PROCEEDINGS OF THE FIFTH SIAM CONFERENCE ON

PARALLEL PROCESSING FOR SCIENTIFIC COMPUTING

Edited by Jack Dongarra
Oak Ridge National Laboratory
and University of Tennessee

Ken Kennedy
Rice University

Paul Messina
California Institute of Technology
and Argonne National Laboratory

Danny C. Sorensen
Rice University

Robert G. Voigt
National Science Foundation

siam

Philadelphia 1992

CHAPTER 9

Using Level 3 BLAS to Solve Almost Block Diagonal Systems

M. Paprzycki*
I. Gladwell†

Abstract

We implement an efficient, stable almost block diagonal solver using level 3 BLAS. The algorithm is an implementation of the "best" sequential algorithm with blocked updates and blocked interchanges. These blocksizes are problem dependent. Also, the code calls blocked algorithms where their "established" blocksizes are system dependent. A numerical comparison is made with the most efficient level 2 BLAS implementation [3].

1. Introduction

In [13], the authors discuss the parallel solution of almost block diagonal (ABD) linear systems arising when a two-point boundary value problem (BVP) for ordinary differential equations (ODE's) is discretized along with its boundary conditions (BC's) using box-type difference schemes. The method extends a "tearing" algorithm for banded symmetric matrices presented by Dongarra and Johnsson [6] and Dongarra and Sameh [7], and later extended by Wright [16]. We make three observations. First, the usefulness of the tearing algorithm for solving this problem is still in question, since relatively small speed-ups are achieved both theoretically and in practice compared with a very efficient sequential algorithm. Second, the performance of the tearing algorithm improves when the number of internal blocks, k, increases, whereas it is essentially unaffected when the size of these internal blocks, n, increases. Third, treating the ABD system as banded and using a parallel banded system solver is computationally inefficient.

In current BVP algorithms (for example PASVA3 [10]), k, the number of meshpoints in the discretization of the BVP, may be large but will not grow unrestrictedly. The number of differential equations, n, may also be large. The arithmetic cost for ABD solvers is dominated by terms in k and n^3. The approach here provides significant speed-up when the number of equations, n, is large.

Our algorithm is based on the level 3 BLAS, Dongarra et al. [5]. It uses block decomposition with a careful interchange strategy. It is similar to the algorithm for Cholesky decomposition of symmetric positive definite banded matrices proposed by Mayes and Radicati [11] and for LU decomposition of a general matrix presented by Bischof et al. [2].

*Department of Mathematics and Computer Science, University of Texas of the Permian Basin, Odessa, TX 79762.
†Department of Mathematics, Southern Methodist University, Dallas, TX 75275.

We use three basic block matrix operations. Two are calls to level 3 BLAS routines:

(i) For a triangular T solve TB = A (or BT = A) for B using routine _TRSM.

(ii) Form a matrix-matrix product C = αAB + βC using routine _GEMM,

The last is a compound operation, utilizing simple level 3, 2 and 1 BLAS:

(iii) Compute a matrix decomposition A-> LU using routines _GETRR (a minor modification of the routine _GETRF, [2]) and _GETRC a column interchange version of _GETRF. Parallelization can be introduced at the level of block operations inside the level 3 BLAS.

2. Algorithm Description

2.1 Block Gaussian Elimination

Consider the ABD system arising when a two point BVP is solved using a box-type difference scheme, Keller [9] (see Figure 1). The size of each internal block is $n \times 2n$, and the BC-type blocks have sizes $q \times n$ and $(n-q) \times n$. (The BC's are assumed to be separated and q is the number of left BC's). For systems which arise from collocation methods [1] the algorithm is similar. Figure 1 presents a typical ABD structure with 5 internal blocks (corresponding to 4 internal meshpoints).

We use block Gaussian elimination. To assure stability and to avoid fill-in we use alternating row and column pivoting, similar to Varah [15]. Consider the structure in Figure 2.

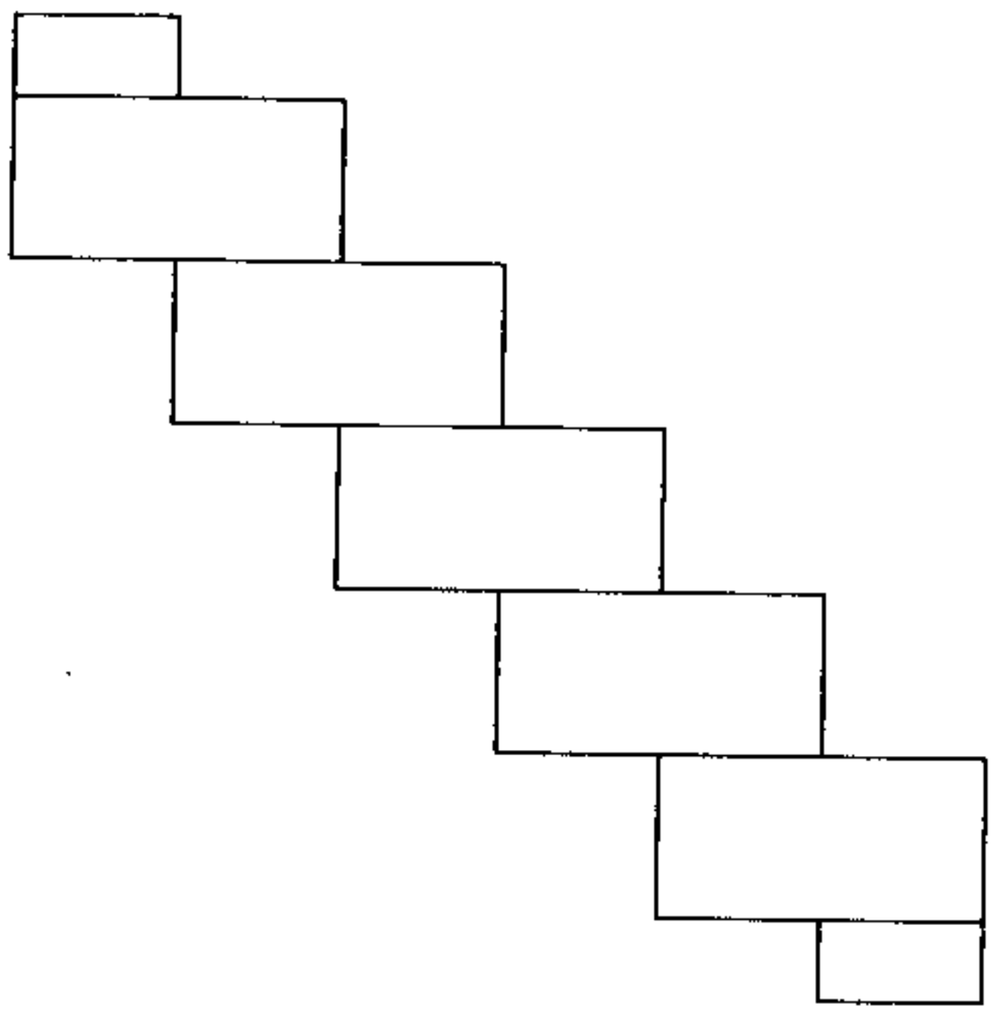

Figure 1. ABD system corresponding to 4 mespoints.

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} = \begin{bmatrix} L_{1,1} & & \\ L_{2,1} & L_{2,2} & \\ L_{3,1} & L_{3,2} & L_{3,3} \end{bmatrix} \begin{bmatrix} U_{1,1} & U_{1,2} & U_{1,3} \\ & U_{2,2} & U_{2,3} \\ & & U_{3,3} \end{bmatrix} = LU$$

$$LU = \begin{bmatrix} L_{1,1}U_{1,1} & L_{1,1}U_{1,2} & L_{1,1}U_{1,3} \\ L_{2,1}U_{1,1} & L_{2,1}U_{1,2} + L_{2,2}U_{2,2} & L_{2,1}U_{1,3} + L_{2,2}U_{2,3} \\ L_{3,1}U_{1,1} & L_{3,1}U_{1,2} + L_{3,2}U_{2,2} & L_{3,1}U_{1,3} + L_{3,2}U_{2,3} + L_{3,3}U_{3,3} \end{bmatrix}$$

Figure 2. Block LU structure for the ABD solver.

The block Gaussian elimination algorithm is:

1) $A_{1,1}$ (of an "established" size) is decomposed: $A_{1,1} -> L_{1,1}U_{1,1}$;

2) Blocks of L and U are calculated:

$$A_{1,2} = L_{1,1}U_{1,2} -> U_{1,2} = L_{1,1}^{-1}A_{1,2} , \quad (1)$$

$$A_{1,3} = L_{1,1}U_{1,3} -> U_{1,3} = L_{1,1}^{-1}A_{1,3} , \quad (2)$$

$$A_{2,1} = L_{2,1}U_{1,1} -> L_{2,1} = A_{2,1}U_{1,1}^{-1} , \quad (3)$$

$$A_{3,1} = L_{3,1}U_{1,1} -> L_{3,1} = A_{3,1}U_{1,1}^{-1} ; \quad (4)$$

3) Updates on the remaining blocks of A are performed:

$$A'_{2,2} = L_{2,1}U_{1,2} + L_{2,2}U_{2,2} -> L_{2,2}U_{2,2} = A_{2,2} - L_{2,1}U_{1,2} , \quad (1)$$

$$A'_{2,3} = L_{2,1}U_{1,3} + L_{2,2}U_{3,2} -> L_{2,2}U_{3,2} = A_{2,3} - L_{2,1}U_{1,3} , \quad (2)$$

$$A'_{3,2} = L_{3,1}U_{1,2} + L_{3,2}U_{2,2} -> L_{3,2}U_{2,2} = A_{3,2} - L_{3,1}U_{1,2} , \quad (3)$$

$$A'_{3,3} = L_{3,1}U_{1,3} + L_{3,2}U_{2,3} + L_{3,3}U_{3,3} -> L_{3,2}U_{2,3} + L_{3,3}U_{3,3} = A_{3,3} - L_{3,1}U_{1,3}; \quad (4)$$

4) The process is repeated starting from step 1) for a "new" $A_{1,1}$ block (which is either all of $A'_{2,2}$ or an upper left corner of $A'_{2,2}$ of the "established" size).

2.2 Block ABD decomposition

Using the block structure described above, we define a two phase per step algorithm for decomposing the ABD system. This is implemented in routine _ABDB3 (see Appendix 1 of

[14]). In each step (except the last) Phase I decomposes a part of an internal block equivalent to a BC-type block. Phase II then decomposes the part of the internal block occuring before the next BC-type block enters. Phase I uses column interchanges and decomposes a rectangular $q \times n$ block. Phase II uses row interchanges and decomposes a rectangular $n \times (n-q)$ block.

Figure 3 shows the block placed inside the system for execution of Phase I. There is the following correspondence between Figure 3 and Figure 2:

a) Blocks E and F were calculated in previous steps,

b) $D = \begin{bmatrix} A_{1,1} & A_{1,2} \end{bmatrix} \in R^{q \times n}$,

c) $A_{1,3} = 0 \in R^{q \times q}$ lies outside the ABD structure,

d) $C = \begin{bmatrix} A_{2,1} \\ A_{3,1} \end{bmatrix} \in R^{n \times q}$,

e) $U = \begin{bmatrix} A_{2,2} \\ A_{3,2} \end{bmatrix} \in R^{n \times (n-q)}$,

f) $Z = \begin{bmatrix} A_{2,3} \\ A_{3,3} \end{bmatrix} \in R^{n \times q}$.

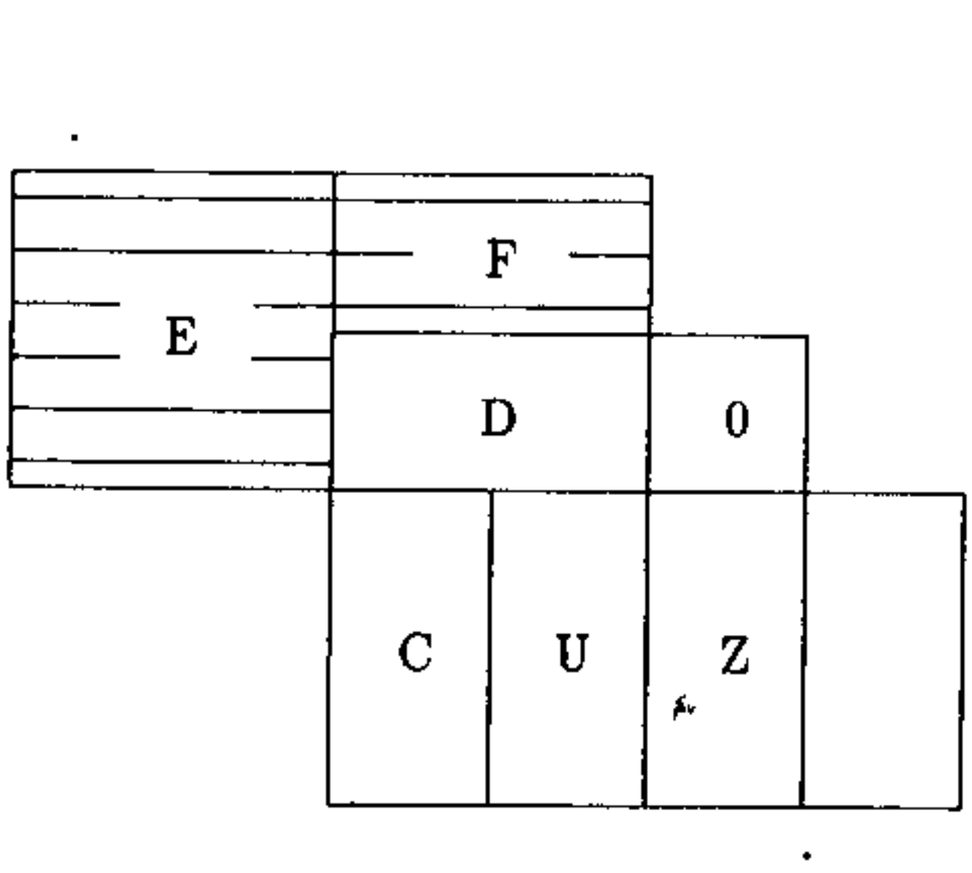

Figure 3. Block structure in Phase I

In Phase I:

1) D is decomposed (operations 1 and 2.1 from block Gaussian elimination). Column interchanges are performed internally in D;
2) These column interchanges are applied in C and U "below" and in F "above" D in a blocked fashion;
3) C is calculated (operations 2.3 and 2.4 from block Gaussian elimination);
4) U is updated (operations 3.1 and 3.3 from block Gaussian elimination).
Because $A_{1,3} = 0$ lies outside the ABD, operations 2.2, 3.2 and 3.4 of block Gaussian elimination are not performed.

Figure 4 shows the parts of the system already in place and the location of the block to be decomposed in Phase II. There is the following correspondence between Figure 4 and Figure 2:

a) Block E was calculated in Phase I.

b) $D = \begin{bmatrix} A_{1,1} \\ A_{2,1} \end{bmatrix} \in R^{n \times (n-q)}$,

c) $A_{3,1} = 0 \in R^{q \times q}$ lies outside the ABD structure,

d) $C = \begin{bmatrix} A_{1,2} & A_{1,3} \end{bmatrix} \in R^{(n-q) \times n}$,

e) $U = \begin{bmatrix} A_{2,2} & A_{2,3} \end{bmatrix} \in R^{q \times n}$,

f) $Z = \begin{bmatrix} A_{3,2} & A_{3,3} \end{bmatrix} \in R^{(n-q) \times n}$.

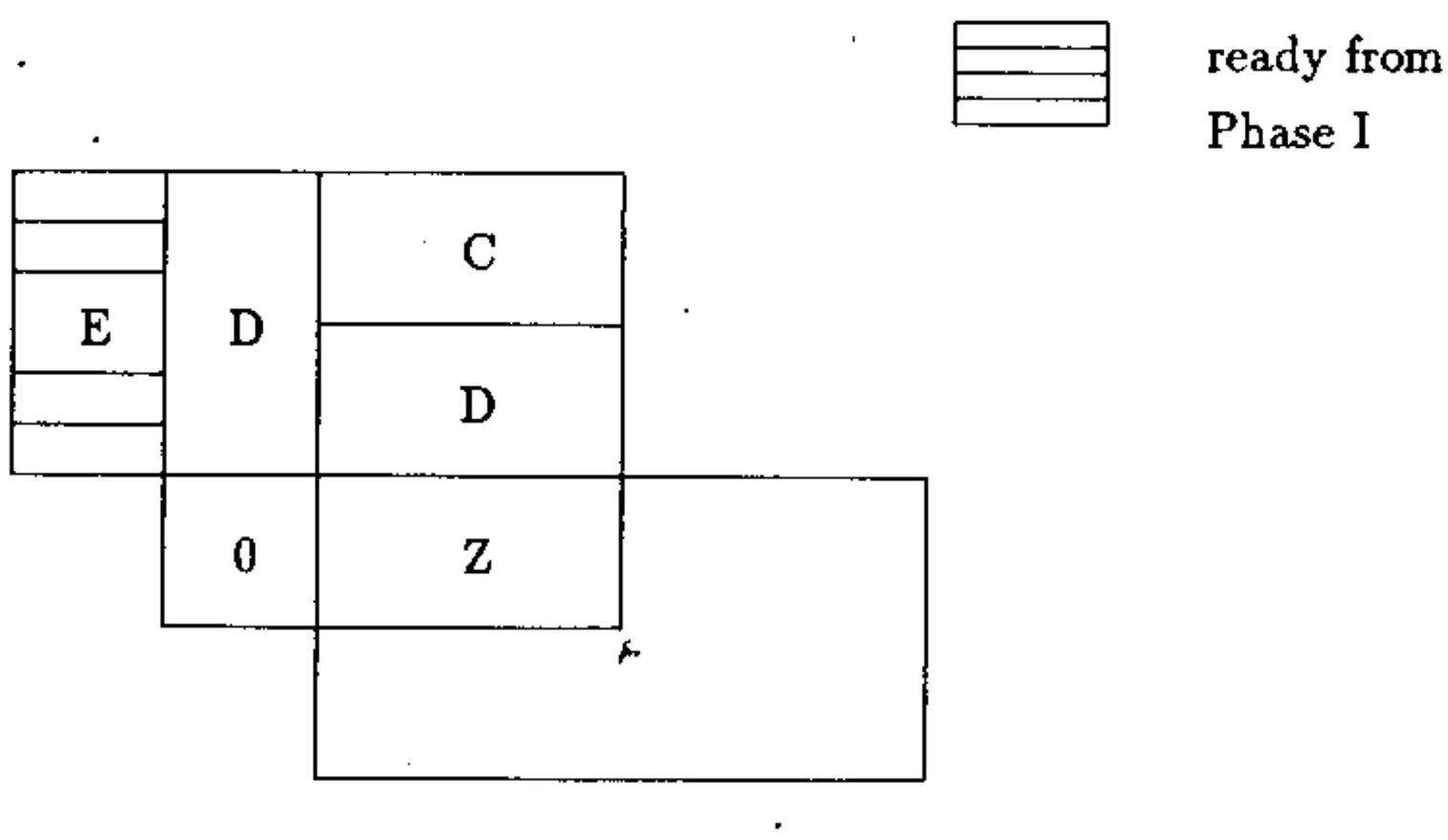

Figure 4. Block structure in Phase II

In Phase II:

1) D is decomposed. (operations 1 and 2.3 from block Gaussian elimination). Row interchanges are performed internally in D;
2) These row interchanges are applied in C and U on the "right" and E on the "left" in a blocked fashion;
3) C is calculated (operations 2.1 and 2.2 from block Gaussian elimination);
4) U is updated (operations 3.1 and 3.2 from block Gaussian elimination).
   Because $A_{3,1} = 0$ lies outside the ABD, operations 2.4, 3.3 and 3.4 of block Gaussian elimination are not performed.

This ends one step of the algorithm. Figure 5 shows the resulting system. Observe that the undecomposed (but updated) part of the ABD has the same structure as the initial system.

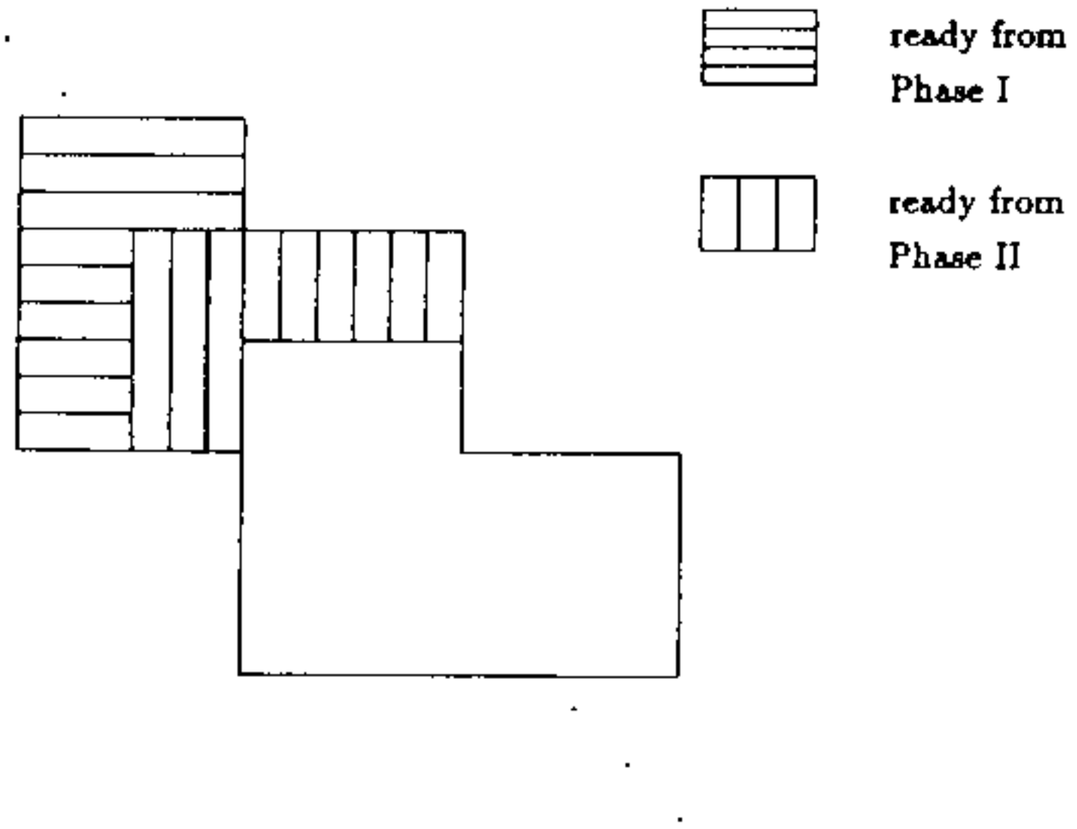

Figure 5. System after one step of decomposition

The block LU decompositions in Phases I and II are performed by calls to _GETRC, and _GETRR. Blocks of L and U are calculated by calls to _TRSM. The remaining part of the blocks in Phases I and II are updated by calls to _GEMM. In the last step both phases are performed on a reduced sized block (see [14]).

2.3 Block backsubstitution

The decomposition above has the form:

$$A = PLUQ$$

where L is unit lower triangular, U is upper triangular, and P and Q are permutation matrices. The solution of $A\underline{x} = \underline{b}$ will be calculated by calling routine _ABSB3, [13] which implements the sequence:

(a) $P\underline{r} = \underline{b}$, (b) $L\underline{s} = \underline{r}$, (c) $U\underline{t} = \underline{s}$, (d) $Q\underline{x} = \underline{t}$.

Only for multiple right hand sides is it possible to obtain any parallelization in steps (a) and (d).

We use a level 3 BLAS algorithm for steps (b) and (c). For multiple right hand sides this is a blocked algorithm. For a single right hand side our code is equivalent to using level 2 BLAS. Figure 6 shows two typical rows of the block division of the ABD system after decomposition. Forward substitution proceeds in a block by block fashion. Substitution for the top BC-type block uses a call to _TRSM. The i-th internal block is treated by updating the appropriate part of the right hand side using information from R_i by a call to _GEMM then solving the system $L_i\tilde{g} = \tilde{r}$ by a call to _TRSM ($\tilde{g}$ and $\tilde{r}$ represent appropriate parts of g and r from (b) above). Back substitution also proceeds a block by block fashion. Back substitution for the bottom BC-type block uses a call to _TRSM. The i-th internal block is treated by updating the appropriate part of the RHS using information from T_i by a call to _GEMM then solving the system $U_i\tilde{t} = \tilde{g}$ by a call to _TRSM ($\tilde{t}$ and $\tilde{g}$ represent appropriate parts of t and g from (c) above).

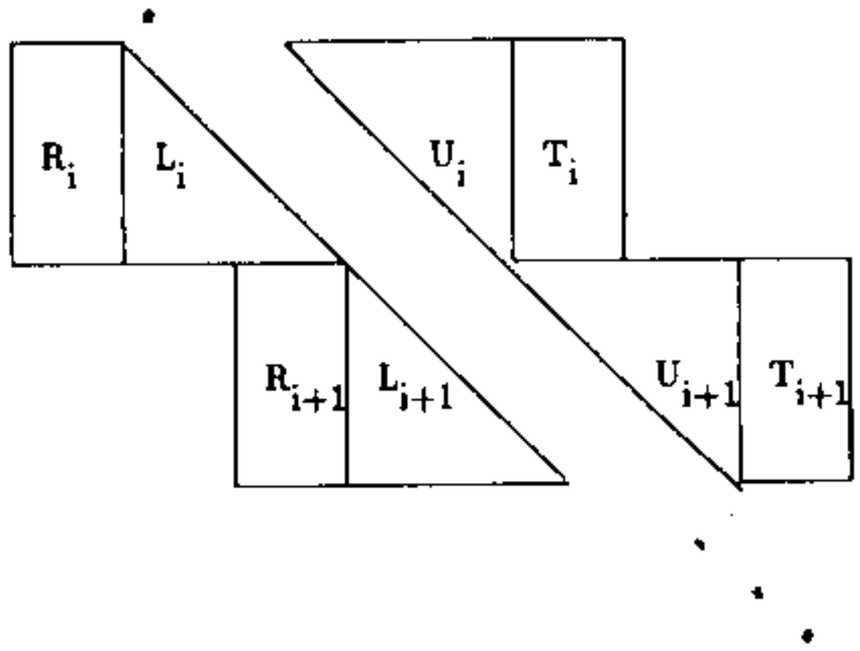

Figure 6. Block subdivision for back substitution

Using the decomposition computed in _ABDB3 we can also solve

$$A^T x = b$$

via

$$Q^T U^T L^T P^T x = b$$

for any right hand side b using _ABTB3 and a sequence of steps similar to (a) – (d) above. Thus we can estimate the condition of A using the algorithm of Higham [8]. This is a necessary with our code since it, like -GETRF, only exits with an error indication when exact (zero pivot) singularity is encountered.

3. Practical considerations

3.1 Routines _GETRR and _GETRC

Routines _GETRR (row interchange) and _GETRC (column interchange) are special cases of the routine _GETRF implementing block Gaussian elimination. Each uses the block structure in Figure 7:

(1) The size of the block $A_{1,1}$ is "established" for a given computer architecture,

(2) $A_{1,1}$ is decomposed: $A_{1,1} -> L_{1,1}U_{1,1}$, with interchanges applied internally;

(3) Interchanges are applied to the outside blocks;

(4) $U_{1,2} = L_{1,1}^{-1}A_{1,2}$ and $L_{2,1} = A_{2,1}U_{1,1}^{-1}$ are calculated;

(5) $A_{2,2}$ is updated, $A'_{2,2} = A_{2,2} - L_{2,1}U_{1,2}$;

(6) the process is repeated starting from 2) for a "new" $A_{1,1}$ block (which is either $A'_{2,2}$ or an upper left part of $A'_{2,2}$ with size established in (1)).

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} = \begin{bmatrix} L_{1,1} & \\ L_{2,1} & L_{2,2} \end{bmatrix} \begin{bmatrix} U_{1,1} & U_{1,2} \\ & U_{2,2} \end{bmatrix} = LU$$

$$LU = \begin{bmatrix} L_{1,1}U_{1,1} & L_{1,1}U_{1,2} \\ L_{2,1}U_{1,1} & L_{2,1}U_{1,2} + L_{2,2}U_{2,2} \end{bmatrix}$$

Figure 7. Block LU decomposition of $A \in R^{M \times N}$.

Thus in our algorithm different blocksizes will be used at different levels of the program. We have problem dependent blocksizes introduced by the algorithm as described in Sections 2.3 and 2.3. However the level 3 BLAS and _GETRF (and our successor routines _GETRR and _GETRC) work internally with their own (machine and/or routine dependent) established blocksizes. The latter are limited by the problem dependent blocksizes but are usually smaller.

The algorithms described above are listed in [14]. There, Appendix 1 contains the decomposition routine DABDB3 and the solution routines DABSB3 and DABTB3 (for a matrix and its transpose). Appendix 2 contains the row interchange routine DGETRR and the column interchange routine DGETRC.

4. Numerical Results

We have compared our block ABD solver with the NAG Library ABD decomposition and solution routines, F01LHF and F04LHF respectively, as described in [3]. The latter use level 2 and level 1 BLAS wherever appropriate. Our solver, DABDB3 for decompositon and DABSB3 for solution, calls level 3 BLAS and in addition some level 2 and 1 BLAS. Our comparisons were made on a Cray Y-MP 8/864 at the University of Texas at Austin. All the software tested calls CRAY assembly coded BLAS. In our test these BLAS are running on one processor. Of course, if the level 3 BLAS were implemented on a multiprocessor version, the new solver would clearly be expected to outperform the current single processor version. Here, we show that the

"overhead" of using level 3 BLAS at various "established" blocksizes leads to no penalty.

We only consider large problems with a significant number of large interior blocks. One variable to be considered in the number of top BC's, q. Another is the "established" size (ES) to be used for the Gaussian elimination. We will show the effect of independent variations in q and ES. Our results are given in megaflops. The reader should bear in mind that the theoretical maximum megaflop rate is for one processor of the Cray Y-MP8/864 is 333. Also note that the Cray Library matrix multiplication rountine, calling level 3 BLAS, has a megaflop rate of about 315 for large problems, [12]. This rate is a realistic upper limit on performance. In Table 1 we consider the effect of varying ES for a fixed, typical value of q. We consider the decomposition routines DABDB3 and F01LHF only, with k = 50 blocks, with blocksizes n = 400 and 401, and with q = 100 left BC's. All computed values are the average of fifty runs on a lightly loaded machine. We see a similar pattern for other large problems. In most cases, DABDB3 outperforms F01LHF by between 1% and 10% whilst performing precisely the same number of arithmetic operations. Hence, the speed-up resulting from using higher level BLAS at least compensates for the use of a blocked algorithm.

	n = 400		n = 401	
ES	DABDB3	F01LHF	DABDB3	F01LHF
1	271	260	287	269
64	272	260	289	269
65	274	260	287	269
128	275	260	287	269
129	275	260	285	269
256	271	260	286	269
257	271	260	287	269

Table 1 Effect of "established" blocksize, ES.

In Table 2 we fix n = 400, k = 50, ES = 420 and we vary q. The values in Table 2 are quite typical. Observe the variation in performance of the solution routine F04LHF as q varies. For other problem sizes, we observe a similar phenomenon of increasing megaflop rate for increaing q. This behavior is unexpected and suggests a reconsideration of the coding of F04LHF.

q	DABDB3	F01LHF	DABSB3	F04LHF
1	261	261	120	82
68	269	264	120	87
134	273	267	120	93
200	273	266	120	99
266	270	266	121	107
332	262	266	121	116
399	250	266	121	127

Table 2 Effect of variations in q

5. Conclusion

We have developed a parallel algorithm for ABD systems. This algorithm is based on block linear algebra and is implemented using level 3 BLAS. This algorithm maximizes exploitation of the underlying structure of the system. It has the same stability properties as other Gaussian elimination algorithms for ABD's with row and column interchanges. We have shown

that there is no additional overhead in comparison with an efficient level 2 BLAS implementation.

Acknowledgement

The authors wish to thank Cliff Cyphers who was responsible for many of the computations on the Cray Y-MP.

References

1. U. Ascher, J. Christiansen and R.D. Russell, Collocation Software for Boundary Value ODEs, ACM Trans. Math. Soft., 7, 1981, pp 209-229.

2. C. Bischof, J. Demmel, J. Dongarra, J.J. Du Croz, A. Greenbaum, S. Hammarling and D. Sorensen, LAPACK Working Note # 5 Provisional Contents, Report ANL-88-38, Argonne National Laboratory, 1988.

3. R. Brankin and I. Gladwell, Codes for Almost Block Diagonal Systems, Computers Math. Applic., 19, 1990, pp. 1-6.

4. C. DeBoor and R. Weiss, SOLVEBLOK: A Package for Solving Almost Block Diagonal Linear Systems, ACM Trans. Math. Soft., 6, 1980, pp 80-87.

5. J. Dongarra, J.J. Du Croz, I. Duff and S. Hammarling, A Set of Level 3 Basic Linear Algebra Subprograms, Report ANL-MCS-P88-1, Argonne National Laboratory, 1988

6. J. Dongarra, and L. Johnsson, Solving Banded Systems on a Parallel Processor, Parallel Comput., 5, 1987, pp 219-246.

7. J. Dongarra, and A.H. Sameh, On Some Parallel Banded System Solvers, Parallel Comput., 1, 1984, pp 223-235.

8. N.J. Higham, FORTRAN Codes for Estimating the One-Norm of a Real or Complex Matrix, With Applications to Condition Estimation, ACM Trans. Math. Softw. 4, 1988, pp 381-396.

9. H.B. Keller, Numerical Solution of Two Point Boundary Value Problems, SIAM, Philadelphia, 1976.

10. M. Lentini, and V. Pereyra, An Adaptive Finite-Difference Solver for Nonlinear Two-Point Boundary Problems With Mild Boundary Layers, SIAM J. Numer. Anal., 14, 1977,.pp 91-111.

11. P. Mayes, and G. Radicati, Banded Cholesky Factorization Using Level 3 BLAS, LAPACK Working Note #12, Mathematics and Computer Science Division, Technical Memorandum No. 34 (ANL/MCS-TM-134), Argonne National Laboratory, 1988.

12. M. Paprzycki and C. Cyphers, Multiplying Matrices on the Cray – Practical Considerations, CHPC Newsletter, University of Texas at Austin, 6, 1991, pp. 77-82.

13. M. Paprzycki and I. Gladwell, Solving Almost Block Diagonal Systems on Parallel Computers, Parallel Comput., 17, 1991, 133-153.

14. M. Paprzycki, and I. Gladwell, Solving Almost Block Diagonal Systems Using Level 3 BLAS, SMU Math. Rept. No. 90-4, 1990.

15. J.M. Varah, Alternate Row and Column Elimination for Solving Certain Linear Systems, SIAM J. Numer. Anal., 13, 1976, pp 71-75.

16. S.J. Wright, Parallel Algorithm for Banded Linear Systems, 1990, submitted for publication.