Decomposition of a Symmetric Matrix

James R. Bunch
Linda Kaufman
Beresford N. Parlett

CU-CS-080-75

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Decomposition of a Symmetric Matrix

James R. Bunch
Department of Mathematics
University of California, San Diego

Linda Kaufman
Computer Science Department
University of Colorado, Boulder

Beresford N. Parlett
Department of Mathematics and
Department of Electrical Engineering
and Computer Science
University of California, Berkeley

#CU-CS-080-75 October 1975

ABSTRACT

An algorithm is presented to compute a triangular factorization and the inertia of a symmetric matrix. The algorithm is stable even when the matrix is not positive definite and is as fast as Choleski. Programs for solving associated systems of linear equations are included.

1. Theoretical Background

A real symmetric matrix A usually possesses a unique triangular factorization

$$A = MDM^t , \tag{1}$$

where M is unit lower triangular, M^t denotes the transpose of M, and D is diagonal. However this factorization does not always exist and, what is worse, it may exist and be hopelessly unstable in the sense that the intermediate quantities $|m_{ij}^2 d_j|$, $i \geq j$, may be arbitrarily greater than the elements of A. The use of interchanges to produce

$$PAP^t = MDM^t \tag{2}$$

where P is a permutation matrix, removes some but not all of the troublesome cases. The simplest recalcitrant example is

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

These difficulties have nothing to do with the condition number of A for inversion, namely $||A|| \cdot ||A^{-1}||$.

It is interesting that an adequate degree of stability can be maintained by accepting the small increase in complexity that comes from allowing D above to be <u>block</u> diagonal with blocks of order 1 or 2. A procedure <u>parsymdec</u> for obtaining the factorization of the form (2) with block diagonal D is presented in this contribution. The algorithm is easily modified to accept complex Hermitian or complex symmetric matrices.

The method described here is the latest in a sequence which can be traced in the articles Parlett and Reid [8], Bunch and Parlett [5], Bunch [3], Aasen [1], Bunch and Kaufman [4]. All these methods are special variants of the familiar triangular decomposition (1) which are designed both to preserve symmetry and to prevent harmful element growth. Crucial points in the implementation of these algorithms are (a) the bounds on the $|m_{ij}{}^2 d_j|$, $i \geq j$, (b) the computing time required for the factorization, (c) the form of the permutation P, and (d) the storage requirements for P, M, and D. There is an additional constraint that the algorithms should be no slower than those in [2] which solve Ax=b without maintaining symmetry.

Block factorization techniques for symmetric A begin with A $(=A^{(n)})$, and produce a sequence of reduced matrices $\{A^{(i)}\}$ as described below. Note that the order of $A^{(i)}$ is i. At the typical step a permutation of $A^{(i)}$ is made and written in partitioned form as

$$P^{(i)t}A^{(i)}P^{(i)} = \begin{bmatrix} X & C^t \\ C & Y \end{bmatrix} ,$$

where the pivot X is s×s and nonsingular with s=1 or 2. The next reduced matrix is defined by

$$A^{(i-s)} = Y - CX^{-1}C^t \quad . \qquad (3)$$

The integer s is chosen to be 2 whenever it is predicted that this will cause less growth in the elements of $A^{(i-s)}$ than would the choice s=1.

The goal is to choose $P^{(i)}$ as simply as possible consistent with a modest bound on element growth at each step. Our strategy depends on a parameter of $0<\alpha<1$, an appropriate value for which is given below. In order to simplify the description of the algorithm let I_{mj} denote the permutation matrix obtained by interchanging rows m and j of the identity matrix I.

The reduction of $A^{(i)}$ to $A^{(i-s)}$ is as follows:

(i) Determine $\lambda = |a_{j1}^{(i)}| = \max_{2\le k\le i} |a_{k1}^{(i)}|$

(ii) If $|a_{11}^{(i)}| \ge \alpha\lambda$ then take $P^{(i)} = I$, s=1, go to (vii)

(iii) Determine $\sigma = \max_{1\le k\le i} |a_{kj}^{(i)}|$, $k\ne j$

(iv) If $|a_{11}^{(i)}|\sigma \ge \alpha\lambda^2$ take $P^{(i)} = I$, s=1, and go to (vii)

(v) If $|a_{jj}^{(i)}| \ge \alpha\sigma$ then take $P^{(i)} = I_{1j}$, s=1, and go to (vii)

(vi) Take $P^{(i)} = I_{2j}$, s=2, and go to (vii)

(vii) Compute $A^{(i-s)}$ according to (3) and decrement i by s.

The following remarks explain the structure of the algorithm. Let $\mu^{(i)} = \max |a_{jk}^{(i)}|$ $j,k=1,\ldots,i$. Note that $\lambda\le\sigma\le\mu^{(i)}$.

Whenever s=1,

$$a_{j-1,k-1}^{(i-1)} = a_{jk}^{(i)} - a_{j1}^{(i)}a_{k1}^{(i)}/a_{11}^{(i)}$$

whence

$$\mu^{(i-1)} \le \mu^{(i)} + \begin{cases} \lambda/\alpha & \text{if (ii) holds} \\ \sigma/\alpha & \text{if (iv) or (v) holds} \end{cases}$$

$$\le \mu^{(i)}(1 + 1/\alpha) \quad .$$

Whenever s=2, a more complicated analysis (see Bunch and Kaufman [4]) shows that

$$\mu^{(i-2)} \le \mu^{(i)}(1 + 2/(1-\alpha))$$

The bound on the growth in passing from $A^{(i)}$ to $A^{(i-2)}$ is minimized when

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

i.e. where $\alpha = \alpha_0 \equiv (1 + \sqrt{17})/8 \doteq 0.6404$.

With this choice

$$\max\{\mu^{(1)}, \mu^{(2)}\} \le (2.57)^{n-1} \mu^{(n)} \quad .$$

See Bunch and Parlett [5] and Bunch [3] for more details.

2. Applicability

The procedure parsymdec has two uses. Combined with the procedure parsymsol it can produce the solution x to the linear equation $Ax = b$ with symmetric A and any number of right hand sides b. Secondly, the decomposition permits the inertia (or, equivalently, the signature) of A to be obtained from D with no further computation.

The inertia of A is the triple (π,ν,ξ) where π,ν,ξ are, respectively, the number of positive, negative, and zero eigenvalues of A. The rank of A is $\pi+\nu$ and its signature is $\pi-\nu$. The inertia is a complete set of invariants of A under congruence transformations, $A \to S^tAS$. It characterizes the quadratic form associated with A.

The procedure inertia can certainly be used in conjunction with a bisection algorithm to determine the eigenvalues of a symmetric matrix. Moreover the determinant is readily evaluated for use with inverse linear interpolation. For examples of other uses of the inertia see Cottle [6].

Consult Bunch and Parlett [5] for natural examples of systems of equations with coefficient matrices which are symmetric but not definite.

The Positive Definite Case ($\pi=n$, $\nu=\xi=0$). If A is known to be positive definite the procedures symdet and symsol or the procedures choldet 1 and cholsol 1 in [7] should be used. If A is positive definite but not known to be so, then using parsymdec might increase the execution time by about 10% on some machines, but it is actually cheaper with others. What is more serious is that parsymdec may utilize some interchanges which may destroy any bandstructure that A enjoys. These interchanges are used to curb the growth of D, but when A is positive definite, the bounds on $||M||$ and $||D||$ are irrelevant because no element growth can occur.

The Indefinite Case ($\pi>0$, $\nu>0$). The algorithm was designed for this case. Nevertheless the procedures unsymdet and unsymsol in [2] can solve $Ax = b$ stably in approximately $n^3/3$ multiplications and n^2 storage by sacrificing symmetry. Rival techniques must not spend so much time choosing and executing interchanges that they exceed this operation count. Parsymdec requires $n^3/6$ multiplications.

Singular Case ($\xi>0$). For the inertia problem it is important that the parsymdec can accept singular matrices.

3. Formal Parameter List

3.1 Input to procedure parsymdec

a	the elements of an $n \times n$ symmetric matrix A. Only the upper triangular portion need be stored.
n	order of the matrix A

Output of procedure parsymdec

a	elements of the MDM^t decomposition of A are stored in its upper triangular portion including the diagonal. Its strictly lower triangular portion is left untouched.
change	a vector recording the interchanges performed on A during the computation of the decomposition and block structure of D.

3.2 Input to procedure parsymsol

a	elements of the MDM^t decomposition of a symmetric matrix A as produced by parsymdec.
change	a record produced by parsymdec of the interchanges performed on the A matrix and of the block structure of D.

b a vector of length n containing the right hand side of the equation $Ax = b$.

n order of the matrix A.

Output from procedure parsymsol

b the solution to the problem $Ax = b$.

fail This is the exit used when A, possibly as a result of rounding errors, is singular.

3.3 Input to procedure inertia

a elements of the MDM^t decomposition of a symmetric matrix A as produced by parsymdec.

change a record produced by parsymdec of the block structure of D.

n order of the matrix A

Output from procedure inertia

poseig the number of positive eigenvalues of A.

negeig the number of negative eigenvalues of A.

4. ALGOL program

```
PROCEDURE PARSYMDEC(A,CHANGE,N);
VALUE N; INTEGER N ;
INTEGER ARRAY CHANGE;   ARRAY A;
COMMENT GIVEN A  SYMMETRIC MATRIX A OF ORDER N, THIS
          PROCEDURE COMPUTES THE DECOMPOSITION A = (PM) D (PM)TRANSPOSE
          WHERE M IS A UNIT LOWER TRIANGULAR MATRIX,D IS A BLOCK
          DIAGONAL MATRIX WITH BLOCKS OF ORDER 1 OR 2 AND M(I+1,I)=0
          WHEN D(I+1,I) IS NONZERO, AND P IS A PERMUTATION MATRIX. THIS
          PROCEDURE USES A PARTIAL PIVOTING SCHEME TO FORM THE
          DECOMPOSITION. A IS ASSUMED TO BE STORED ONLY IN ITS UPPER
          TRIANGULAR PART. M AND D ARE WRITTEN OVER A AND THE DIAGONAL
          OF A WILL BE DESTROYED. ON OUTPUT THE INTEGER ARRAY CHANGE OF
          LENGTH N CONTAINS A RECORD OF THE INTERCHANGES PERFORMED,
          I.E. THE PERMUTATION MATRIX P;
BEGIN INTEGER I,J,K,M,IP1,IP2;
     REAL DEN,S,T,ALPHA,LAMBDA,SIGMA,AII,AIP1,AIP1I;

     PROCEDURE INTERCHANGE(I);
     VALUE I; INTEGER I;
     COMMENT THIS PROCEDURE INTERCHANGES ROW AND COLUMN J OF A AND
               ROW AND COLUMN I WHERE I < J, AND A IS THE REDUCED
               MATRIX OF ORDER N-I+1;
     BEGIN FOR K := J+1 STEP 1 UNTIL N DO
          BEGIN T := A[J,K]; A[J,K] := A[I,K] ;
                               A[I,K] := T  END ;
          FOR K := I+1 STEP 1 UNTIL J-1 DO
          BEGIN T := A[I,K]; A[I,K] := A[K,J];
                               A[K,J] := T  END ;
          T := A[I,I]; A[I,I] := A[J,J]; A[J,J] := T
     END INTERCHANGE ;

     ALPHA := (1+SQRT(17)  )/8;
     CHANGE[N] := N ;
     I := 1 ;
REDUCE: IF I < N THEN
     BEGIN IP1 := I+1; IP2 := I+2 ;
          AII := ABS(A[I,I]) ; CHANGE[I] := I ;

          COMMENT FIND THE MAXIMUM ELEMENT IN THE FIRST COLUMN OF
                    THE REDUCED MATRIX BELOW THE DIAGONAL;

          LAMBDA := ABS(A[I,IP1]);  J := IP1 ;
          FOR M := IP2 STEP 1 UNTIL N DO
          IF ABS(A[I,M])> LAMBDA THEN
          BEGIN J :=M ; LAMBDA := ABS(A[I,M]) END ;
          T := ALPHA*LAMBDA ;
          IF AII ≥ T THEN GOTO  ONEBYONE;

          COMMENT DETERMINE THE MAXIMUM ELEMENT IN THE JTH
               COLUMN OF THE REDUCED MATRIX OFF THE DIAGONAL;

          SIGMA := LAMBDA ;
          FOR M := IP1 STEP 1 UNTIL J-1 DO
          IF ABS(A[M,J]) > SIGMA THEN SIGMA := ABS(A[M,J]);
          FOR M := J+1 STEP 1 UNTIL N DO
          IF ABS(A[J,M]) > SIGMA THEN SIGMA := ABS(A[J,M]);
          IF SIGMA*AII ≥ T*LAMBDA THEN GOTO ONEBYONE;
          IF ABS(A[J,J]) ≥ ALPHA*SIGMA THEN
          BEGIN  COMMENT INTERCHANGE THE ITH AND JTH ROW AND
                              COLUMN AND DO A 1 X 1 PIVOT;
```

```
                CHANGE[I] := J; INTERCHANGE(I);  GOTO ONEBYONE;
           END;

           COMMENT WE DO A 2 X 2 PIVOT;

           IF J > IP1 THEN
           BEGIN INTERCHANGE(IP1); T := A[I,J];
                A[I,J] := A[I,IP1];  A[I,IP1] := T;
           END;
           DEN := A[I,I]*A[IP1,IP1]/A[I,IP1]-A[I,IP1];
           AIP1I := A[I,IP1];   AIP1 := A[IP1,IP1];
           AII := A[I,I]/A[I,IP1];
           CHANGE[I] := J;  CHANGE[IP1] := -1;
           FOR J:= IP2 STEP 1 UNTIL N DO
           BEGIN T := (A[I,J] - AII*A[IP1,J])/DEN;
                S :=-(AIP1*T + A[IP1,J])/AIP1I;
                FOR K:= J STEP 1 UNTIL N DO
                     A[J,K] := A[J,K] + A[I,K]*S + A[IP1,K]*T;
                A[I,J] := S;  A[IP1,J] := T
           END J ;
           I:= IP2;
           GOTO REDUCE ;

ONEBYONE:  IF A[I,I]¬= 0 THEN
           BEGIN  AII := A[I,I];
                FOR J:= IP1 STEP 1 UNTIL N DO
                BEGIN  S := -A[I,J]/AII;
                     FOR K:= J STEP 1 UNTIL N DO
                          A[J,K] := A[J,K] + S*A[I,K];
                     A[I,J] := S
                END J ;
           END ;
           I := IP1 ;
           GOTO REDUCE
      END I ;
END PARSYMDEC ;


PROCEDURE PARSYMSOL(A,CHANGE,B,N,FAIL);
VALUE N; INTEGER N;
ARRAY A,B;  INTEGER ARRAY CHANGE;  LABEL FAIL;
BEGIN COMMENT THIS SUBROUTINE USES THE DECOMPOSITION COMPUTED IN THE
      SUBROUTINE PARSYMDEC TO SOLVE A X =B WHERE A IS A NONSINGULAR
      SYMMETRIC MATRIX OF ORDER N AND B IS A VECTOR OF ORDER N. THE
      VECTOR CHANGE OF LENGTH N IS GENERATED IN PARSYMDEC. THE SOLUTION
      IS RETURNED IN THE VECTOR B . IF IT IS DETECTED THAT THE MATRIX
      A IS SINGULAR, CONTROL WILL PASS TO THE LOCATION GIVEN BY THE
      LABEL FAIL;
      REAL DEN,TEMP,SAVE;
      INTEGER II,I,J,K,IP1;

      COMMENT SOLVE M D Y = B AND STORE Y IN B;

      I:=1;
REPEAT: IF I < N THEN
      BEGIN IP1:=I+1;
           SAVE:= B[CHANGE[I]];
           IF CHANGE[IP1] > 0 THEN
           BEGIN B[CHANGE[I]]:=B[I];
                IF A[I,I] = 0 THEN GOTO FAIL;
```

```
                B[I]:=SAVE/A[I,I];
                FORJ:= IP1 STEP 1 UNTIL N DO
                    B[J] := B[J] + A[I,J]*SAVE;
                I := IP1
            END  ELSE
            BEGIN TEMP:= B[I];  B[CHANGE[I]]:=B[IP1];
                DEN := A[I,I]*A[IP1,IP1]/A[I,IP1] - A[I,IP1];
                B[IP1] := (SAVE*A[I,I]/A[I,IP1] - TEMP)/DEN;
                B[I] := (SAVE-A[IP1,IP1]*B[IP1])/A[I,IP1];
                FOR J:= I+2 STEP 1 UNTIL N DO
                    B[J] := B[J] + A[I,J]*TEMP + A[IP1,J]*SAVE;
                I:= I+2;
            END;
            GOTO REPEAT
        END;

        IF I = N THEN
        BEGIN  IF A[I,I] = 0 THEN GOTO FAIL;
            B[I] := B[I]/A[I,I];  I := N-1
        END  ELSE I := N-2;

         COMMENT NOW SOLVE M(TRANSPOSE)X=Y FOR X,WHERE Y IS STORED IN
                 THE VECTOR B AND STORE X IN B ;

CALC: IF I > 0 THEN
        BEGIN IF CHANGE[I] < 0 THEN II:=I-1 ELSE II:= I;
            FOR K:= II STEP 1 UNTIL I DO
            BEGIN  SAVE:= B[K];
                FOR J:= I+1 STEP 1 UNTIL N DO
                    SAVE:= SAVE +A[K,J]*B[J];
                B[K] := SAVE
            END;
            B[I]:= B[CHANGE[II]];  B[CHANGE[II]]:=SAVE;
            I:=II-1;  GOTO CALC
        END
END;


PROCEDURE INERTIA(A,CHANGE,N,POSEIG,NEGEIG);
VALUE N;  INTEGER N;
ARRAY A;  INTEGER ARRAY CHANGE;  INTEGER POSEIG,NEGEIG;
COMMENT THIS PROCEDURE USES THE DECOMPOSITION COMPUTED BY THE
        SUBROUTINE PARSYMDEC TO COMPUTE THE INERTIA OF A
        SYMMETRIC MATRIX A OF ORDER N. THE INTEGER VECTOR CHANGE OF
        LENGTH N IS GENERATED IN PARSYMDEC. THE PROCEDURE RETURNS THE
        NUMBER OF POSITIVE EIGENVALUES IN POSEIG AND THE NUMBER OF
        NEGATIVE EIGENVALUES IN NEGEIG. THE DECOMPOSITION IN THE A
        MATRIX IS NOT AFFECTED;
BEGIN INTEGER I;
      NEGEIG := 0;  POSEIG := 0;
      I:=1;
NEXT: IF I<N THEN
      BEGIN  IF CHANGE[I+1]>0 THEN
           BEGIN COMMENT A 1 X 1 BLOCK HAS BEEN FOUND;
                 IF A[I,I]  < 0 THEN NEGEIG := NEGEIG + 1;
                 IF A[I,I]  > 0 THEN POSEIG := POSEIG + 1;
                 I:=I+1;
           END  ELSE
           BEGIN COMMENT A 2 X 2 BLOCK HAS BEEN FOUND;
                 POSEIG := POSEIG + 1;  NEGEIG := NEGEIG + 1;  I:= I+2;
```

```
                END;
                GOTO NEXT;
            END I;
            IF I = N THEN
            BEGIN  IF A[I,I] > 0 THEN POSEIG := POSEIG +1;
                IF A[I,I] < 0 THEN NEGEIG := NEGEIG + 1;
            END
        END INERTIA;
```

5. Organizational and Notational Details

a. The procedure parsymdec takes A in conventional form as an $n \times n$ array. Without increasing execution time A can be stored as a one dimensional array of length $n(n+1)/2$. This saves storage but the square array has the advantage of being able to hold M, D, and the strictly lower triangular part of A. The diagonal of A must be saved separately. Snapshots of two possible configurations after two steps of the algorithm are given below.

$$\begin{array}{cccccc} d_{11} & d_{21} & m_{31} & m_{41} & m_{51} & m_{61} \\ a_{21} & d_{22} & m_{32} & m_{42} & m_{52} & m_{62} \\ a_{31} & a_{32} & d_{33} & m_{43} & m_{53} & m_{63} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{array} \qquad \begin{array}{cccccc} d_{11} & d_{21} & m_{31} & m_{41} & m_{51} & m_{61} \\ a_{21} & d_{22} & m_{32} & m_{42} & m_{52} & m_{62} \\ a_{31} & a_{32} & d_{33} & d_{43} & m_{53} & m_{63} \\ a_{41} & a_{42} & a_{43} & d_{44} & m_{54} & m_{64} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{array}$$

Note that $m_{j+1,j} d_{j+1,j} = 0$ always, so M and D never overlap.

b. The derivation of the permutation P from the sequence of interchanges is a standard technique, see Wilkinson [10], p. 206, for more details. Let P_i denote the interchange at i. If a 2×2 pivot is used on $A^{(i)}$ then, by convention, $M_{i-1} = P_{i-1} = I$. Also let M^{-t} denote $(M^{-1})^t$.

The block diagonal matrix D is computed in stages as

$$M_{n-1}^{-1} \cdots M_2^{-1} P_2^t M_1^{-1} P_1^t A P_1 M_1^{-t} P_2 M_2^{-t} \cdots M_{n-1}^{-t} = D.$$

This can be rewritten in the form

$$M^{-1}P^tAPM^{-t} = D$$

where

$$P=P_1P_2 \ldots P_{n-1}, M^{-1} = M_{n-1}^{-1} (P_{n-1}^t M_{n-2}^{-1} P_{n-1}) \ldots (P_{n-1}^t \ldots P_2^t M_1^{-1} P_2 \ldots P_{n-1}).$$

The matrices in parentheses are all unit lower triangular.

c. In the table below comparing three different methods for computing factorizations of symmetric matrices, the algorithm comsymdec is the complete pivoting scheme of Bunch and Parlett[5] which searches each $A^{(i)}$ to find its largest element.

Method	Restrictions	Multiplications	Additions	Comparisons	Element Growth
Choleski	pos.def.	$\frac{1}{6}n^3+\frac{3}{2}n^2+\frac{1}{3}n$	$\frac{1}{6}n^3+n^2-\frac{7}{6}n$	0	1
comsymdec	sym.	$\le\frac{1}{6}n^3+\frac{7}{4}n^2-n$	$\le\frac{1}{6}n^3+\frac{5}{4}n^2-\frac{7}{6}n$	$\le\frac{1}{6}n^3+\frac{1}{2}n^2$	$<3nf(n)$*
parsymdec	sym.	$\le\frac{1}{6}n^3+\frac{7}{4}n^2+3n$	$\le\frac{1}{6}n^3+\frac{5}{4}n^2-\frac{7}{6}n$	$\le n^2-1$	$<(2.57)^{n-1}$

* $f(n) = (2^1 \cdot 3^{1/2} \cdot 4^{1/2} \ldots n^{1/(n-1)})^{1/2} \sim n^{(\log n)/4}$.

More details are given in Bunch and Kaufman [4] and Bunch [3].

d. The information describing P is encoded in a single array of length n, called change.

At step j the current matrix is $A^{(n-j+1)}$. If a 1x1 pivot is used and rows 1 and k are exchanged, then change [j] is set to k. If a 2x2 pivot is used and rows 2 and k are exchanged, then change [j] is set to k and change [j+1] is set to -1. Thus, a negative number in change signifies the existence of a 2x2 pivot.

Note that the permutation matrices are not applied directly to M.

e. The test for singularity in parsymsol is the most stringent one, i.e. a test for 0 rather than $\varepsilon||A||$.

f. If the user wishes to recover the original matrix, as for use with iterative refinement, then the original diagonal should be saved before parsymdec is invoked.

6. Discussion of Numerical Properties

Because of rounding errors the computed matrices M and D satisfy

$$MDM^t = A+E$$

It turns out that $||E||$ does not depend on $||M||$ and $||D||$ alone but on $\max_{i \le n} \mu^{(i)}$ where $\mu^{(i)} = \max_{k,j} |a_{k,j}^{(i)}|$.

Thus $\max_{k,j} |e_{k,j}| \le g_n \mu^{(n)}$

where $A^{(n)} = A$ and $g_n = (\max_{i \le n} \mu^{(i)})/\mu^{(n)}$ is the element growth to which reference has been made throughout the article.

In Bunch [3] it is shown that with the algorithm comsymdec, described in Bunch and Parlett [5], $g_n < 3nf(n) \sim 3n^{(1+\log(n)/4)}$ when $\alpha = \alpha_0$, whereas with parsymdec $g_n < (2.57)^{n-1}$ when $\alpha = \alpha_0$. This exponential growth seems alarming but the important fact is that the reduced matrices cannot grow abruptly from step to step. No example is known where significant element growth occurs at every step.

In Bunch and Kaufman [4] it is shown that element growth can be monitored at a modest extra cost. However the extreme rarity of significant growth disuaded us from incorporating this device into parsymdec.

7. Test Results

The execution time for the combination parsymdec and parsymsol has been compared with that of other programs on the CDC 6400 and the Burroughs 6700. The results are given in Tables 1 and 2. In Table 1 the matrices are positive definite and are given by

$$a_{ij} = n+1-i \ , \quad i \geq j \ , \qquad a_{ji} = a_{ij} \quad j>i$$

where n is the order of the matrix. The code choldet1* was obtained by deleting the determinant calculation from choldet1 [7]. In Table 2 the matrices are given by

$$a_{ij} = abs(i-j) \quad i \neq j \ , \qquad a_{ii} = 1.69$$

These matrices are indefinite and the D matrices produced by parsymdec contained a number of 2x2 blocks. The code unsymdet* was obtained by deleting the determinant calculation from unsymdet [2] and computing inner products in line in single precision. In order to make fair comparisons unsymsol [2], which uses a matrix for the righthand side(s), was modified to unsymsol*, which stores b in a one dimensional array.

Timing comparisons are, of course, dependent on the compiler and machine.

Table 1: Execution times on a positive definite matrix

Burroughs 6700

Order of Matrix	parsymdec and parsymsol	choldet 1 and cholsol 1	choldet 1* and cholsol 1
10	.025 sec.	.037 sec.	.037 sec.
20	.123	.205	.183
40	.795	1.265	1.132
80	5.672	8.817	7.883

CDC 6400

	parsymdec and parsymsol	choldet 1* and cholsol 1
10	.027 sec.	.026 sec.
20	.127	.126
40	.742	.777
80	4.957	5.277

Table 2: Execution times on an indefinite matrix

Burroughs 6700

Order of Matrix	parsymdec and parsymsol	unsymdet* (without determinant calculation) and unsymsol*
10	.023 sec.	.047 sec.
20	.127	.275
40	.817	1.902
80	6.003	14.093

CDC 6400

	parsymdec and parsymsol	unsymdet* (without determinant calculation) and unsymsol*
10	.025 sec.	.058 sec.
20	.117	.297
40	.629	2.001
80	4.349	13.395

Table 3: Ratio to parsymdec and parsymsol on the Burroughs 6700

Order of Matrix	parsymdec and parsymsol	choldet 1 and cholsol 1 on a positive def. matrix	choldet 1* and cholsol 1 on a positive def. matrix	unsymdet* and unsymsol* on an indefinite matrix
10	1.00	1.48	1.48	2.04
20	1.00	1.67	1.49	2.17
40	1.00	1.59	1.42	2.33
80	1.00	1.55	1.39	2.35

Table 4: Ratio to parsymdec and parsymsol on the CDC 6400

Order of Matrix	parsymdec and parsymsol	choldet 1* and cholsol 1 on a positive def. matrix	unsymdet* and unsymsol* on an indefinite matrix
10	1.00	.96	2.32
20	1.00	.99	2.54
40	1.00	1.05	3.18
80	1.00	1.06	3.08

In factoring positive definite matrices one might suppose that pivoting would give parsymdec-parsymsol a slight edge over choldet-cholsol 1 as regards accuracy. In many cases, such as Example 1, this is so. However, in Example 2 the Choleski method wins. There are two points to be made here. Firstly, all the answers are satisfactorily accurate. Secondly, pivoting is designed solely to keep step by step element growth to a modest level and that will not necessarily enhance accuracy.

In a similar view we emphasize that the fact that Method A has a smaller error bound than Method B in no way implies that it produces smaller errors. In fact the diagonal pivoting method usually produces slightly better approximations than Gaussian Elimination with partial pivoting.

All examples were run on the CDC 6400, with machine precision $2^{-48} \sim 3.5_{10}-15$.

Matrix					True Solution	Right Hand Side
4	4	24	40	-24	-7	-436
4	7	45	13	-39	-2	-490
24	45	296	91	-289	-1	-3519
40	13	91	964	-420	-4	-8033
-24	-39	-289	-420	572	9	7363

Computed Solutions:

parsymdec and parsymsol	choldet 1* and cholsol 1
-6.9999999999964	-6.9999999969468
-2.0000000000075	-2.0000000078596
- .99999999999910	- .99999999900285
-4.0000000000001	-4.0000000001073
9.0000000000000	9.000000000173

Example 2:

Matrix					True Solution	Right Hand Side
1	-2	3	7	-9	-6	78
-2	8	-6	2	50	-5	-320
3	-6	18	-15	-18	-8	-81
7	2	-15	273	173	5	222
-9	50	-18	173	1667	-7	-10856

Computed Solutions:

parsymdec and parsymsol	choldet 1* and cholsol 1
-5.9999999998895	-6.0000000000000
-4.9999999999870	-5.0000000000000
-8.0000000000170	-8.0000000000000
4.9999999999959	5.0000000000000
-6.9999999999995	-7.0000000000000

Example 3:

Matrix					True Solution	Right Hand Side
-3	-3	-18	-30	18	-7	327
-3	-1	-4	-48	8	-2	291
-18	-4	-6	-274	6	-1	1290
-30	-48	-274	119	19	-4	275
18	8	6	19	216	9	1720

Computed Solutions:

parsymdec and parsymsol	unsymdet* and unsymsol*
-6.9999999999914	-6.9999999922446
-2.0000000000214	-2.0000000207192
- .99999999999727	- .99999999732927
-4.0000000000003	-4.0000000002640
+9.0000000000000	9.0000000000702

Example 4:

Matrix

-4	0	-16	-32	28
0	1	5	10	-6
-16	5	-37	-66	64
-32	10	-66	-85	53
28	-6	64	53	-15

True Solution	Right Hand Side
-8	448
-3	-111
-2	1029
-5	1207
8	-719

Computed Solutions:

parsymdet and parsymsol	unsymdet* and unsymsol*
-8.0000000015746	-7.9999999993815
-3.0000000020098	-2.9999999992122
-1.9999999994367	-2.0000000002219
-5.0000000000716	-4.9999999999711
8.0000000000152	7.9999999999945

References

1. J.O. Aasen, "On the reduction of a symmetric matrix to tridiagonal form", BIT, 11(1971), pp. 233-242.

2. H.J. Bowdler, R.S. Martin, G. Peters, and J. H. Wilkinson, "Solution of Real and Complex Systems of Linear Equations", Numer Math 8, (1966), pp. 217-234.

3. J.R. Bunch, "Analysis of the diagonal pivoting method", SIAM Numerical Analysis, 8 (1971), pp. 656-680.

4. J.R. Bunch and L.C. Kaufman, "Some Stable Methods for Solving Symmetric Linear Systems", Department of Computer Science, University of Colorado, Report #63, March 1975.

5. J.R. Bunch and B.N. Parlett, "Direct Methods for solving symmetric indefinite systems of linear equations", SIAM Numerical Analysis, 8 (1971), pp. 639-655.

6. R.W. Cottle, "Manifestations of the Schur Complement", Linear Algebra and its Applications, 8 (1974), pp. 189-211.

7. R.S. Martin, G. Peters, J.H. Wilkinson, "Symmetric Decomposition of a Positive Definite Matrix", Numer Math 7 (1965), pp. 362-383.

8. B.N. Parlett and J.K. Reid, "On the solution of a system of linear equations whose matrix is symmetric but not definite", BIT, 10 (1970), pp. 386-397.

9. G. Peters and J. H. Wilkinson, "Eigenvalues of $Ax=\lambda Bx$ with band symmetric A and B," Comp. J. 12, 398-404.

10. J.H. Wilkinson, The Algebraic Eigenvalue Problems, Clarendon Press, Oxford, 1965.