NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2007; **14**:237–254
Published online 29 January 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla.529

The retraction algorithm for factoring banded symmetric matrices

Linda Kaufman*, †

Computer Science Department, William Paterson University, Wayne, NJ 07074, U.S.A.

SUMMARY

Let A be an $n \times n$ symmetric matrix of bandwidth $2m + 1$. The matrix need not be positive definite. In this paper we will present an algorithm for factoring A which preserves symmetry and the band structure and limits element growth in the factorization. With this factorization one may solve a linear system with A as the coefficient matrix and determine the inertia of A, the number of positive, negative, and zero eigenvalues of A. The algorithm requires between $1/2nm^2$ and $5/4nm^2$ multiplications and at most $(2m + 1)n$ locations compared to non-symmetric Gaussian elimination which requires between nm^2 and $2nm^2$ multiplications and at most $(3m + 1)n$ locations. Our algorithm reduces A to block diagonal form with 1×1 and 2×2 blocks on the diagonal. When pivoting for stability and subsequent transformations produce non-zero elements outside the original band, column/row transformations are used to retract the bandwidth. To decrease the operation count and the necessary storage, we use the fact that the correction outside the band is rank-1 and invert the process, applying the transformations that would restore the bandwidth first, followed by a modified correction. This paper contains an element growth analysis and a computational comparison with LAPACKs non-symmetric band routines and the Snap-back code of Irony and Toledo. Copyright © 2007 John Wiley & Sons, Ltd.

Received 22 May 2006; Revised 16 November 2006; Accepted 9 December 2006

KEY WORDS: symmetric; indefinite; band

1. INTRODUCTION

Let A be an $n \times n$ symmetric matrix of bandwidth $2m+1$. In this paper we will present an algorithm for factorizing A which preserves symmetry and the band structure and limits element growth in the factorization. With this factorization one may solve a linear system $Ax = b$ and determine the inertia of A, the number of positive, negative, and zero eigenvalues of A. The impetus for this research was trying to design an optical fibre using the model defined by Marcuse [1] for a radially symmetric

*Correspondence to: Linda Kaufman, Computer Science Department, William Paterson University, Wayne, NJ 07074, U.S.A.
†E-mail: kaufmanl@wpunj.edu

Contract/grant sponsor: Center for Research at William Paterson

fibre wound around a spool that leads to a 7 diagonal matrix with largest elements on the outer band. It may also be useful within the context of a shift and invert Lanczos algorithm for determining eigenvalues or in general when solving pde-eigenvalue problems defined on a regular grid.

Our factorization is based on the MDM^{T} algorithm of Bunch and Kaufman [2] which uses elementary transformations to reduce a general symmetric matrix A to a block diagonal form D where each block is either 1×1 or 2×2. (See Section 2 for further details.) For a positive definite matrix A, the matrix D is diagonal and each 2×2 block corresponds to a positive and negative pair of eigenvalues. The backward stability of the Bunch and Kaufman algorithm was formally proved by Higham [3]. Bunch and Marcia [4] indicate how one should apply the algorithm to tridiagonal matrices and in [2] it is shown how to specialize the algorithm for tridiagonal and 5 diagonal matrices. It was thought that for $m>2$ the bandwidth could grow to the full matrix especially if the larger elements were on the outermost band.

Jones and Patrick [5] claimed that for problems where there is either a small number of positive eigenvalues or a small number of negative eigenvalues the bandwidth growth for version D of the MDM^{T} algorithm [2] would be small and one could just have a work array to capture those elements in the reduced matrix which go beyond the original bandwidth. Version D does not pivot for stability if the matrix is positive or negative definite and has a higher element growth bound than version A which appears in LAPACK [6]. Their experiments indicate that this approach uses significantly less storage and is more efficient than using the non-symmetric codes in LAPACK which implement Gaussian elimination with partial pivoting. When the idea of Jones and Patrick was applied to the Marcuse model the bandwidth at most tripled, which begged the question of whether can one construct an algorithm based on the MDM^{T} algorithm which would limit the bandwidth growth.

The algorithm proposed here, the 'retraction algorithm', retracts the transformed matrix with column/row transformations each time the original MDM^{T} algorithm with row/column transformations creates elements outside the original band. The version corresponding to version D theoretically requires between $1/2nm^2$ and $5/4nm^2$ multiplications for the factorization if one uses stabilized elementary transformations throughout and never resorts to orthogonal transformations. Using orthogonal transformations for the retraction phase increases the theoretical bound on the multiplication count to $3.5nm^2$. If one is just trying to determine the inertia of matrix and hence the transformations need not be saved, then one just requires $(m+1)n$ locations to specify the matrix. If one needs to save the transformation in order to solve a system of linear equations then one needs $(2m+1)n$ locations.

Several approaches do exist for solving the banded problem while preserving the band structure. One can ignore symmetry and use Gaussian elimination with partial pivoting at the cost of between nm^2 and $2nm^2$ multiplications and $(3m+1)n$ locations. However, one cannot get the inertia from this factorization because symmetry is not preserved.

One can also use a 'chasing' algorithm that applies orthogonal transformations to reduce A to tridiagonal form (see [7–9]). These algorithm chase a bulge down the matrix to retain their bandwidth. The tridiagonal algorithm of [2, 10] can be used if one wants to retain symmetry or one can use a non-symmetric tridiagonal solver if one does not. The tridiagonal matrix can also be fed into one of the eigenvalue solvers in LAPACK. The problem with the chasing algorithms is that they require $0(n^2m)$ multiplications although they have been formulated to take advantage of parallelism.

A third alternative is the 'Snap-back' algorithm of Irony and Toledo presented at CERFACS 2004 [11]. This was the first algorithm that requires $0(nm^2)$ multiplications, preserves the band

DOI: 10.1002/nla

structure and symmetry and is backwards stable. They claim the element growth of the reduced matrices is bounded by $4^{(n-1)}$. Their algorithm is based on Aasen's technique for reduction to tridiagonal form [12] and the Parlett–Reid algorithm [13]. Whenever transformations are done that would increase the bandwidth, other transformations are used to Snap-back the reduced matrix to the original band width. (Hence, the very descriptive name of their algorithm.) On positive definite matrices, both the Snap-back algorithm and the retraction algorithm behave similarly. Irony and Toledo have graciously lent me their code and a comparison between different versions of Snap-back and retraction is given in Section 4. The implementation I received claimed that $(6m+1)n$ locations would be necessary.

In Section 2 of this paper the retraction algorithm is developed. In Section 3, an analysis of element growth is given. We note that the growth of the matrix is bounded, but like the original algorithm in Bunch and Kaufman, the growth in the factors is not. Thus, this algorithm, like the original in [2] can cause the problems sighted by Ashcraft *et al.* [14]. A comparison with the unsymmetric algorithm in LAPACK and the Snap-back algorithm is given in Section 4.

2. THE RETRACTION ALGORITHM

In this section we describe a partial pivoting strategy for transforming an $n \times n$ symmetric indefinite banded matrix A of bandwidth $2m+1$ by congruent transformations into a block diagonal matrix D where each block is of order 1 or 2. The algorithm generates a sequence of matrices $A^{(k)}$ of order k and bandwidth $2m+1$ according to the formula

$$A^{(k-s)} = Q^{\mathrm{T}}(B - YE^{-1}Y^{\mathrm{T}})Q \tag{1}$$

where Q is a set of transformations designed to ensure $A^{(k-s)}$ has a bandwidth not greater than $2m+1$ and $A^{(k)}$, or a symmetric permutation of $A^{(k)}$, is partitioned as

$$\begin{pmatrix} E & Y^{\mathrm{T}} \\ Y & B \end{pmatrix} \tag{2}$$

where E is an $s \times s$ non-singular matrix, Y is a $(k-s) \times s$ matrix, and B is a $(k-s) \times (k-s)$ matrix and s is either 1 or 2. If E is $s \times s$ we say an $s \times s$ pivot is being used.

Following Algorithm D of Bunch and Kaufman [2], the decision of whether the dimension of D is either 1 or 2 goes as follows:

(1) Let $\lambda^{(k)} = |a^k_{r,1}| =$ maximum element in absolute value in the first column.
(2) If $|a^{(k)}_{1,1}| \geqslant \alpha\lambda^{(k)}$, use a 1×1 pivot to obtain $A^{(k-1)}$, decrease k by 1 and return to step (1). Here, α is a scalar of about $\frac{1}{3}$ to balance element growth.
Else
(3) Determine $\sigma^{(k)}$, largest element in absolute value of the rth column.
(4) If $\alpha\lambda^{(k)^2} \leqslant |a^{(k)}_{1,1}|\sigma^{(k)}$, use a 1×1 pivot to obtain $A^{(k-1)}$, decrease k by 1 and return to step (1).
Else
(5) Interchange the rth and second rows and columns of $A^{(k)}$.

DOI: 10.1002/nla

(6) Perform a 2 by 2 pivot to obtain $A^{(k-2)}$.
(7) Fix it up as given below if elements stick out beyond the original band width, decrease k by 2 and return to (1).

Algorithm D of Bunch and Kaufman [2] is designed for general matrices and is missing step (7). The E matrices in (2) make up the D matrix in the MDM^{T} decomposition of A. To illustrate the algorithm, consider the matrix

$$A = \begin{pmatrix} 1 & 4 & 0 \\ 4 & 1 & 4 \\ 0 & 4 & 1 \end{pmatrix}$$

The test in step (4) dictates that a 2×2 pivot be used to reduce the matrix to $A^{(1)} = 1 - (0\ 4) \begin{pmatrix} 1 & 4 \\ 4 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 4 \end{pmatrix} = \frac{31}{15}$ so the D matrix is

$$\begin{pmatrix} 1 & 4 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & \frac{13}{15} \end{pmatrix}$$

The test in step (4) is designed to limit element growth in the reduced matrices. Each 1×1 pivot requires at most $m^2/2$ multiplications if one stores only a triangular form of the symmetric matrix. If the original matrix were positive definite, only 1×1 pivots would be chosen, and the total multiplication count would be $nm^2/2$.

The fix up algorithm is as follows: because the increase in bandwidth is largest when r in step (1) is $m + 1$, let us consider that situation. If m is 5, at the beginning of step (6) we have the following situation:

$$\begin{pmatrix} x & x & x & x & x & x & & & & & & \\ x & x & x & x & x & x & j & a & b & c & d & \\ x & x & x & x & x & x & x & x & & & & \\ x & x & x & x & x & x & x & x & x & & & \\ x & x & x & x & x & x & x & x & x & x & & \\ x & x & x & x & x & x & x & & & & & \\ & j & x & x & x & x & x & x & x & x & x & x \\ & a & x & x & x & & x & x & x & x & x & x \\ & b & & x & x & & x & x & x & x & x & x \\ & c & & & x & & x & x & x & x & x & x \\ & d & & & & & x & x & x & x & x & x \\ & & & & & & x & x & x & x & x & x \end{pmatrix} \tag{3}$$

DOI: 10.1002/nla

The elements on the wings have been purposely labelled to be referenced later. Consider the partition given in (2) and the rule given by (1) and let $Z = E^{-1}Y^{\mathrm{T}}$. Let us denote the elements of Z as

$$\begin{pmatrix} x & x & x & x & x & x & x & x & x \\ p & q & r & s & x & x & x & x & x \end{pmatrix} \tag{4}$$

The top part of $A^{(k-2)}$ matrix would then appear at the end of step (6) as

$$\begin{pmatrix} x & x & x & x & x & x & bp & cp & dp & \\ x & x & x & x & x & x & x & cq & dq & \\ x & x & x & x & x & x & x & x & dr & \\ x & x & x & x & x & as & bs & cs & ds & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & as & x & x & x & x & x & x \\ bp & x & x & bs & x & x & x & x & x & x \\ cp & cq & x & cs & x & x & x & x & x & x \\ dp & dq & dr & ds & x & x & x & x & x & x \\ & & & & x & x & x & x & x & x \end{pmatrix} \tag{5}$$

Eliminating the element dp using ds would also eliminate cp and bp since they are multiples of cs and bs. If we use a Givens transformation to do the elimination and $t = (p \times p + s \times s)^{1/2}$, then after the transformation we are left with

$$\begin{pmatrix} x & x & x & x & x & x & & & & \\ x & x & x & x & x & x & x & cq & dq & \\ x & x & x & x & x & x & x & x & dr & \\ x & x & x & x & x & at & bt & ct & dt & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & at & x & x & x & x & x & x \\ & x & x & bt & x & x & x & x & x & x \\ & cq & x & ct & x & x & x & x & x & x \\ & dq & dr & dt & x & x & x & x & x & x \\ & & & & x & x & x & x & x & x \end{pmatrix} \tag{6}$$

DOI: 10.1002/nla

Eliminating dq using dt would also eliminate cq giving us

$$\begin{pmatrix} x & x & x & x & x & x & & & & \\ x & x & x & x & x & x & x & & & \\ x & x & x & x & x & x & x & x & u & \\ x & x & x & x & x & x & x & x & m & \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ & x & x & x & x & x & x & x & x & x \\ & & x & x & x & x & x & x & x & x \\ & & u & m & x & x & x & x & x & x \\ & & & & x & x & x & x & x & x \end{pmatrix} \tag{7}$$

We can now get back to our original band structure by eliminating u with m.

The transformations that form $Q^{(k)}$are in planes $(1, r-2)$, $(2, r-2)$, $(3, r-2) \ldots (r-3, r-2)$. The matrix $Q^{(k)}$ can be written as a product of the form $Q^{(k)} = Q^{(k,1)} Q^{(k,2)} \ldots Q^{(k,r-3)}$. If $v^{(1)\mathrm{T}}$ denotes the first $r-2$ elements of the second row of Z (those denoted by $p\ q\ r\ s$ in (4)), and $v^{(i+1)\mathrm{T}} = v^{(i)\mathrm{T}} Q^{(k,i)}$, then $Q^{(k,i)}$ is designed to annihilate $v_i^{(i)\mathrm{T}}$ using $v_{r-2}^{(i)\mathrm{T}}$ and the first i elements of $v^{(i+1)\mathrm{T}}$ are zero.

Two facts should stand out from the above pictures.

(1) One could use just the elements in the first part of Z to determine the Givens or elementary transformations and never bother to actually form the bulge.
(2) The transformations are designed to undo the bulges formed by applying the second column of Y to the first $r-2$ columns of B. Thus, it would be wasteful to do a full rank 2 correction as defined by forming $W = B - YZ$.

There is another way of looking at the algorithm which also makes the solution process faster and gives us space for saving the fix-up transformations Q.

Rather than the rule in (1), we can say

$$A^{(k-2)} = Q^{\mathrm{T}}(B - YZ)Q = Q^{\mathrm{T}}BQ - Q^{\mathrm{T}}YZQ \tag{8}$$

We would then invert the process: apply Q as gleaned from Z and then do a rank-2 correction taking care to take advantage of the structure of $Q^{\mathrm{T}}Y$ and ZQ and arriving at the following reverse implementation.

Reverse formulation for steps (6) *and* (7)

(a) Determine Z and from Z determine Q.
(b) Apply the congruence transformations to B to form $F = Q^{\mathrm{T}}BQ$.
(c) Form $H = Q^{\mathrm{T}}Y$ and $G = ZQ$.
(d) Set $A^{(k-2)} = F - HG$.

DOI: 10.1002/nla

$\tilde{r}$-3	1(1)	2	4
$m-\tilde{r}+2$	2(1)	3(2)	5
$\tilde{r}$-1	4(0)	5(1)	6(1)

Figure 1. Diagram of the rank of the correction in step (d) of the Reverse algorithm. The number in parenthesis gives the rank of the correction. The numbers on the left give the number of rows where $\tilde{r} = \min(r, \max(0, k-m))$.

Determining Z in (a) requires $4(m+r-2)$ multiplications. The cost of determining F depends on the type of transformations employed. After permutation, the matrix B would have the form of (3), and as the transformations are applied to B, its $(r-2)$nd column (the short one) increases by one non-zero for each transformation. Each column transformation $Q^{(k,i)}$ requires at most m multiplications if stabilized elementary transformations are used and $4m$ multiplications if Givens transformations are used. Applying row transformation $Q^{(k,i)\mathrm{T}}$ requires i multiplications if stabilized elementary transformations are used and $4i$ if Givens are used. Thus, in the worst case, $3/2m^2$ multiplications would be required for each step (b) if stabilized elementary transformations are used and $6m^2$ multiplications if Givens transformations are used. Since there are at most $n/2$ 2×2 pivots, in the worst situation step (b) contributes $3/4m^2n$ multiplications for stabilized elementary transformations and $3m^2n$ multiplications with Givens transformations.

Because Q affects at most $m-1$ rows of Y, the zero structure of H in step (c) will mirror that of Y. In (3), the matrix Y is the first 2 columns of the figure except for the first two rows and thus H will have zeroes below row $m-1$ in its first column and may be partitioned as

$$H = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \\ 0 & H_{32} \end{pmatrix} \tag{9}$$

where H_{11} has $r-3$ rows and H_{21} has $m-r+2$ rows.

The Q matrix was designed such that the first $r-3$ elements of the second row of $G=ZQ$ would be zero. Moreover, because of the zeroes in the first column of Y, and because $Z=D^{-1}Y^{\mathrm{T}}$, G may be partitioned into

$$G = \begin{pmatrix} G_{11} & G_{12} & G_{13} \\ 0 & G_{22} & G_{23} \end{pmatrix} \tag{10}$$

where G_{11} has $r-3$ columns, G_{12} has $m-r+2$ columns and $G_{23}=a_{11}^k/a_{12}^k G_{13}$.

In Figure 1, six regions of $A^{(k-2)}$ are distinguished. The number in parenthesis for each region gives the rank of the correction in step (d) of the reverse formulation. The number of rows in each region is given at the left, and if $k\leqslant m$, so that we have a full matrix, only region 3 exists.

DOI: 10.1002/nla

Table I. Multiplication count.

All 1×1 pivots	$1/2m^2n$	
All 2×2 pivots	Stabilized elementary	Givens transformations
Steps (a) and (c)	$0(mn)$	$0(nm)$
Step (b) (worst case)	$3/4m^2n + 0(mn)$	$3m^2n + 0(mn)$
Step (d)	$1/2m^2n + 0(nm)$	$1/2m^2n + 0(mn)$
Total 2×2	$5/4m^2n + 0(mn)$	$7/2m^2n + 0(mn)$

The blocks on the diagonal are symmetric and one would be doing a symmetric correction. In the original formulation regions 1–3 would suffer a rank-2 update and regions 4–6 be affected only by a rank 1 update because of the 0's in the first column of Y in (2). In the new formulation in step (d), because of the 0's in the second row of G in Figure 1, only region 3 suffers a rank-2 correction, regions 1, 2, 5, and 6 are affected by a rank-1 correction and region 4 is not changed at all. Assuming that we only store and manipulate the lower triangular portion of $A^{(k-2)}$ means that each of the possible $n/2$ 2×2 pivots would require in step (d) at most m^2 multiplications compared to $2.5m^2$ for a naive implementation of the original formulation for the rank-2 correction.

If one is processing only the lower triangular portion of $A^{(k-2)}$, the structure of G implies there is no need to compute H_{12}.

The structure of G has important implications in the solution phase when solving $Ax = b$. If the solution phase follows the reverse formulation, one would be working with G and not Z. Thus, one can take advantage of the zeroes in G and skip multiplications. One can also use the fact that G_{23} is parallel to G_{13}.

The structure of G also provides us with some space to store the transformations that make up the Q matrices. Ordinarily, if one were saving Z and not G, one would need at most $(2m + 1)n$ locations to store D and Z plus another $(m - 2)n$ locations to store the pivot information and the multipliers in stabilized elementary transformations and the sines and cosines in the Givens transformations that form the Q matrices. However, we are given just enough space in the zero portion of G to store the multipliers in the stabilized elementary or the sines in the Givens. If we do not store G_{23} because we know it is parallel to G_{13}, we now have sufficient space to store the pivoting information or the cosines. Thus, this inverted algorithm reduces the storage to $(2m+1)n$ storage locations, a distinct advantage over using Gaussian elimination with partial pivoting.

As shown in Table I, the total cost of (a)–(d) for a 2×2 pivot is $5/2m^2$ multiplication or a maximum of $5/4m^2n$ if all the pivots are 2×2 for stabilized elementary and $7m^2n/2$ for Givens.

2.1. Implementation details

One would certainly like to use the basic linear algebra subroutines (BLAS) to implement the retraction algorithm and be able to store and manipulate only a triangular portion of the symmetric matrix. For a 1×1 pivot one would naturally use DSYR for the rank 1 correction.

For a 2×2 pivot choosing the appropriate BLAS routine which would take into consideration the zero structure of H in (9) and G in (10) depends on whether the appropriate subroutine is implemented well on the system in use. If one's implementation of the BLAS does not optimize the level 2 BLAS, as in the reference BLAS, one might choose to use only DAXPY for all the rank-1 and

DOI: 10.1002/nla

rank-2 corrections. Otherwise one might wish to use DGER for regions 2 and 5. For region 1, DSYR might be used because $G = D^{-1}H^{\mathrm{T}}$ and conversely $H = (DG)^{\mathrm{T}}$ which implies the correction $-H_{11}G_{11}$, can be written as a symmetric correction $-a_{11}^{(k)}G_{11}^{\mathrm{T}}G_{11}$. Similarly for region 6, DSYR might be used because the correction $-H_{33}G_{33} = -\theta H_{33}H_{33}^{\mathrm{T}}$ where $\theta = a_{11}^{(k)}/(a_{21}^{(k)}a_{21}^{(k)} - a_{11}^{(k)}a_{22}^{(k)})$, i.e. the (2,2) element of D^{-1}.

For region 3 or for the whole correction if one chooses to ignore the block structure of the correction, one might want to use DSYTRIDIAG_RK [15]. Unfortunately, because there is no reference implementation on netlib.org/blas at this time for this subroutine, in practice DSYTRIDIAG_RK is not suitable. One could also consider DSYR2K for region 3, but for the ATLAS BLAS [16] on our SUN computer using DSYR2K was about 4 times slower than using DAXPY's for each column on a triangular region. Using DGEMV was slightly faster than using 2 DAXPY's for the rank-2 correction for each column of the triangular regions.

Clouding the choice of BLAS for the correction is the fact that some of the vector lengths may be very small and one should keep in mind that whenever retractions is needed, the cost of the rank-2 correction is not that important.

3. ELEMENT GROWTH AND DETERMINATION OF α

In this section we analyse the element growth of the matrices in the retraction algorithm when stabilized elementary transformations are used in the fix-up step. We will need to look at three situations, a 1×1 step, a 2×2 step when retraction is not required and a 2×2 step when retraction is necessary. We will use this element growth to determine an appropriate value of α.

Let $\mu = \max_{1 \leqslant i,j \leqslant n} |a_{ij}|$ and $\mu^{(k)} = \max_{1 \leqslant i,j \leqslant k} |a_{ij}^k|$ for each reduced matrix $A^{(k)}$ that is computed. Because $\lambda^{(k)} = \max_{1 \leqslant j \leqslant \min(m+1, n-k+1)} |a_{j,1}^k|$, we know $\lambda^{(k)} \leqslant \mu^{(k)}$. Similarly, since $\sigma^{(k)} =$ maximum absolute value of the elements in the rth column, $\sigma^{(k)} \leqslant \mu^{(k)}$.

If a 1×1 pivot is chosen in step (2), we have for $1 \leqslant i, j \leqslant m$

$$a_{ij}^{(k-1)} = a_{i+1,j+1}^{(k)} - a_{i+1,1}^{(k)} a_{j+1,1}^{(k)} / a_{11}^{(k)} \tag{11}$$

which implies

$$\mu^{(k-1)} \leqslant \mu^{(k)} + \lambda^{(k)} \lambda^{(k)} / a_{11}^{(k)} \tag{12}$$

Because in step (2), $1/\alpha \geqslant \lambda^{(k)}/|a_{1,1}^{(k)}|$

$$\mu^{(k-1)} \leqslant \mu^{(k)} + \lambda^{(k)}/\alpha \leqslant \mu^{(k)}(1 + 1/\alpha)$$

If a 1×1 pivot is chosen in step (4),

$$\alpha \lambda^{(k)2} \leqslant |a_{1,1}^{(k)}| \sigma^{(k)}$$

or $\lambda^{(k)2}/|a_{1,1}^{(k)}| \leqslant \sigma^{(k)}/\alpha$ which means that (11) suggests

$$\mu^{(k-1)} \leqslant \mu^{(k)} + \lambda^{(k)} \lambda^{(k)} / a_{11}^{(k)} \leqslant \mu^{(k)} + \sigma^{(k)}/\alpha \leqslant \mu^{(k)}(1 + 1/\alpha) \tag{13}$$

DOI: 10.1002/nla

If a 2×2 is used and no elements from the elimination go beyond the bandwidth, then after permuting $A^{(k)}$, we have for $1\leqslant i,\ j\leqslant m-1$

$$a_{ij}^{(k-2)}=a_{i+2,j+2}^{(k)}-((a_{i+2,1}^{(k)}a_{22}^{(k)}-a_{i+2,2}^{(k)}a_{21}^{(k)})a_{j+2,1}^{(k)}$$
$$+(a_{i+2,2}^{(k)}a_{11}^{(k)}-a_{i+2,1}^{(k)}a_{21}^{(k)})a_{j+2,2}^{(k)})/((a_{11}^{(k)}a_{22}^{(k)}/a_{21}^{(k)}-a_{21}^{(k)})a_{21}^{(k)}) \tag{14}$$

Since a 2×2 is used when $\alpha|a_{21}^{(k)}|^2>|a_{11}^{(k)}|\sigma^{(k)}\geqslant|a_{11}^{(k)}||a_{22}^{(k)}|$, we know

$$|a_{11}^{(k)}||a_{22}^{(k)}|/|a_{21}^{(k)}|\leqslant|a_{21}^{(k)}|\alpha$$

which implies

$$|a_{21}^{(k)}-a_{11}^{(k)}a_{22}^{(k)}/a_{21}^{(k)}|>|a_{21}^{(k)}|-|a_{21}^{(k)}|\alpha=|a_{21}^{(k)}|(1-\alpha)$$

or

$$1/|a_{21}^{(k)}-a_{11}^{(k)}a_{22}^{(k)}/a_{21}^{(k)}|<1/(\lambda^{(k)}(1-\alpha)) \tag{15}$$

Because $|a_{22}^{(k)}|\leqslant\sigma^{(k)}$, (14) and (15) suggest

$$\mu^{(k-2)}\leqslant\mu^{(k)}+((\lambda^{(k)}\sigma^{(k)}+\sigma^{(k)}\lambda^{(k)})\lambda^{(k)}+(\sigma^{(k)}|a_{11}^{(k)}|+|a_{21}^{(k)}|^2)\sigma^{(k)})/({\lambda^{(k)}}^2(1-\alpha)) \tag{16}$$

which with the condition $|a_{11}^{(k)}|\sigma^{(k)}<\alpha|a_{21}^{(k)}|^2$, implies

$$\mu^{(k-2)}\leqslant\mu^{(k)}+(\sigma^{(k)}+\sigma^{(k)}+(\alpha+1)\sigma^{(k)})/(1-\alpha)$$

or

$$\mu^{(k-2)}\leqslant\mu^{(k)}(1+(3+\alpha)/(1-\alpha)) \tag{17}$$

For completeness we should mention the case when $m\leqslant i\leqslant m+\max(1,r-1)$, $j\leqslant i$ and no retraction is needed. For these indices the elements of the first column of Y are zero so that the formula for $A^{(k-2)}$ reduces to

$$a_{ij}^{(k-2)}=a_{i+2,j+2}^{(k)}-(-a_{i+2,2}^{(k)}a_{21}^{(k)}a_{j+2,1}^{(k)}+a_{i+2,2}^{(k)}a_{11}^{(k)}a_{j+2,2}^{(k)})/((a_{11}^{(k)}a_{22}^{(k)}/a_{21}^{(k)}-a_{21}^{(k)})a_{21}^{(k)})$$

and the bound on element growth, using (15), is essentially

$$\mu^{(k-2)}\leqslant\mu^{(k)}+(\sigma^{(k)}\lambda^{(k)}\lambda^{(k)}+\sigma^{(k)}|a_{11}^{(k)}|\sigma^{(k)})/({\lambda^{(k)}}^2(1-\alpha)) \tag{18}$$

which simplifies to

$$\mu^{(k-2)}\leqslant\mu^{(k)}(1+(1+\alpha)/(1-\alpha)) \tag{19}$$

The bound given in (19) is obviously smaller than that given in (17).

From the bound in (19) and the bound in (12), we can determine a value of α assuming 2 steps of a 1×1 pivot is equivalent to a 2×2 pivot. We would thus need to determine α such that

$$(1+1/\alpha)^2=(1+(3+\alpha)/(1-\alpha))$$

DOI: 10.1002/nla

or $\alpha = 0.6404$. With this value of α, the element growth for n steps is bounded by $(1 + 1/\alpha)^{n-1}$ or 2.57^{n-1}.

When retraction is needed, so that Q in (1) is not the identity matrix, it is easy to determine element growth using the implementation in (8). Recall in that implementation

$$A^{(k-2)} = Q^{\mathrm{T}} B Q + HG$$

where $H = Q^{\mathrm{T}} Y$ can be partitioned as in (9) and $G = ZQ$ can be partitioned as in (10).

Referring to Figure 1 for region 3, which is not affected by Q but is affected by both columns of H, the formula for $A^{(k-2)}$ is given by (14) and hence the bound on element growth is given by (17).

For regions 5 and 6, the formula for $A^{(k-2)}$ is given by (18) and hence the bound on element growth is given by (17).

To analyse the element growth in regions 1 and 2 we need first to determine a bound on the elements of the first row of Z and then to examine what happens to the first row of Z, the first column of Y and B, when Q is applied to them. The elements of the first row of Z are given by the formula

$$z_{i,1} = (a^{(k)}_{i+2,1} a^{(k)}_{22} - a^{(k)}_{2,i+1} a^{(k)}_{21})/((a^{(k)}_{11} a^{(k)}_{22}/a^{(k)}_{21} - a^{(k)}_{21}) a^{(k)}_{21}) \tag{20}$$

Let $\beta = \max_{1 \leqslant i \leqslant r-2} |z_{i,1}|$. Then based on (15) and (16) we have

$$\beta \leqslant (\alpha \lambda^{(k)} \sigma^{(k)} + \sigma^{(k)} \lambda^{(k)})/({\lambda^{(k)}}^2 (1 - \alpha)) \leqslant (1 + \alpha)/(1 - \alpha) \tag{21}$$

To determine the affect of Q on B, Y, and Z let us first witness the affect of $Q^{(k)}$ when applied as a column transformation on a row vector v^{T} of length $r - 2$. The matrix $Q^{(k)}$ can be written as a product of the form $Q^{(k)} = Q^{(k,1)} Q^{(k,2)} \ldots Q^{(k,r-3)}$ where each $Q^{(k,i)}$ is a stabilized elementary transformation in planes i and $r - 2$ having the form $Q^{(k,i)} = P^{(k,i)}(I + x_{k,i} e_i e^{\mathrm{T}}_{r-2})$ where $P^{(k,i)}$ is a permutation matrix in planes i and $r - 2$, $|x_{k,i}| \leqslant 1$, and e_i is the ith column of the identity matrix.

Lemma 1
Let $v^{(1)\mathrm{T}} = v^{\mathrm{T}}$, and $v^{(i+1)\mathrm{T}} = v^{(i)\mathrm{T}} Q^{(k,i)}$ and assume $\beta = \max_{1 \leqslant j \leqslant r-2} |v_j|$. Then $|v^{(i)}_j| \leqslant 2\beta$ for $1 \leqslant j \leqslant i - 1$ and $|v^{(i)}_j| \leqslant \beta$ for $i \leqslant j \leqslant r - 2$.

Proof
The proof of the Lemma follows by induction on i. It is obviously true for $i = 1$. Let us assume for $|v^{(i)}_j| \leqslant 2\beta$ for $1 \leqslant j \leqslant i - 1$ and $|v^{(i)}_j| \leqslant \beta$ for $i \leqslant j \leqslant r - 2$ and consider $\tilde{v}^{(i)\mathrm{T}} = {v_i}^{\mathrm{T}} P^{(k,i)}$. Whether $P^{(k,i)}$ is the identity matrix or interchanges the ith or $(r - 2)$th elements of $\tilde{v}^{(i)\mathrm{T}}$, the elements of $\tilde{v}^{(i)\mathrm{T}}$ have the same bounds as $v^{(i)\mathrm{T}}$. Except for the ith element, all elements of $v^{(i+1)\mathrm{T}}$ are identical to $\tilde{v}^{(i)\mathrm{T}}$ and $v^{(i+1)\mathrm{T}}_i = \tilde{v}^{(i)\mathrm{T}}_i + x_{k,i} \tilde{v}^{(i)\mathrm{T}}_{r-2}$. Thus, by induction $|v^{(i+1)\mathrm{T}}_i| \leqslant \beta + |x_{k,i}|\beta \leqslant 2\beta$. This means that $|v^{(i+1)}_j| \leqslant 2\beta$ for $1 \leqslant j \leqslant i$ and $|v^{(i+1)}_j| \leqslant \beta$ for $i + 1 \leqslant j \leqslant r - 2$. □

From Lemma 1 and (21), it follows that in region 2 where the first columns of Z and the first $r - 2$ columns of B are multiplied on the right by Q that

$$\mu^{(k-2)} \leqslant 2\mu^{(k)} + \mu^{(k)} 4/(1 - \alpha)$$

DOI: 10.1002/nla

In region 1, which is affected by both row transformations of Q^{T} and column transformations of Q but is not affected by the second column of H, we get

$$\mu^{(k-2)} \leqslant 4\mu^{(k)} + \mu^{(k)} 8/(1-\alpha) \tag{22}$$

Obviously the element growth is largest in region 1, and from the bound in (22) and the bound in (12) we can suggest a value of α assuming 2 steps of a 1×1 pivot is equivalent to a 2×2 pivot. We would thus need to determine α such that

$$(1 + 1/\alpha)^2 = 4 + 8/(1-\alpha)$$

or $\alpha = \frac{1}{3}$. With this value of α, the element growth for $n-1$ steps is bounded by $(1 + 1/\alpha)^{n-1}$ or 4^{n-1}.

4. COMPUTATIONAL EXPERIENCE

The inverse formulation of the retraction algorithm was implemented in FORTRAN and in C and compared on the Sun Enterprise 450r system with four Ultra SPARC II processors using the ATLAS BLAS [16] with the Snap-back Algorithm of Irony and Toledo [11] in C and the unsymmetric banded codes DGBTRF and DGBTF2 from LAPACK [6]. In FORTRAN for the retraction scheme column elements of the matrix are stored consecutively, while in C the row elements are stored consecutively.

In the first set of examples outlined in Table II, we tried to exercise various parts of our code. The first problem was positive definite and only 1×1 pivots were used. In the other examples the largest elements appear on the outermost diagonal. The rationale for this structure was that it would produce many 2×2 pivots and show the largest execution times both in our code and the unsymmetric LAPACK codes. In the original motivating example in optical fibre design, the elements on the outermost diagonal were several orders of magnitude larger than the elements on the other off-diagonals. This type of structure might also be exhibited in a two-dimensional problem from a partial differential equation on a regular grid where the coupling in one dimension is stronger than the coupling in the other dimension. Although a positive/negative eigenvalue pair is necessary for a 2×2 pivot, as shown by our second example in Table IV, it is not sufficient, and one can have an equal number of positive and negative eigenvalues and not have all 2×2 pivots.

The timing comparison in Table III suggests that one can expect to take less time using the retraction algorithm than using LAPACK's DGBTF2 [6], a general banded algorithm which ignores

Table II. Characteristics of examples with $n = 1000$ and $m = 100$ which yield a range of computation times for different algorithms.

Example	1	2	3	4
Diagonal	100	10	10	1
Offdiagonal except for outer	1	1	1	$10 \times$ number of subdiagonal
Outer diagonal	1	100	10000	$10 \times m$
Condition number	7.4	97.5	5.4	2.1×10^4
Number of + eigenvalues	1000	502	500	498

DOI: 10.1002/nla

Table III. Timings (s) with ATLAS BLAS for $n = 1000, m = 100$.

Example	1	2	3	4
Retraction with stabilized elementary	0.106	0.168	0.330	0.241
Retraction with orthogonal	0.106	0.240	0.520	0.220
Snap-back with stabilized elementary	0.140	0.550	1.170	1.300
Snap-back with orthogonal	0.140	0.590	2.200	1.230
DGBTF2 from LAPACK	0.202	0.392	0.392	0.392
DGBTRF from LAPACK	0.143	0.232	0.232	0.232

Table IV. Behaviour of the retraction algorithm.

Problem	1	2	3	4
With elementary numbers of 2×2's	0	161	500	397
With orthogonal numbers of 2×2's	0	221	486	172
With elementary average r in retraction	0	66.6	89.7	57.8
With orthogonal average r in retraction	0	57.9	90.6	30.7
With elementary element growth	1	11.4	1.001	57.3
With orthogonal element growth	1	9.2	1.001	121.5

symmetry, but in some cases it may take more time than LAPACK's DGBTRF [6], which uses block transformations. The comparison on the positive definite problem with DGBTF2 supports our theory of a decrease in time by a factor of 2. The comparison with Snap-back [11] suggests that all their comparisons indicating their superiority to chasing schemes, as in [9], also hold for the retraction algorithm.

For the positive definite example, the first example, Snap-back and the retraction scheme took approximately the same time with the reference BLAS. The difference in the times in Table III points to a data locality difficulty with the implementation of the Snap-back algorithm. The implementation of Snap-back in C stored the upper triangular part of the matrix by columns and the code saved it in an array of $(6m + 1)n$ space. In comparison the two versions of our code involved an array of $(2m + 1)n$ locations. Thus, cache misses might have a greater effect on the performance of the Snap-back implementation than on the performance of the retraction implementation. Thus, the favourable times across the four problems for the retraction algorithm *versus* Snap-back might be rather different if the data layout scheme for Snap-back was changed.

Some of the variation of the times given by Table III for the retraction algorithm is explained by Table IV. For example 3, almost every pivot was a 2×2 pivot that entailed interchanging the outermost diagonal with the first subdiagonal. The multiplication count for the elementary version is $1/2m^2n + p(mr + r^2/2)$ where p is the number of 2×2 pivots and hence bounded by $n/2$. For example 3, we have $r \approx m$ and $p \approx n/2$ giving $5/4m^2n$ counts, while for example 2, $r \approx 2m/3$

DOI: 10.1002/nla

Table V. Accuracy comparison with $n = 1000$ and $m = 100$.

Example	1	2	3	4
Retraction with elementary	3.6×10^{-15}	1.6×10^{-13}	3.2×10^{-13}	2.3×10^{-11}
Retraction with orthogonal	3.6×10^{-15}	1.3×10^{-13}	9.5×10^{-14}	5.5×10^{-12}
Snap-back with elementary	5.1×10^{-15}	4.3×10^{-11}	2.2×10^{-4}	3.1×10^{-5}
Snap-back with orthogonal	5.1×10^{-15}	1.3×10^{-13}	4.4×10^{-12}	3.3×10^{-12}
DGBTF2 and DGBTRF	6.9×10^{-15}	1.0×10^{-14}	5.7×10^{-15}	1.1×10^{-12}

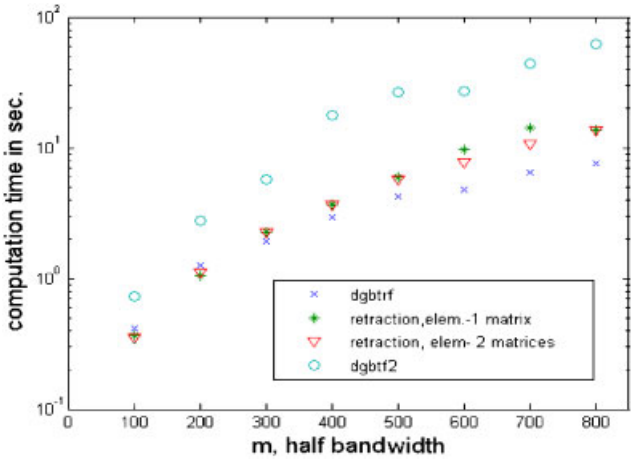

Figure 2. Time *versus* bandwidth on random matrices on [0,1] with $n = 2000$.

and $p \approx n/6$ suggesting one should expect that example 2 should require a little bit more than $\frac{1}{2}$ the time of example 3. Example 4 is somewhat of an anomaly. Because the number of 2×2 pivots for the orthogonal version is less than half the number of 2×2 pivots for the stabilized elementary version and because the average r is significantly less, the computation time for the orthogonal version is actually less than that of the elementary version. What is equally surprising is the large element growth for the orthogonal version on that example. This large element growth is also evident for some random examples given in Figure 5.

For Table V we constructed the right-hand side so that the solution is a vector of all 1's. The accuracy gives the largest deviation from 1.0 in the solution. We have certainly discovered examples where using orthogonal transformations with Snap-back by Irony and Toledo [11] is absolutely necessary. The results for Snap-back motivated us to implement the retraction algorithm with orthogonal transformations. Orthogonal transformations improved the situation a bit, but we were hoping for 4 bits. A more sophisticated pivoting scheme might produce better results. Unlike DSYTRF in LAPACK [6], the retraction algorithm as given in this paper does not permute rows and columns for stability when using a 1×1 pivot. Using permutations as in DSYTRF does not destroy the band structure, but for stabilized elementary transformations, it raises the worst case multiplication count from $5/4m^2n$ to $2m^2n$ and increases the space bound from $(2m + 1)n$ to

DOI: 10.1002/nla

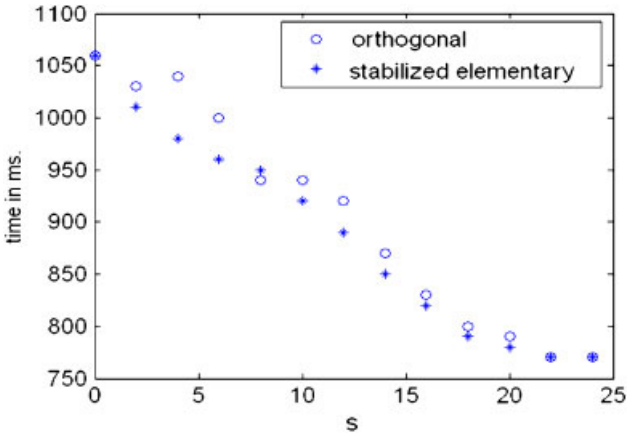

Figure 3. Computational time as a function of s on $A + sI$, A random on $[-1, 1]$, $m = 200$, $n = 2000$.

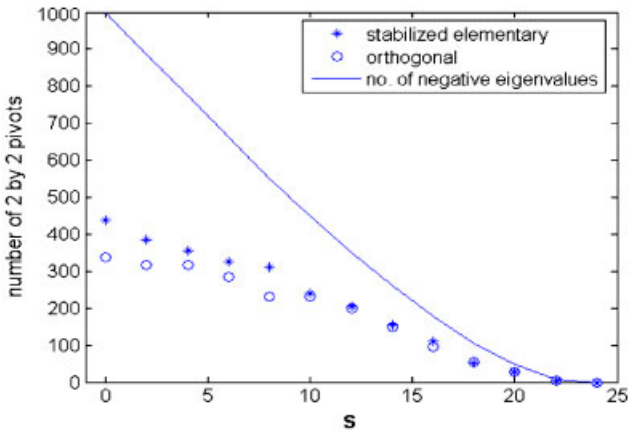

Figure 4. Number of 2×2 pivots as a function of s on $A + sI$, A random on $[-1, 1]$, $m = 200$, $n = 2000$.

$(3m + 1)n$. A rook pivoting scheme as in [14, 17] can be implemented for tridiagonal matrices, but a full rook scheme for general banded matrices would probably lead to a 'chasing' scheme as in [7, 9]. One could, however, allow a burn-at-both-ends scheme, as in [18], which would start at the bottom whenever that strategy would lead to a smaller multiplier.

Figure 2 portrays our computational experience on random examples on [0,1] with $n = 2000$ and variable bandwidth using ATLAS BLAS and LAPACK's DGBTF2 and DGBTRF for unsymmetric banded matrices and two implementations of the retraction algorithm with elementary transformations. The first implementation of the retraction algorithm used one array with $2m + 1$ rows and second used two arrays, one with $m + 1$ rows on which the computations were performed and another to hold the transformations as suggested in [5]. For DGBTRF a block size of 48 was used, which seemed to be the best of the values we considered. There was about 5% variation for blocksizes between 32 and 96. Blocksizes above 96 consistently produced higher computation times. The number of positive eigenvalues was usually close to 1000 and there were between 300

DOI: 10.1002/nla

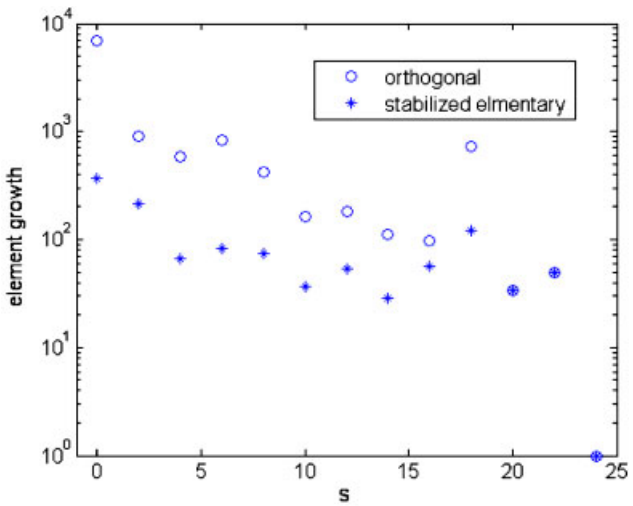

Figure 5. Element growth as a function of s on $A + sI$, A random on $[-1, 1]$, $m = 200$, $n = 2000$.

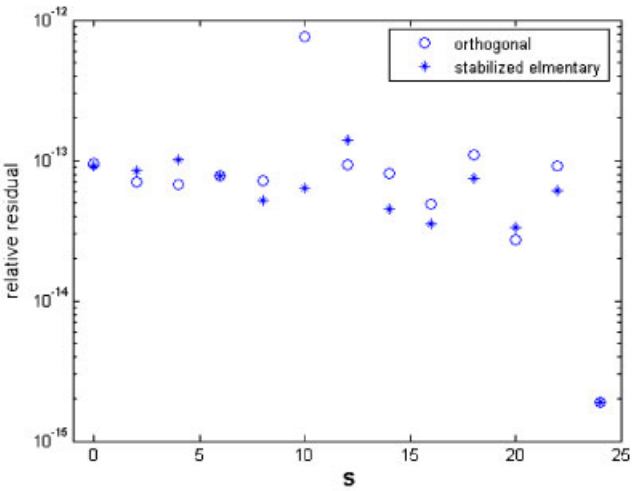

Figure 6. Relative residual as a function of s on $A + sI$, A random on $[-1, 1]$, $m = 200$, $n = 2000$.

and 400 2×2 pivots. For bandwidths below 300, both versions of the retraction code are faster than the blocked unsymmetric code, but as the bandwidth increases, the blocked unsymmetric code is superior. The times for the retraction algorithm still more closely resemble the times for the blocked unsymmetric code rather than the unblocked one. The differences between the two implementations of the retraction algorithm suggest that for higher bandwidths, data locality is important. Clearly, cache misses are expensive in this environment.

Originally, the author assumed that using orthogonal transformations would be much more expensive but more accurate. These assumptions might hold for some examples, but for the shifted random examples on $[-1, 1]$ (s denoted the shift) used in Figures 3–6, that was not the case.

DOI: 10.1002/nla

The ratio of the computation time for orthogonal transformations *versus* that for stabilized transformations was decreased in part because there were fewer 2×2 pivots as shown in Figure 4.

Figure 5 indicates that for these random examples, as with example 4 in Table IV, the element growth was more for the orthogonal version than for the stabilized elementary version. The relative residuals given in Figure 6 gave no discernible trend. Sometimes the relative residual was larger for the orthogonal version. Indeed our experimentation raised some interesting questions.

5. CONCLUSIONS

Our experimentation has shown that the computation time for the inverse formulation of the retraction algorithm is close to the time for DGBTRF, the blocked version for unsymmetric Gaussian elimination in LAPACK and is in general about half that for the unblocked version DGBTF2 of LAPACK. As the bandwidth increases, the time differences between the retraction and DBGTF2 increase. We have given examples where our implementation with stabilized elementary transformations in the retraction phase is as accurate as an implementation of Snap-back with orthogonal transformations and takes about $\frac{1}{6}$ the time. On random problems we have compared the retraction algorithm using stabilized elementary transformations in the retraction phase and using orthogonal transformations in the retraction phase and have found that using stabilized elementary transformation is slightly, but not significantly, faster. In addition, using orthogonal transformations does not always lower the relative residuals and often increases the element growth.

REFERENCES

1. Marcuse D. Field deformation and loss caused by curvature of optical fibers. *Journal of the Optical Society of America* 1972; **66**(4):311–320.
2. Bunch JR, Kaufman L. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of Computation* 1977; **31**(137):163–179.
3. Higham NJ. Stability of the diagonal pivoting method with partial pivoting. *SIAM Journal on Matrix Analysis and Applications* 1997; **18**(1):52–65.
4. Bunch JR, Marcia RF. A simplified pivoting strategy for symmetric tridiagonal matrices. *Numerical Linear Algebra with Applications* 2004; **12**(9):135–153.
5. Jones MT, Patrick ML. Bunch Kaufman algorithm for real symmetric indefinite banded matrices. *SIAM Journal on Matrix Analysis and Applications* 1993; **14**(2):553–559.
6. Anderson E, Bai Z, Bischof C, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Ostrouchov S, Sorensen D. *LAPACK Users' Guide* (2nd edn). Society for Industrial and Applied Mathematics: Philadelphia, PA, 1995.
7. Kaufman L. Band reduction algorithms revisited. *ACM Transactions on Mathematical Software* 2000; **26**(4): 551–567.
8. Lang B. A parallel algorithm for reducing symmetric banded matrices to tridiagonal form. *SIAM Journal on Scientific Computing* 1993; **14**(6):1320–1338.
9. Bischof CH, Lang B, Sun X. A framework for symmetric band reduction. *ACM Transactions on Mathematical Software* 2000; **26**(4):581–601.
10. Bunch JR. Partial pivoting strategies for symmetric matrices. *SIAM Journal on Numerical Analysis* 1974; **11**:521–528.
11. Irony D, Toledo S. The snap-back pivoting method for symmetric banded indefinite matrices. *SIAM Journal on Matrix Analysis and Applications* 2006; **14**(2):398–424.
12. Aasen J. On the reduction of a symmetric matrix to tridiagonal form. *British Information Technology* 1971; **11**:233–242.
13. Parlett BN, Reid JK. On the solution of a system of linear equations whose matrix is symmetric but not definite. *British Information Technology* 1970; **10**:386–397.

DOI: 10.1002/nla

14. Ashcraft C, Grimes RG, Lewis JG. Accurate symmetric indefinite linear equation solvers. *SIAM Journal on Matrix Analysis and Applications* 1998; **20**(2):515–561.
15. Blackford LS, Demmel J, Dongarra J, Duff I, Hammarling S, Henry G, Heroux M, Kaufman L, Lumsdaine A, Petitet A, Pozo R, Remington K, Whaley RC. An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software* 2002; **28**(2):135–151.
16. Whaley RC, Petitet A, Dongarra JJ. Automated emperical optimizations of software and the ATLAS project. *Parallel Computing* 2001; **27**(11):3–35.
17. Chang X-W. Some features of Gaussian elimination with rook pivoting. *BIT Numerical Mathematics* 2002; **42**:66–83.
18. Babuska I. Numerical stability in problems of linear algebra. *SIAM Journal on Numerical Analysis* 1972; **26**(4):53–72.

DOI: 10.1002/nla