Factorizations of Band Matrices Using Level 3 BLAS

Jeremy Du Croz
N.A.G. Ltd

Peter Mayes
N.A.G. Ltd

Giuseppe Radicati
IBM ECSEC Rome

Abstract

This paper describes block algorithms for Cholesky and *LU* factorizations of band matrices. The algorithms can be implemented using Level 3 BLAS. Although they involve some extra data movement and extra arithmetic operations, on some machines they can outperform unblocked algorithms which use Level 2 BLAS. We present performance measurements to demonstrate this, on an IBM 3090 VF, a Cray 2, and a Siemens VP 400.

1 Introduction

This paper describes the development of "block algorithms" for factorizations of band matrices. By block algorithms, we mean algorithms that are rich in matrix-matrix operations [9]. On many modern high performance computers, block algorithms have been found to be capable of giving superior performance to the traditional algorithms—as implemented in Linpack [4], for example—which we refer to as "unblocked".

The Linpack software was written using Level 1 BLAS [11], but can easily be rewritten to use Level 2 BLAS [6], with improved performance especially on vector-processing machines. Software for block algorithms requires Level 3 BLAS [5].

Our algorithms have been developed as a contribution to the LAPACK project [3]. The original plans for LAPACK [1] did not envisage that block algorithms would be useful for factorizing band matrices. However we show that on some machines they are significantly faster than unblocked algorithms—as long as the bandwidth is not too small.

Section 2 describes a block algorithm for Cholesky factorization of a symmetric positive-definite band matrix (an earlier report on this was given in [12]). Section 3 describes a block algorithm for *LU* factorization of a general band matrix. Section 4 presents timings measured on an IBM 3090/VF, a Cray XMP, a Cray 2, and a Siemens VP 400.

Our software assumes that band matrices are stored compactly in the same band storage format as is used in Linpack — that is, columns of the matrix are stored in columns of the array, and diagonals of the matrix are stored in rows of the array. However, we describe the algorithms here in mathematical terms, with reference to the matrix elements rather than the array elements.

2 Cholesky factorization

We assume that we are given the upper triangle of a symetric positive-definite band matrix A of order n, with k superdiagonals. We wish to compute the Cholesky factorization $A = U^T U$, with U overwriting the upper triangle of A.

The proposed LAPACK routines for Cholesky factorization—for both full and band matrices—allow options for working with either the upper or the lower triangle. However for simplicity, we describe here only the upper triangular version of the algorithm.

There are three different ways of organizing the Cholesky factorization of a full matrix (see [12]), and they can all be adapted to the factorization of a band matrix.

Our block algorithm is based on a "right-looking" variant, and is derived by equating submatrices in the following equation. We use a fixed block size nb. A_{11} is the leading nb by nb submatrix.

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix} = \begin{pmatrix} U_{11}^T & 0 \\ U_{12}^T & U_{22}^T \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} = \begin{pmatrix} U_{11}^T U_{11} & U_{11}^T U_{12} \\ U_{12}^T U_{11} & U_{12}^T U_{12} + U_{22}^T U_{22} \end{pmatrix}$$

Hence for a full matrix, the algorithm performs the following matrix-matrix operations:

1. $A_{11} \rightarrow U_{11}^T U_{11}$
2. $U_{12} \leftarrow (U_{11}^T)^{-1} A_{12}$
3. $A'_{22} \leftarrow A_{22} - U_{12}^T U_{12}$

The algorithm then repeats the process on the updated submatrix A'_{22}.

If the matrix is a band matrix, then the submatrices A_{12} and A_{22} are truncated by the band structure, and only part of A_{22} is affected by the update. A typical stage in the algorithm is shown in Figure 1, where the "active block" within the band has been partitioned 3 by 3. We assume that the block size nb does not exceed the semi-bandwidth k—otherwise we use the unblocked algorithm.

The submatrix A_{13} lies partly outside the band. In order to operate on it using calls to Level 3 BLAS, we copy it into a square internal work array, of size nb by nb, with the upper triangle explicitly set to zero.

Each major stage in the computation consists of the following steps, using calls to the named Level 3 BLAS routines. U_{ij} denotes the submatrix of the Cholesky factor U which corresponds to A_{ij} and overwrites A_{ij} in the computation.

1. factorize the diagonal block A_{11} as $U_{11}^T U_{11}$ (using Level 2 BLAS)

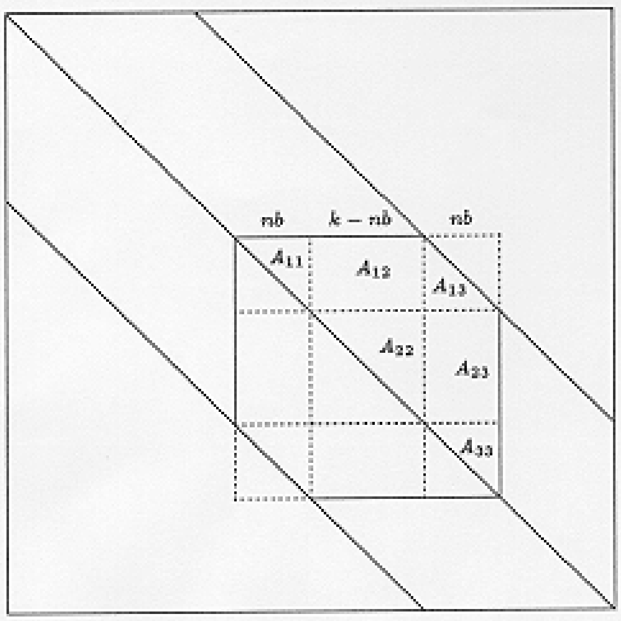

Figure 1: Block Cholesky factorization of a band matrix

2. $U_{12} \leftarrow (U_{11}^T)^{-1} A_{12}$ (using DTRSM)
3. $A'_{22} \leftarrow A_{22} - U_{12}^T U_{12}$ (using DSYRK)
4. copy the lower triangle of A_{13} into the work array
5. $U_{13} \leftarrow (U_{11}^T)^{-1} A_{13}$ (using DTRSM, with U_{13} overwriting A_{13} in the work array)
6. $A'_{23} \leftarrow A_{23} - U_{12}^T U_{13}$ (using DGEMM)
7. $A'_{33} \leftarrow A_{33} - U_{13}^T U_{13}$ (using DSYRK)
8. copy the lower triangle of U_{13} back into the main array in band storage.

Some of these steps are omitted when the active block hits the lower right hand corner of the matrix. When the bandwidth is large, the biggest fraction of the work is performed by the call to DSYRK in step 3.

The overhead of the block implementation includes setting the upper triangle of the work array to zero (done only once), copying the lower triangle of A_{13} into the work array, performing extra arithmetic operations with zero elements of A_{13} or U_{13}, and copying U_{13} back into the main array.

3 *LU* factorization

Now we assume that we are given an unsymmetric matrix of order n with kl subdiagonals and ku superdiagonals. An additional kl superdiagonals may be generated because of fill-in during the factorization. The fill-in results from the need to perform row-interchanges.

For simplicity in the description, we assume that the additional kl superdiagonals are used from the start, so that the number of superdiagonals is $ku' = ku + kl$. In practice the fill-in usually builds up gradually as the algorithm proceeds, and the code takes advantage of this to avoid unnecessary work in the early stages.

There are different ways of organizing the LU factorization of a full matrix (see, for example, [2], [13]). However the need to preserve band structure, while at the same time performing row interchanges, enforces the use of a "right-looking" variant for the factorization of a band matrix. Moreover at each stage, the row interchanges are applied only to the columns to the right of the current pivot column, not to the whole matrix.

A block algorithm can be derived by the same approach as was used in Section 2 for Cholesky factorization. A typical stage in the algorithm is illustrated in Figure 2. We assume that the block size nb does not exceed the number of subdiagonals, kl, which in turn is less than the number of superdiagonals, ku', allowing for fill-in.

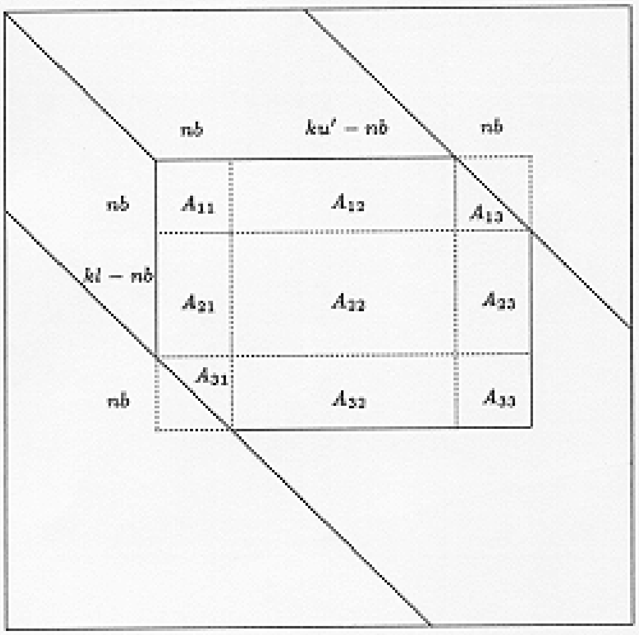

Figure 2: Block LU factorization of a banded matrix

The submatrices A_{13} and A_{31} lie partly outside the band; they are copied into internal work arrays with the elements outside the band set to zero, before they are operated on.

A typical stage consists of the following steps. L_{ij} and U_{ij} denote the submatrices of the factors L and U which correspond to A_{ij} and overwrite A_{ij} in the computation.

1. Perform an LU factorization with row interchanges, on the block column

$$\begin{pmatrix} A_{11} \\ A_{21} \\ A_{31} \end{pmatrix}$$

   using Level 2 BLAS. This block column is a rectangular band matrix, whose upper triangle is normally full. It is factorized using the standard unblocked algorithm, with one difference: the row interchanges must be applied to all columns in the block before using L_{11}, L_{21} and L_{31} in steps 2 to 8; this introduces nonzero elements into the lower triangle of L_{31}. So during this step, L_{31} is copied into a square work array, and is stored as a full matrix.

2. $U_{12} \leftarrow L_{11}^{-1} A_{12}$ (using DTRSM)

3. $A'_{22} \leftarrow A_{22} - L_{21} U_{12}$ (using DGEMM)

4. $A'_{32} \leftarrow A_{32} - L_{31} U_{12}$ (using DGEMM)

5. copy the lower triangle of A_{13} into a second square work array, with zeros stored in its upper triangle

6. $U_{13} \leftarrow L_{11}^{-1} A_{13}$ (using DTRSM)

7. $A'_{23} \leftarrow A_{23} - L_{21} U_{13}$ (using DGEMM)

8. $A'_{33} \leftarrow A_{33} - L_{31} U_{13}$ (using DGEMM)

9. copy the lower triangle of U_{13} back into the main array according to the band storage scheme

10. partially undo the row interchanges which were performed on the block column

$$\begin{pmatrix} L_{11} \\ L_{21} \\ L_{31} \end{pmatrix}$$

    to restore the band structure, and at the same time copy the nonzero elements of L_{31} back into the main array according to the band storage scheme.

Some of these steps are omitted when the active block hits the lower right hand corner of the matrix. Step 3 accounts for the largest fraction of the work when the bandwidth is large.

4 Performance measurements

Because the block algorithms involve some extra data movement and some extra arithmetic operations, it is not obvious that they will be faster than the unblocked algorithms, even on machines where Level 3 BLAS perform significantly faster than Level 2 BLAS.

We report measurements on an IBM 3090-E/VF, a Cray XMP, a Cray 2 and a Siemens VP 400. We chose these machines because efficient implementations of the necessary Level 2 and Level 3 BLAS are available on all of them—in the ESSL Library [7] on the IBM machine, in the SCILIB Library on the Cray machines, and developed by Geers [8] and Grasemann [10] for the Siemens machine.

We measured the speed of the routines for a fixed order of matrix ($n = 2048$), and with varying semi-bandwidth (k) and block-size (nb).

4.1 Performance of Cholesky factorization

Figure 3 shows the speed on an IBM 3090-E/VF of 4 different routines for Cholesky U^TU factorization:

1. the Linpack routine DPBFA
2. an unblocked algorithm calling Level 2 BLAS
3. the proposed LAPACK routine DPBTRF implementing the block algorithm described in Section 2 with a block size of 32, and calling Level 3 BLAS
4. the ESSL routine DPBF (which actually performs an LDL^T factorization, in terms of the storage scheme that we are assuming, but its results are included for comparison)

It is clear that for bandwidths of 100 or more the LAPACK block algorithm easily outperforms the unblocked (Level 2 BLAS) algorithm, despite the extra work done by the block algorithm.

	speed unblocked	speed blocked	optimum block-size	crossover value of k
IBM 3090 VF	22	40	32	80
Cray XMP	160	167	16	64
Cray 2	188	223	32	80
Siemens VP 400	57	123	64	80

Table 1: Performance of U^TU factorization

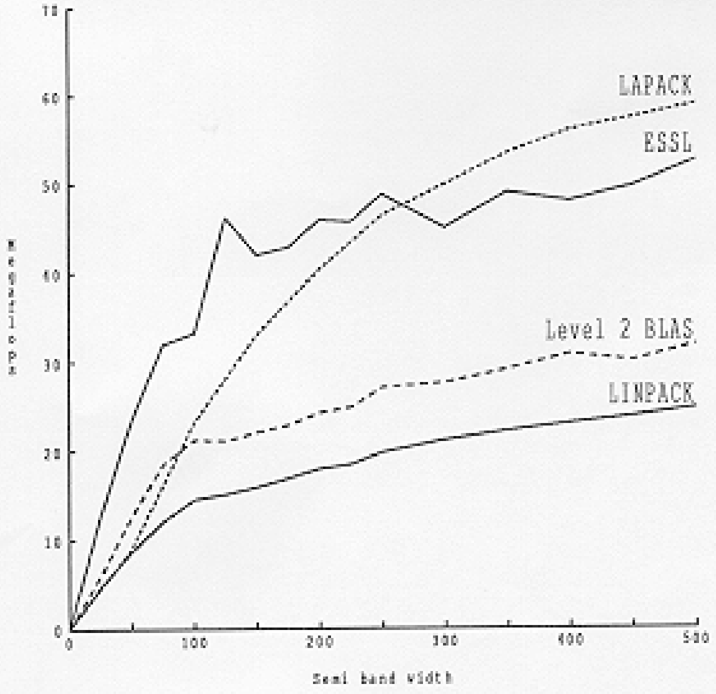

Figure 3: Speed of banded U^TU factorization on an IBM 3090-E/VF

Table 1 summarises results from different machines. It gives the speed of the unblocked algorithm, the speed of the blocked algorithm (both for $k = 192$), the optimal block-size, and the crossover value of k—the value of k beyond which the block algorithm was faster. The last two values are approximate. As the block size increases (with fixed bandwidth), the performance of the block algorithm reaches a plateau, beyond which it slowly declines. There is a slight tendency for the optimum block size to increase as the bandwidth increases.

The blocked algorithm is clearly superior—for large enough bandwidths— on the IBM, Cray 2 and Siemens machine. In fact the obvious superiority on the Siemens machine results from the poor performance of the unblocked algorithm. A different variant of the unblocked algorithm (relying on the Level 2 BLAS routine DSYR instead of DTRSV) was much faster; the block algorithm only became faster for k greater than 200. We did not expect to see a significant superiority on a Cray XMP, since on that machine Level 2 and Level 3 BLAS run at roughly the same speed.

4.2 Performance of LU factorization

We set $kl = ku = k$ and used an approximate value of $4nk^2$ for the number of floating-point operations in order to calculate megaflop rates; this may be a slight over-estimate since it assumes that maximum fill-in occurs right at the start of the algorithm.

Figure 4 shows the speed of both the blocked algorithm (NB = 32), calling Level 3 BLAS, and an unblocked algorithm (NB = 1), calling Level 2 BLAS, on an IBM 3090-E/VF; the blocked algorithm used a block-size of 32.

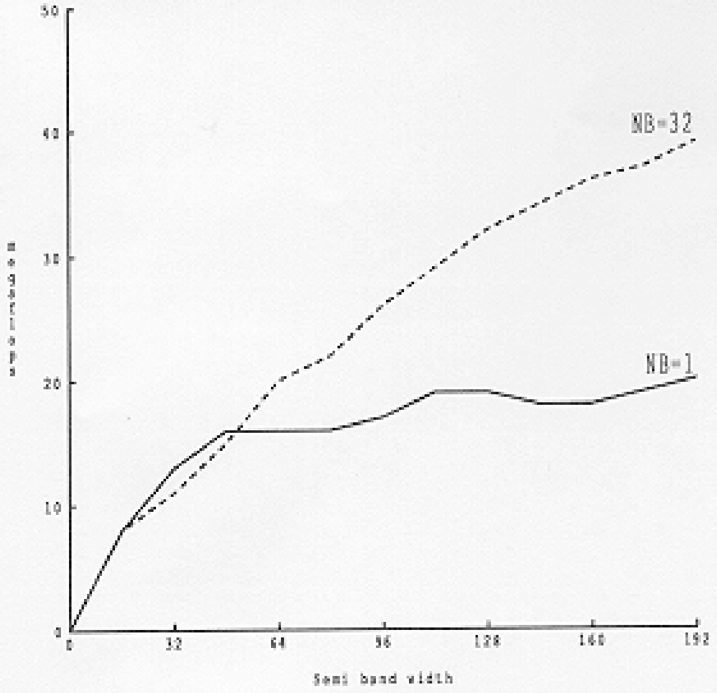

Figure 4: Speed of banded LU factorization on an IBM 3090-E/VF

Table 2 summarises the results from different machines.

	speed unblocked	speed blocked	optimum block-size	crossover value of k
IBM 3090 VF	20	39	32	64
Cray XMP	198	189	24	—
Cray 2	180	282	32	80
Siemens VP 400	281	365	16	96

Table 2: Performance of LU factorization

Again, there is clear advantage from using the blocked algorithm, on the IBM, Cray 2 and Siemens machines, for bandwidths greater than 100 or so, whereas on the Cray XMP the unblocked algorithm is always faster.

5 Concluding Remarks

Our results demonstrate the advantage of using a block algorithm for factorizing band matrices of bandwidths of 100 or more. Band matrices of order $O(1000)$ and bandwidth $O(100)$ do occur in practical applications. We aim to extend the timings to machines on which the Level 3 BLAS make use of parallel processors, in the hope that this will demonstrate an additional advantage for the block algorithms.

Acknowledgements

The authors would like to thank Ed Anderson of the University of Tennessee for the Cray 2 timings, and Klaus Geers of the University of Karlsruhe for the Siemens VP 400 timings. The second author would like to thank IBM for enabling him to visit ECSEC in Rome.

References

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

[2] Dayde, M. and Duff, I.S., Use of Level 3 BLAS in LU Factorization on the Cray 2, the ETA-10P, and the IBM 3090-200/VF. CERFACS report TR 88/1, 1988.

[3] Demmel, J., Dongarra, J.J., Du Croz, J.J., Greenbaum, A., Hammarling, S.J. and Sorensen, D., Prospectus for the Development of a Linear Algebra Library for High-Performance Computers. ANL/MCS-TM-97, Argonne National Laboratory, MCS Division, 1987.

[4] Dongarra, J., Bunch, J., Moler, J., and Stewart, G., LINPACK Users' Guide. SIAM Philadelphia, 1979

[5] Dongarra, J.J., Du Croz, J.J., Duff, I.S. and Hammarling, S.J., A Set of Level 3 Basic Linear Algebra Subprograms. ANL/MCS-TM-88 (revision 1), Argonne National Laboratory, MCS Division, 1988.

[6] Dongarra, J.J., Du Croz, J.J., Hammarling, S.J. and Hanson, R., An Extended Set of Fortran Basic Linear Algebra Subprograms. ACM Trans. Math. Software, 14, pp. 1–17, 1988.

[7] Engineering and Scientific Subroutine Library. Guide and Reference. SC23-0184-3

[8] Geers, N., Optimization of Level 2 BLAS for Siemens VP Systems. Technical report no. 37.89, University of Karlsruhe, Computer Center, 1989.

[9] Golub, G.H. and Van Loan, C.F., Matrix Computations, 2nd edition. The Johns Hopkins University Press, Baltimore, 1989.

[10] Grasemann, H., Optimization of Level 3 BLAS for Siemens VP Systems. Technical report no. 38.89, University of Karlsruhe, Computer Center, 1989.

[11] Lawson, C., Hanson, R.J., Kincaid, D. and Krogh F.T., Basic Linear Algebra Subprograms for Fortran Usage. ACM Trans. Math. Software, 5, pp. 308–323, 1979.

[12] Mayes, P.J.D. and Radiati, G., LAPACK Working Note #12: Banded Cholesky Factorization Using Level 3 BLAS. ANL/MCS-TM-134, Argonne National Laboratory, MCS Division, 1989.

[13] Mayes, P.J.D. and Radicati, G., Block Factorization Algorithms on the IBM 3090/VF. in Proceedings of the 1989 International Conference on Supercomputing, Crete, Greece, pp. 263–270, 1989.