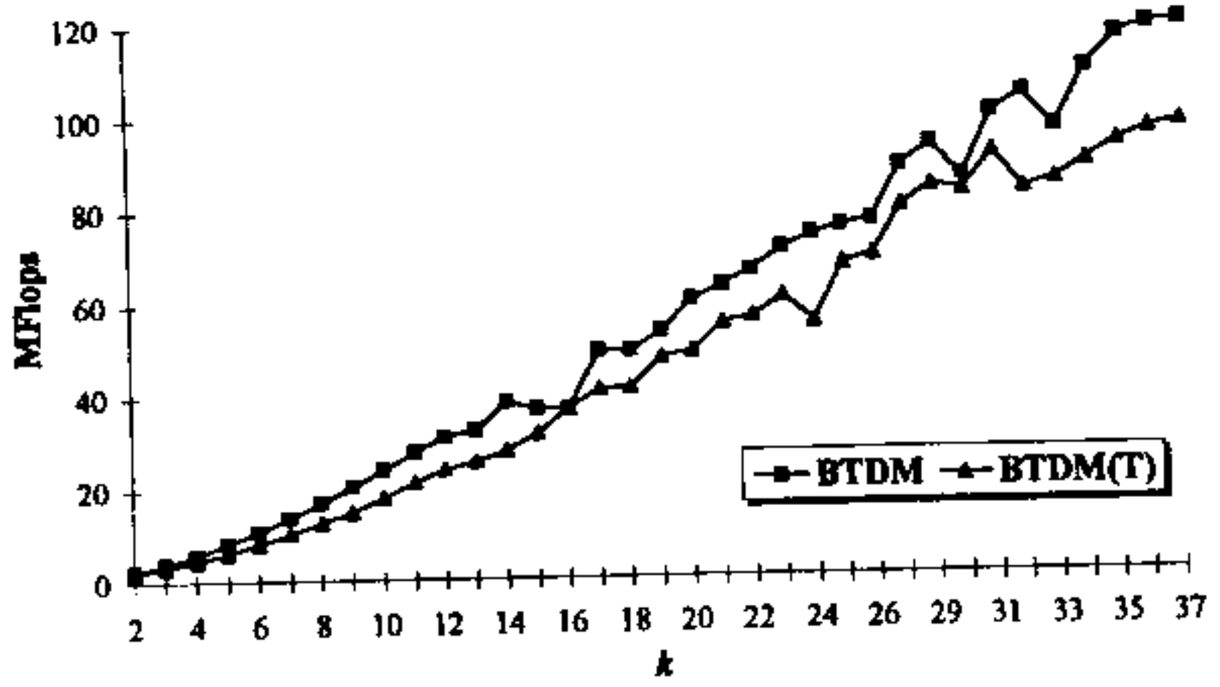

FIGURE 2 Matrix-vector multiplication; fixed n; increasing k; results in MFlops

The results presented above as well as additional experiments indicate that matrix-vector multiplication reaches approximately 120 MFlops (about 62% of the practical peak performance of 195 MFlops, see also [8]). This should be expected, as matrix-vector multiplication is a level 2 *BLAS* operation and thus less efficient than level 3 *BLAS* operations. In addition, transposed multiplication is less efficient because the data is being accessed in a row-oriented fashion causing memory section conflicts.

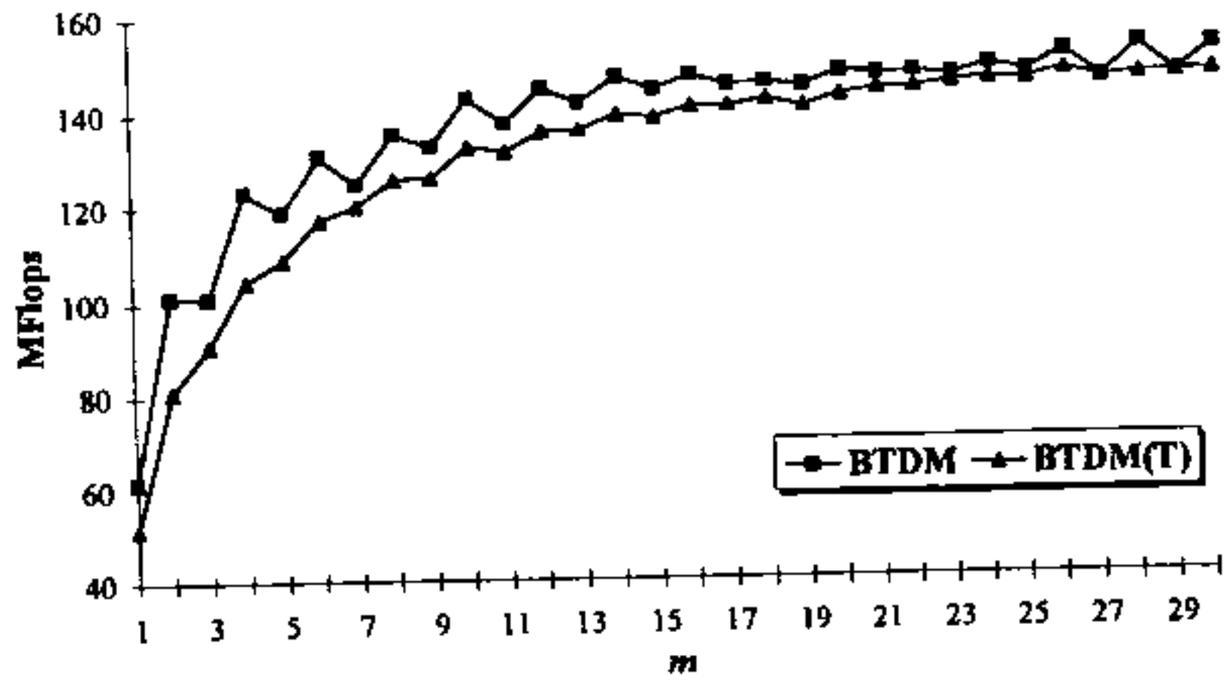

FIGURE 3 Matrix-matrix multiplication; fixed n and k; increasing m; results in MFlops

In the second series of experiments we have studied the effect of increasing the number of vectors that the block tridiagonal matrix is multiplied by (we have multiplied $M \times B$, where B is of size $kn \times m$). The results for $n = 20$, $k = 30$ and $m = 1, 2, \ldots, 30$ are summarized in Figure 3.

The results as well as our remaining experiments indicate that matrix-matrix multiplication reaches its peak performance when matrix B has a minimum of about $m = 20$ columns. We have also observed that as the number of columns of B increases, the difference between the performance of normal and transposed multiplication disappears. We have experimented with matrix-matrix multiplication, where B was a full matrix and increased the block size k. The performance steadily increased from about 18 MFlops at $k = 2$, to 187 MFlops for approximately $k = 40$ where it stabilized (reaching approximately 96% of the practical peak performance). We have discovered that transposed multiplication is sensitive to memory section conflicts for particular values of k; for instance a performance drop of 20 MFlops was observed at $k = 32$. Finally, we have detected that the number of blocks (n) has almost no effect on the performance of matrix multiplication from n > 7 on. This value of n is much smaller than what arises in computational practice.

2. 2. Matrix Factorization

We consider the solution of a linear system $M\underline{x} = \underline{b}$, where matrix M is block tridiagonal (Figure 1). Matrix M can be factorized using block-Gaussian elimination. Each factorization step i (except the last one) involves a 4-block submatrix of M

$$\begin{pmatrix} A_i & C_i \\ B_{i+1} & A_{i+1} \end{pmatrix}.$$

In the first step of factorization, block A_i will be $L_i U_i$ decomposed using Gaussian elimination with partial pivoting (row or column). To avoid the generation of fill-in, pivoting is performed only inside this block. The decomposition is performed using a call to the *LAPACK* routine *_GETRF* [6]. To optimize the performance of the algorithm, the decomposition is performed in a blocked fashion if k (the size of the block A_i) is larger than the optimal block size for a given machine or in an unblocked fashion otherwise. In the second step, block B_{i+1} is multiplied from the left by U_i^{-1} (resulting in B^*_{i+1}) and block C_i is multiplied from the right by L_i^{-1} (resulting in C_i^*). These two steps are accomplished by back substitution

using calls to the level 3 *BLAS* routine *_TRSM* [5]. In the final step, block A_{i+1} is updated by $B^*_{i+1} C_i^*$ by calling the level 3 *BLAS* routine *_GEMM*. The steps of the algorithm are then repeated starting from the just updated A_{i+1}. In the case of the last block (A_n), only the factorization step is applied. We have developed two versions of this algorithm: with row pivoting and with column pivoting.

It is clear that the algorithm can be unstable as the pivoting is applied only inside the diagonal blocks. The following result by Varah [1] provides the stability condition for the proposed algorithm. It can be shown that if matrix M is block diagonally dominant the block decomposition algorithm described above is numerically stable [1], where matrix M is *block diagonally dominant* with respect to the matrix norm $\|\cdot\|$ if:

$\|A_i^{-1}\|(\|B_i\| + \|C_i\|) \le 1$, where $i = 1, \ldots, n$.

Figure 4 presents the performance factorization routines (with row and column pivoting) for $n = 20$ blocks and increasing block sizes $k = 2, \ldots, 66$.

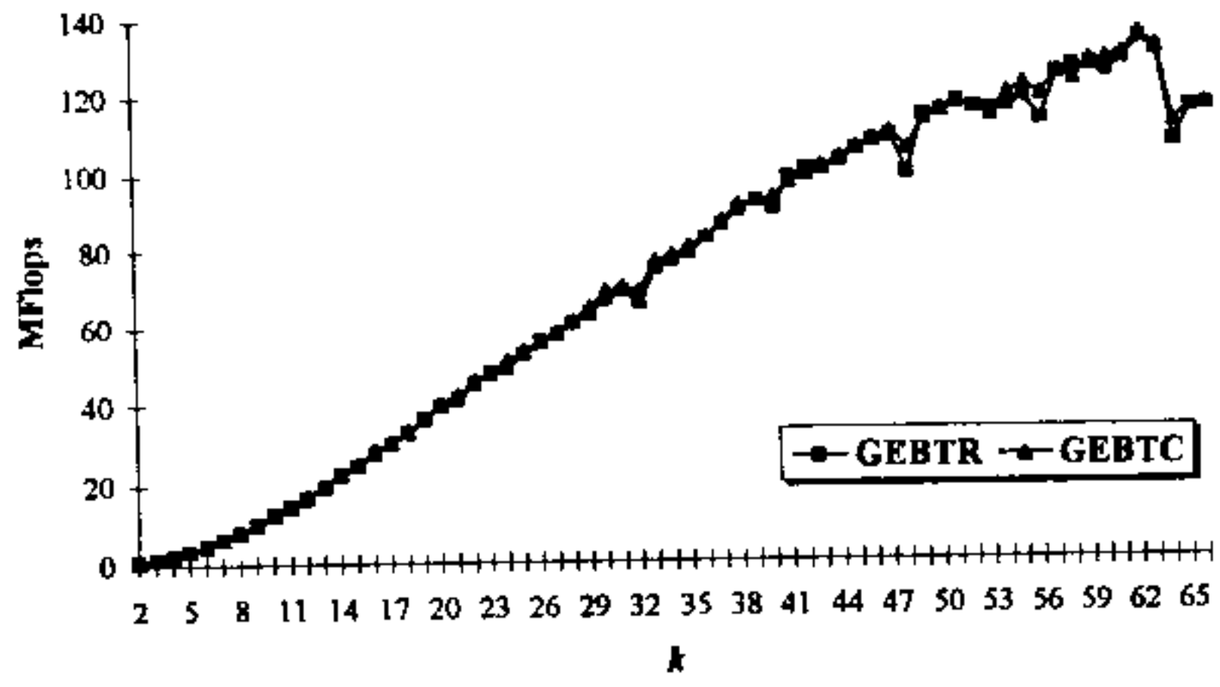

FIGURE 4 Matrix factorization; fixed n; increasing k; results in MFlops

The factorization routines are not as efficient as matrix multiplication. As the block size increases the efficiency of factorization improves from 0.8 MFlops for $k = 2$ to about 140 MFlops for $k = 63$ (approximately 72% of the practical peak performance). Both the row- and column-oriented versions behave similarly. Typical effects of memory section conflicts can be observed in both routines for $k = 32, 48, 64$ (the increase of the block size by 16) with the performance dip being

particularly visible at $k = 64$. The results are in agreement with [9, 10]. Our experiments indicate that the number of blocks n has a minimal effect on performance. As in the case of matrix multiplication, efficiency remains unchanged for $n > 7$.

2.3. Back Substitution

The back substitution consists of a number of calls the level 3 *BLAS* routine *_TRSM*. Four versions of the back solver corresponding to the two versions of factorization and the solution of a normal and transposed linear systems have been developed. The first series of experiments studied the performance of all four back solvers for $n = 20$, one right hand side and increasing block size $k = 2, 3, ..., 36$. The results are presented in Figure 5.

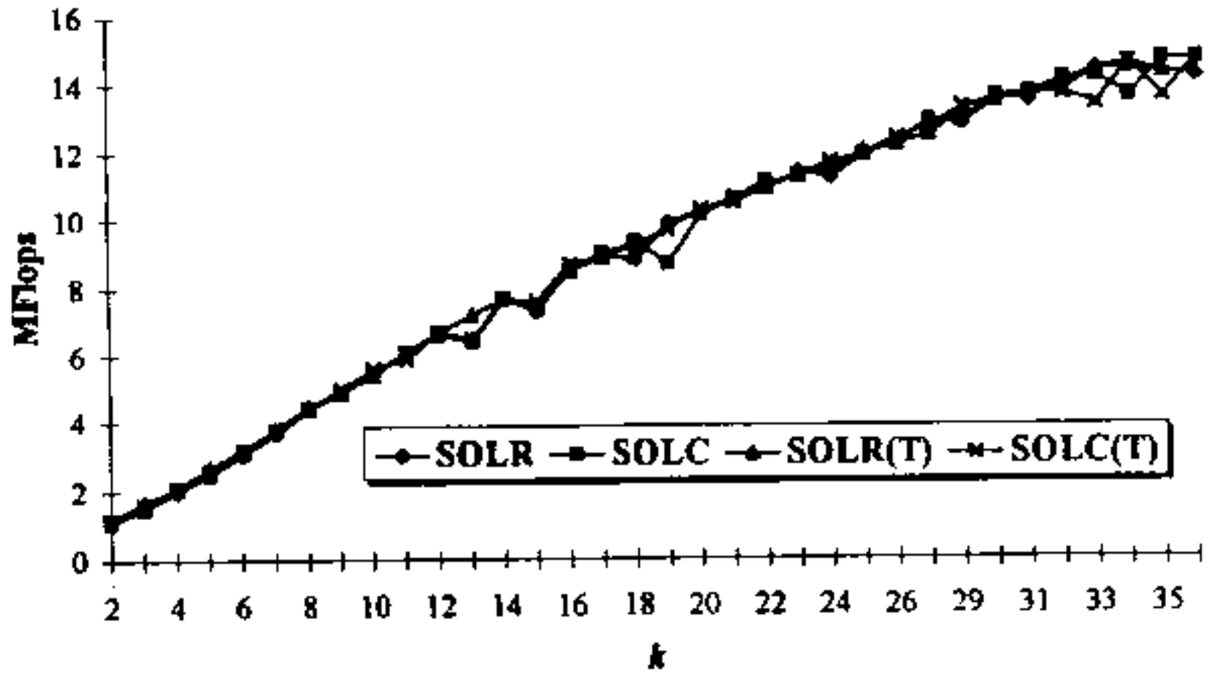

FIGURE 5 Back substitution; one right hand side; fixed n; increasing k; results in MFlops.

As the block size increases the performance of the back solver improves from 1.2 MFlops for $k = 2$ to almost 15 MFlops for $k = 36$. The performance is relatively low as, not only the sizes of individual blocks are relatively small, but also for one right hand side the performance is effectively reduced to the level 2 *BLAS* performance. All four back solvers behave similarly. No performance degradation is observed for the transposed systems. Figure 6 summarizes the performance results for all four back solvers for n = 10, k = 20 and an increasing number of right hand sides $NRHS = 1, 2, ..., 20$.

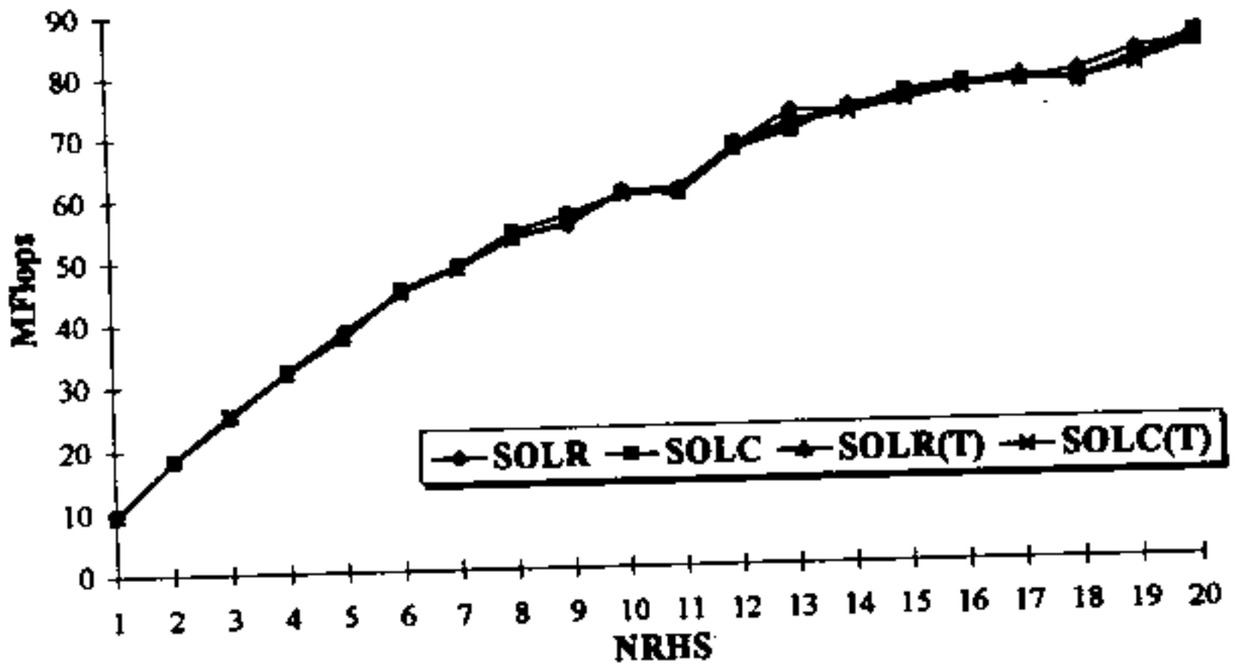

FIGURE 6 Back substitution; fixed n and k; increasing *NRHS*; results in MFlops.

As the number of right hand sides increases the performance of back solvers improves from 10 MFlops for $NRHS = 1$ to approximately 80 MFlops for NRHS = 20 (reaching approximately 41% of the practical peak performance). There is no performance difference between all four back solvers. Change in the number of right hand sides does not seem to lead to memory section conflicts.

3. CONCLUSION

A subroutine library for performing operations on block tridiagonal linear systems has been presented. For large enough blocks level 3 *BLAS* based subroutines reach more than 96% efficiency for matrix multiplication, 72% efficiency for factorization and 41% for back substitution. This makes them good candidates for the usage on one processor systems as well as parts of the parallel tearing-type algorithms. We were also able to observe the effects of memory section conflicts typical for the Cray J-916 architecture. The operations on transposed matrices are more sensitive to memory section conflicts than operations on normal matrices. (These effects can be avoided by choosing odd leading dimension of the matrices involved.) Finally, the performance for small block sizes is very poor which indicates the existence of a serious problem for the solution of the narrowly banded linear systems. In the next step, we plan to compare the performance of our approach to the highly optimized banded solvers available in various

libraries (block tridiagonal systems can be represented as banded systems) to see how successful this approach can be.

ACKNOWLEDGMENTS

A computer time grant from the Center for Scientific Computing of the University of Texas in Austin is kindly acknowledged.

REFERENCES

[1] J. Varah, On the Solution of Block-Tridiagonal Systems Arising from Certain Finite-Difference Equations, *SIAM J. Numer. Analysis*, **13**, (1976), 71-75.

[2] V. Mehrmann, Divide and Conquer Methods for Block Tridiagonal Systems, *Parallel Computing*, **19** (1993), 257-279.

[3] M. Berry and A. Sameh, Multiprocessor Schemes for Solving Block Tridiagonal Linear Systems, *The International J. of Supercomp. Appli.*, **2**, (1988), 37-57.

[4] P. Amodio, L. Brugnano and T. Politi, Parallel factorizations for Tridiagonal Matrices, *SIAM J. Numer. Analysis*, **30**, (1993), 813-823.

[5] J. Dongarra, J. Du Croz and S. Hammarling, A Set of Level 3 BLAS Basic Linear Algebra Subprograms, *Technical Report ANL-MCS-TM57*, Argonne National Laboratory, 1988.

[6] 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.

[7] C. Cyphers, M. Paprzycki and I. Gladwell, A Level 3 BLAS Based Solver for Almost Block Diagonal Linear Systems, *Software Report 92-3*, Southern Methodist University, 1992.

[8] M. Paprzycki and C. Cyphers, Multiplying Matrices on the Cray – Practical Considerations, *CHPC Newsletter*, **6**, (1991), 77-82.

[9] C. Cyphers and M. Paprzycki, Gaussian Elimination on a Cray Y-MP, *CHPC Newsletter*, **6** (1991), 43-47.

[10] M. Paprzycki, Comparison of Gaussian Elimination Algorithms on a Cray Y-MP, *Linear Algebra and Applications*, **172** (1992), 57-69.