Numerische Mathematik 8, 203—216 (1966)

Handbook Series Linear Algebra

Iterative Refinement of the Solution of a Positive Definite System of Equations*

Contributed by R. S. MARTIN, G. PETERS and J. H. WILKINSON

1. Theoretical Background

In an earlier paper in this series [*1*] the solution of a system of equations $Ax=b$ with a positive definite matrix of coefficients was described; this was based on the Cholesky factorization of A. If A is ill-conditioned the computed solution may not be sufficiently accurate, but (provided A is not almost singular to working accuracy) it may be improved by an iterative procedure in which the Cholesky decomposition is used repeatedly.

This procedure may be defined by the following relations

$$x^{(0)}=0, \qquad r^{(s)}=b-Ax^{(s)}, \qquad (LL^T)\,d^{(s)}=r^{(s)}, \qquad (1)$$
$$x^{(s+1)}=x^{(s)}+d^{(s)}.$$

In each iteration the residual $r^{(s)}$ corresponding to the current $x^{(s)}$ is computed and the correction $d^{(s)}$ to be added to $x^{(s)}$ is determined using the computed LL^T factorization.

The only reason that the first solution is not exact is because of the intervention of rounding errors. Since rounding errors are also made in the iterative refinement it is by no means obvious that the successive $x^{(s)}$ will be improved solutions. It has been shown by WILKINSON [*2*] that if A is not "too ill-conditioned" the $x^{(s)}$ will converge to the correct solution "to working accuracy" provided $r^{(s)}$ is computed to double-precision. Since the elements of A and of $x^{(s)}$ are single-precision numbers this may be achieved merely by accumulating inner-products. The remaining steps may be performed in single-precision.

Two terms used above call for some explanation. We say that $x^{(s)}$ is the correct solution "to working accuracy" if $\|x^{(s)}-x\|_\infty/\|x\|_\infty \leqq 2^{-t}$ where the mantissae of our floating-point numbers have t binary digits. For the meaning of the term "too ill-conditioned" we must refer to the error analysis of the Cholesky method. It has been shown ([*2*, *3*]) that the computed solution of $Ax=b$ is the exact solution of a system $(A+\delta A)x=b$ where δA is dependent on b but is uniformly bounded. The upper bound m of $\|\delta A\|_2$ is dependent on the details of the arithmetic used, particularly upon whether or not inner-products are

* *Editor's note.* In this fascicle, prepublication of algorithms from the Linear Algebra series of the Handbook for Automatic Computation is continued. Algorithms are published in ALGOL 60 reference language as approved by the IFIP. Contributions in this series should be styled after the most recently published ones. Inquiries are to be directed to the editor. F. L. BAUER, Munich

accumulated without intermediate rounding. We shall say that A is "too ill-conditioned" for the precision of computation that is being used if $m\|A^{-1}\|_2 \geqq 1$. In this case $A+\delta A$ could be singular and in any case $(A+\delta A)^{-1}b$ may not agree with $A^{-1}b$ in any of its figures. If $m\|A^{-1}\|_2 = 2^{-p}$ $(p>0)$ then successive iterates certainly satisfy the relation

$$\|x - x^{(s+1)}\|_2 \leqq \|x - x^{(s)}\|_2 2^{-p}/(1-2^{-p}). \tag{2}$$

Usually the statistical distribution of rounding errors will ensure that the $x^{(s)}$ will improve more rapidly than this.

A similar process may be used to refine an inverse derived from the Cholesky factorization of A as described in [1]. If we denote the first computed inverse by $X^{(1)}$ then we have the iterative refinement procedure defined by

$$\begin{aligned} X^{(s+1)} &= X^{(s)} + X^{(s)}(I - AX^{(s)}), \\ &= X^{(s)} + X^{(s)} B^{(s)}, \\ &= X^{(s)} + Z^{(s)}. \end{aligned} \tag{3}$$

In order to be effective it is essential that the right-hand side of (3) should not be expressed in the form $2X^{(s)} - X^{(s)}AX^{(s)}$. The "residual matrix" $I - AX^{(s)}$ must be computed using double-precision or accumulation of inner-products, each component being rounded only on completion. The conditions for convergence are the same as those for the iterative refinement of the solution of $Ax = b$, but if the condition is satisfied the convergence is effectively quadratic. Indeed ignoring rounding errors we have

$$I - AX^{(s+1)} = (I - AX^{(s)})^2. \tag{4}$$

In [1] several different procedures were given based on symmetric decompositions of A; here we give only the procedures related to *choldet 1*, *cholsol 1* and *cholinversion 1*. Analogous procedures could be made based on *symdet*, *symsol* and *syminversion*. Since A must be retained in order to form the residuals the decompositions in which the upper half of A is stored as a linear array of $\frac{1}{2}n(n+1)$ elements are not relevant.

2. Applicability

These procedures may be used to compute accurate solutions of systems of equations with positive definite matrices or to compute accurate inverses of such matrices. An essential feature of the algorithms is that in each iteration an accurate residual must be determined corresponding to the current solution. The i-th components r_i of a residual is given by

$$r_i = b_i - \sum_{j=1}^{n} a_{ij} x_j, \tag{5}$$

and since only the upper-half of A is stored this must be written in the form

$$r_i = b_i - \sum_{j=1}^{i-1} a_{ji} x_j - \sum_{j=i}^{n} a_{ij} x_j, \tag{6}$$

taking advantage of the symmetry of A. Similar remarks apply to the computation of $I-AX$ in the procedure for an accurate inverse.

In order to avoid an excessively complicated inner-product procedure it is convenient to compute the expression on the right-hand side of (6) in two stages. In the first stage $s_i=b_i-\sum_{j=1}^{i-1} a_{ji}x_j$ is computed and in the second stage $\sum_{j=i}^{n} a_{ij}x_j$ is subtracted from s_i. It is essential that s_i should not be rounded before subtracting the second inner-product since if this is done the refinement procedure is completely ineffective. This means that the inner-product procedure should be capable of starting from a previously computed double-precision inner-product when necessary.

Modified versions of *choldet 1*, *cholsol 1* and *cholinversion 1* are included in which the required inner-products are computed by means of the inner-product procedure used for the residuals. On many computers the accurate computation of inner-products may be very time consuming and in this case the versions of *choldet 1* etc. given in [1] may be used.

The procedures are

acc solve Corresponding to r given right-hand sides this either produces the correctly rounded solutions of the equation $Ax=b$ or indicates that A is too ill-conditioned for this to be achieved without working to higher precision (or is possibly singular). *acc solve* must be preceded by *choldet 1* so that the Cholesky decomposition of A is available. *acc solve* may be used any number of times after one use of *choldet 1*.

acc inverse This either produces the correctly rounded inverse of a positive definite matrix or indicates that it is too ill-conditioned. The first approximation to the inverse is produced by *cholinversion 1*.

3. Formal Parameter List

3.1. Input to procedure *acc solve*.

n order of the matrix A.

r number of right-hand sides for which $Ax=b$ is to be solved.

a an $n\times n$ array consisting of the upper-triangle of A and the subdiagonal elements of the lower-triangle L of the Cholesky decomposition of A produced by procedure *choldet 1*.

p the reciprocals of the diagonal elements of L as produced by procedure *choldet 1*.

b the $n\times r$ matrix consisting of the r right-hand sides.

eps the largest number for which $1+eps=1$ on the computer.

Output of procedure *acc solve*.

x the $n\times r$ matrix consisting of the r solution vectors.

l the number of iterations.

bb the $n\times r$ matrix consisting of the r residual vectors.

ILL This is the exit which is used when there is no perceptible improvement from one iteration to the next.

3.2. Input to procedure *acc inverse*.

n order of the matrix A.

a an $(n+1)\times n$ array the elements of the upper-triangle being those of A.
eps the largest number for which $1+eps=1$ on the computer.

Output of procedure *acc inverse*.

a an $(n+1)\times n$ array the elements of the upper-triangle being those of A and the elements of the lower-triangle those of X, the accepted inverse.
l the number of corrections added to the first inverse.
fail the exit which is used if A, possibly as the result of rounding errors, is not positive definite.
ILL the exit used if there is no perceptible improvement from one iteration to the next. Both "*fail*" and "*ILL*" indicate that A is too ill-conditioned for the algorithm to be successful without working to higher precision.

4. ALGOL Programs

```
procedure innerprod(l, s, u, c1, c2, ak, bk) bound variable:(k) result:(d1, d2);
value l, s, u, c1, c2;
integer l, s, u, k;
real c1, c2, ak, bk, d1, d2;
comment This procedure accumulates the sum of products ak×bk and adds
        it to the initial value (c1, c2) in double precision. The bound variable
        k is to be used to indicate the subscript in the components of the vec-
        tors ak, bk over which a scalarproduct is to be formed.
        Throughout the Handbook Series Linear Algebra, the actual para-
        meters corresponding to ak and bk will be restricted to be real variables.
        If they are subscripted, all subscript expressions will be linear func-
        tions of the bound variable parameter k, or will be independent of k.
        This allows high efficiency handcoding of the subscript evaluation in
        the for statement (loop) of the procedure.
        The body of this procedure can not be expressed within ALGOL,
begin real s1, s2,
    (s1, s2) := c1+c2,                    comment dbl. pr. acc,
    for k := l step s until u do
    (s1, s2) := (s1, s2)+ak×bk,           comment dbl. pr. acc,
    d1 := (s1, s2)rounded,
    d2 := ((s1, s2) - d1)rounded
end innerprod;
```

```
procedure choldet1 (n) data and result:(a) result:(p, d1, d2) failure exit:(fail);
value n; integer d2, n; real d1; array a, p; label fail;
comment The upper triangle of a positive definite symmetric matrix, A, is
        stored in the upper triangle of an n×n array a[i, j], i = 1(1)n, j = 1(1)n.
        The Cholesky decomposition A = LU, where U is the transpose of L,
        is performed and stored in the remainder of the array a except for
        the reciprocals of the diagonal elements which are stored in p[i],
        i = 1(1)n, instead of the elements themselves. A is retained so that
        the solution obtained can be subsequently improved. The determi-
        nant, d1×2↑d2, of A is also computed. The procedure will fail if A,
```

```
        modified by the rounding errors, is not positive definite. Uses the
        procedure innerprod;
begin   integer i, j, k; real x, xx;
        d1 := 1;
        d2 := 0;
        for i := 1 step 1 until n do
        for j := i step 1 until n do
        begin innerprod(1, 1, i-1, -a[i, j], 0, a[i, k], a[j, k], k, x, xx);
              x := -x;
              if j=i then
              begin
                    d1 := d1×x; if x≦0 then goto fail;
              L1:   if abs(d1) ≧ 1 then
                    begin d1 := d1×0.0625;
                          d2 := d2+4;
                          go to L1
                    end;
              L2:   if abs(d1) < 0.0625 then
                    begin d1 := d1×16;
                          d2 := d2-4;
                          go to L2
                    end;
                    p[i] := 1/sqrt(x)
              end
              else
              a[j, i] := x×p[i]
        end ij
end choldet1;

procedure acc solve(n) data: (r, a, p, b, eps) result: (x, bb, l) failure exit: (ILL);
value   n, r, eps;
integer n, r, l;
real    eps;
array   a, p, x, b, bb;
label   ILL;
comment Solves A x=b where A is an n×n positive definite symmetric matrix
        and b is an n×r matrix of right hand sides, using the procedure
        cholsol1. The procedure must be preceded by choldet1 in which L is
        produced in a[i, j] and p[i]. The residuals bb=b-A x are calculated
        and A d=bb is solved, overwriting d on bb. The refinement is re-
        peated, as long as the maximum correction at any stage is less than
        half that at the previous stage, until the maximum correction is less
        than 2epsilon times the maximum x. Exits to label ILL if the solution
        fails to improve. Uses the procedure innerprod which forms accurate
        innerproducts. l is the number of iterations;
```

```
begin
    procedure cholsol 1 (n) data: (r, a, p, b) result: (x);
    value r, n; integer r, n; array a, b, p, x;
    comment Solves A x=b, where A is a positive definite symmetric
            matrix and b is an n×r matrix of r right-hand sides. The
            procedure cholsol 1 must be preceded by choldet 1 in which
            L is produced in a[i, j] and p[i], from A. A x=b is solved
            in two steps, L y=b and U x=y, and x is overwritten on y;
    begin   integer i, j, k; real y, yy;
            for j := 1 step 1 until r do
            begin comment solution of L y=b;
                  for i := 1 step 1 until n do
                  begin innerprod(1, 1, i−1, b[i, j], 0, a[i, k],
                                                  x[k, j], k, y, yy);
                        x[i, j] := −p[i]×y
                  end i;
                  comment solution of U x=y;
                  for i := n step −1 until 1 do
                  begin innerprod(i+1, 1, n, x[i, j], 0, a[k, i],
                                                  x[k, j], k, y, yy);
                        x[i, j] := −p[i]×y
                  end i
            end j
    end cholsol 1;

    integer i, j, k, d2;
    real d0, d1, c, cc, xmax, bbmax;
    for j := 1 step 1 until r do
    for i := 1 step 1 until n do
    begin   x[i, j] := 0;
            bb[i, j] := b[i, j];
    end;
    l := 0;
    d0 := 0;
L3: cholsol 1(n, r, a, p, bb, bb);
    l := l+1;
    d2 := 0;
    d1 := 0;
    for j := 1 step 1 until r do
    for i := 1 step 1 until n do
    x[i, j] := x[i, j] + bb[i, j];
    for j := 1 step 1 until r do
    begin   xmax := bbmax := 0;
            for i := 1 step 1 until n do
```

```
                begin if abs(x[i,j])>xmax then xmax:=abs(x[i,j]);
                      if abs(bb[i,j])>bbmax then bbmax:=abs(bb[i,j]);
                      innerprod(1, 1, i-1, -b[i,j], 0, a[k,i],
                                                  x[k,j], k, c, cc);
                      innerprod(i, 1, n, c, cc, a[i,k], x[k,j], k, c, cc);
                                                  bb[i,j]:=-c
                end;
                if bbmax/xmax>d1 then d1:=bbmax/xmax;
                if bbmax>2×eps×xmax then d2:=1;
          end;
          if d1>d0/2 and l+1 then goto ILL;
          d0:=d1; if d2=1 then goto L3;
end acc solve;

procedure acc inverse(n) data:(eps) data and result:(a) result:(l)
          failure exit:(fail, ILL);
value  n, eps;
integer n, l;
real   eps;
array  a;
label  fail, ILL;
comment The upper triangle of a positive definite symmetric matrix, A, is
        stored in the upper triangle of an (n+1)×n array a[i,j], i=1(1)n+1,
        j=1(1)n. X, the inverse of A, is formed in the remainder of the array
        a[i,j] by the procedure cholinversion 1. The inverse is improved by
        calculating X=X+Z until the correction, Z, is such that maximum
        abs(Z[i,j]) is less than 2eps times maximum abs(X[i,j]), where
        Z=XB and B=I-AX. b is an n×n array and z is a 1×n array,
        X being overwritten a row at a time. Exits to the label fail if A is
        not positive definite and to the label ILL if the maximum correction
        at any stage is not less than half that at the previous stage. l is the
        number of corrections applied. Uses the procedure innerprod;
begin
        procedure cholinversion 1 (n) data and result:(a) failure exit:(fail);
        value n; integer n; array a; label fail;
        comment The upper triangle of a positive definite symmetric matrix,
                A, is stored in the upper triangle of an (n+1)×n array
                a[i,j], i=1(1)n+1, j=1(1)n. The Cholesky decomposition
                A=LU, where U is the transpose of L, is performed and
                L is stored in the remainder of the array a. The reciprocals
                of the diagonal elements are stored instead of the elements
                themselves. L is then replaced by its inverse and this in
                turn is replaced by the lower triangle of the inverse of A.
                A is retained so that the inverse can be subsequently
                improved. The procedure will fail if A, modified by the
                rounding errors, is not positive definite. Uses the procedure
                innerprod;
```

```
begin   integer i, j, k, i1, j1;
        real x, xx, y;
        comment formation of L;
        for i := 1 step 1 until n do
        begin i1 := i+1;
              for j := i step 1 until n do
              begin j1 := j+1;
                    innerprod(1, 1, i-1, -a[i, j], 0,
                          a[i1, k], a[j1, k], k, x, xx); x := -x;
                    if j=i then
                    begin if x≦0 then go to fail;
                          a[i1, i] := y := 1/sqrt(x)
                    end
                    else
                    a[j1, i] := x×y
              end j
        end i;
        comment inversion of L;
        for i := 1 step 1 until n do
        for j := i+1 step 1 until n do
        begin j1 := j+1; innerprod(i, 1, j-1, 0, 0, a[j1, k],
                                          a[k+1, i], k, x, xx);
              a[j1, i] := -a[j1, j]×x
        end ij;
        comment calculation of the inverse of A;
        for i := 1 step 1 until n do
        for j := i step 1 until n do
        innerprod(j+1, 1, n+1, 0, 0, a[k, j],
                                    a[k, i], k, a[j+1, i], xx)
end cholinversion 1;

integer i, j, k, j1;
real c, d, xmax, zmax, e;
array b[1:n, 1:n], z[1:n];
e := 1;
l := 0;
cholinversion 1 (n, a, FAIL);
L1: for i := 1 step 1 until n do
for j := 1 step 1 until n do
begin j1 := j+1;
        if j≧i then
        begin innerprod(1, 1, i, if i=j then -1 else 0, 0,
                    a[k, i], a[j1, k], k, c, d);
              innerprod(i+1, 1, j, c, d, a[i, k], a[j1, k], k, c, d);
              innerprod(j1, 1, n, c, d, a[i, k], a[k+1, j], k, c, d)
        end
        else
```

```
                begin innerprod(1, 1, j, if i=j then −1 else 0, 0,
                                a[k, i], a[j1, k], k, c, d);
                      innerprod(j1, 1, i, c, d, a[k, i], a[k+1, j], k, c, d);
                      innerprod(i+1, 1, n, c, d, a[i, k], a[k+1, j], k, c, d)
                end;
          b[i, j] := −c
          end;
          xmax := zmax := 0;
          for i := 1 step 1 until n do
          begin     for j := 1 step 1 until i do
                    begin innerprod(1, 1, i, 0, 0, a[i+1, k], b[k, j], k, c, d);
                          innerprod(i+1, 1, n, c, d, a[k+1, i], b[k, j], k, z[j], d)
                    end;
                    for j := 1 step 1 until i do
                    begin c := abs(a[i+1, j]);
                          d := abs(z[j]);
                          if c > xmax then xmax := c;
                          if d > zmax then zmax := d;
                          a[i+1, j] := a[i+1, j]+z[j];
                    end;
          end;
          l := l + 1;
          d := zmax/xmax;
          if d > e/2 then goto ILL;
          e := d;
          if d > 2×eps then goto L1;
end acc inverse;
```

5. Organisational and Notational Details

The details of *choldet 1*, *cholsol 1* and *cholinversion 1* are almost identical with those given in [1] except that the inner-products are all performed using the accurate inner-product procedure. This is not an essential change but it increases to some extent the range of matrices for which the iterative refinement will be successful.

In *acc solve* the first set of r solution vectors are taken to be null vectors so that the first residuals are the original right-hand sides. The successive residuals $r^{(s)}$ are stored in the array *bb* and are overwritten by the corresponding $d^{(s)}$. Iteration is terminated when

$$\|d^{(s)}\|_\infty \leqq 2 \text{ eps } \|x^{(s+1)}\|_\infty \tag{7}$$

for *all* right hand sides simultaneously, where *eps* is the relative machine precision. If in any iteration

$$\max\left(\|d^{(s)}\|_\infty / \|x^{(s+1)}\|_\infty\right) > \tfrac{1}{2} \max\left(\|d^{(s-1)}\|_\infty / \|x^{(s)}\|_\infty\right) \tag{8}$$

(where the maximization is over the r right-hand sides) the matrix is too ill-conditioned for reliable solutions to be obtained and a failure indication is given. The process may fail also at the stage when *choldet 1* is performed if A is almost singular to working accuracy.

In the inner-product procedure the parameters *c1* and *c2* are effectively the two halves of a double-precision number. They are both normalized single precision numbers and added together give an initial value to which the inner-product is added. Since the inner-product procedure must be performed in machine code, the means used to achieve the required effect will differ from one computer to another.

acc solve may be used any number of times after *choldet 1*. In practice if there are storage problems we may take $r=1$ and process one right-hand side at a time.

In *acc inverse* the lower triangles of successive approximations to the inverse are all overwritten on the original approximation produced by *cholinversion 1*. In each iteration the whole of the $n\times n$ matrix $B^{(s)}=I-AX^{(s)}$ is produced and stored in the array b. The matrix $Z^{(s)}=X^{(s)}B^{(s)}$ is produced a row at a time, remembering that since $Z^{(s)}$ and $X^{(s)}$ are symmetric we require only elements 1 to i of row i of $Z^{(s)}$. Each of the partial rows of $Z^{(s)}$ is overwritten in the $1\times n$ array z. When each partial row is completed it can be added to $X^{(s)}$ since the corresponding row of $X^{(s)}$ is not required in the computation of the remaining rows of $X^{(s)}B^{(s)}$.

6. Discussion of the Numerical Properties

The error analysis of the procedures *choldet 1, cholsol 1* and *cholinversion 1* was discussed in [*1*]. When inner-products are accumulated the computed LL^T satisfies the relation

$$LL^T=A+F \quad \text{where} \quad \|F\|_2\leqq k_1 n^{\frac{1}{2}}2^{-t}\|A\|_2, \tag{9}$$

where here and later k_i is used to denote a constant of order unity. The computed solution $\bar{x}$ of $Ax=b$ satisfies the relation

$$(A+G(b))\,\bar{x}=b \quad \text{where} \quad \|G(b)\|_2\leqq k_2 n^{\frac{1}{2}}2^{-t}\|A\|_2. \tag{10}$$

Although $G(b)$ is dependent on b it is uniformly bounded. From (10) we have

$$\|\bar{x}-x\|_2/\|x\|_2<\alpha/(1-\alpha), \tag{11}$$

where

$$\alpha=k_2 n^{\frac{1}{2}}2^{-t}\|A\|_2\|A^{-1}\|_2=k_2 n^{\frac{1}{2}}2^{-t}\varkappa(A). \tag{12}$$

The iterative refinement of a solution therefore converges if

$$\alpha/(1-\alpha)<1 \quad \text{i.e.} \quad \alpha<\tfrac{1}{2}.$$

For segments of the Hilbert matrix and similar matrices even the bound for $G(b)$ given above, satisfactory though it is, is a severe overestimate. This is well illustrated by the example in section 8.

The computed inverse X given by *cholinversion 1* with accumulation of inner-products satisfies the relation

$$\|X-A^{-1}\|_2/\|A^{-1}\|_2<\beta/(1-\beta), \tag{13}$$

where

$$\beta=k_3 n^{\frac{1}{2}}2^{-t}\varkappa(A) \tag{14}$$

and convergence is therefore guaranteed if $\beta<\frac{1}{2}$.

It is natural to ask whether *acc solve* can converge to a wrong solution. From the error analysis we know that this is impossible if $\alpha < \frac{1}{2}$. We observe that if this condition is not satisfied A is very ill-conditioned and we have an additional safeguard in *choldet 1*. It seems to be extremely difficult to construct an ill-conditioned matrix which does not reveal its shortcomings either in *choldet 1* or in the iterative refinement, and in practice it appears to be very safe to take the results of *acc solve* at their face value.

With *acc inverse* the situation is theoretically stronger since we have $I - AX$ for the alleged inverse X. If $I - AX = B$ and $\|B\| < 1$ in any norm then A is non-singular and

$$\|X - A^{-1}\| / \|A^{-1}\| \leqq \|B\|. \tag{15}$$

If $\|B\| < \frac{1}{2}$ we certainly have a guarantee that the process will converge to the correctly rounded solution. When solutions corresponding to many different right-hand sides are required it seems attractive to compute X using *acc inverse* particularly since this takes full advantage of symmetry. If the iterative refinement converges and $\|I - AX\|_\infty$ for the accepted inverse is appreciably less than unity then we have a rigorous proof that X is the correctly rounded inverse. Unfortunately this does not mean that Xb is the correctly rounded solution corresponding to any right-hand side b. (For a discussion see [2], Chapter 3.) We would still have to perform an iterative process

$$x^{(s)} = 0, \qquad r^{(s)} = b - A\,x^{(s)}, \qquad x^{(s+1)} = x^{(s)} + X r^{(s)} \tag{16}$$

and its rate of convergence is no better than that of *acc solve*! When A is large it seems difficult to justify the use of *acc inverse* unless one is specifically interested in the inverse itself.

7. Examples of the Use of the Procedures

The uses of these procedures are self-evident.

8. Test Results

The procedure *acc solve* and *acc inverse* have been used on KDF 9 in connexion with the leading principal minor of order seven of the Hilbert matrix. KDF 9 has a 39 binary digit mantissa and is well designed for the accumulation of inner-products. As in [1], the matrix was scaled by the factor 360360 so that all coefficients were integers. In order to provide a comparison with the results given in [1] the accurate inner-product procedure was used in *choldet 1*, *cholsol 1* and *cholinversion 1* as well as in the calculation of the residuals.

acc solve was used with the seven right-hand sides given by the columns of the unit matrix. For economy of presentation we give only the results corresponding to the first column; the behaviour of the other columns is exactly analogous. The initial solution was correct to within 1 part in 10^7, most components being even more accurate than this. This is between two and three decimals better than the results obtained in [1] for *choldet 1* without accumulation of inner-products. The second solution was already correct to working accuracy and the third iterative merely served to show that the second was correct. (It

should be appreciated that all published values were obtained by converting the values in the computer from binary to decimal. If the $d^{(1)}$ in the computer had been added to the $x^{(1)}$ in the computer without rounding, the result would have been correct to more than single-precision.) Notice that the first set of residuals is the smaller in spite of the fact that it corresponds to a less accurate solution. This is predictable from the error analysis. The modified *choldet 1* also gave a considerably more accurate determinant evaluation. Comparative results are

choldet 1 (without accumulation)	$8.4741\,88295\,03_{10}-2\times 2^{52}$;
choldet 1 (with accumulation)	$8.4735\,38254\,98_{10}-2\times 2^{52}$;
Correct value	$8.4735\,3913\ldots_{10}-2\times 2^{52}$.

acc solve was also used with the seven right-hand sides given by the columns of ($360360\times$the unit matrix) for which the solutions are integers. As before the second iteration gave results which were "correct to working accuracy" but since the components are integers this means that the rounded components of $x^{(1)}+d^{(1)}$ were exact. Hence at the second iteration all residuals were exactly zero. For economy we present the results corresponding to the fifth column which has the solution with the largest norm.

acc inverse was also used in connexion with the scaled Hilbert matrix, the first solution being derived from the modified *cholinversion 1* with the accurate inner-product for comparison with the results in [*1*]. It will be seen that the inverse obtained using inner-product accumulation is much the more accurate. The second computed inverse was correct to working accuracy, the third inverse being necessary only to confirm this.

Example

"acc solve" with right-hand side (1, 0, ..., 0)

$d^{(0)}$	$x^{(1)}=0+d^{(0)}$	$r^{(1)}=b-Ax^{(1)}$
$+1.3597\,5135\,069_{10}-4$	$+1.3597\,5135\,069_{10}-4$	$-4.9253\,7921\,559_{10}-9$
$-3.2634\,0327\,676_{10}-3$	$-3.2634\,0327\,676_{10}-3$	$-5.8762\,2217\,552_{10}-9$
$+2.4475\,5247\,690_{10}-2$	$+2.4475\,5247\,690_{10}-2$	$-7.9203\,1329\,411_{10}-10$
$-8.1585\,0829\,840_{10}-2$	$-8.1585\,0829\,840_{10}-2$	$+2.1578\,9164\,315_{10}-10$
$+1.3461\,5387\,384_{10}-1$	$+1.3461\,5387\,384_{10}-1$	$+2.4667\,5568\,860_{10}-10$
$-1.0769\,2310\,159_{10}-1$	$-1.0769\,2310\,159_{10}-1$	$+1.6282\,6196\,970_{10}-10$
$+3.3333\,3341\,518_{10}-2$	$+3.3333\,3341\,518_{10}-2$	$+1.1155\,1656\,801_{10}-10$

$d^{(1)}$	$x^{(2)}=(x^{(1)}+d^{(1)})$ *rounded*	$r^{(2)}=b-Ax^{(2)}$
$+9.0578\,0905\,471_{10}-13$	$+1.3597\,5135\,975_{10}-4$	$+5.4264\,0421\,486_{10}-9$
$+1.3358\,5198\,681_{10}-11$	$-3.2634\,0326\,340_{10}-3$	$+4.8596\,3802\,532_{10}-9$
$-2.9347\,7858\,424_{10}-10$	$+2.4475\,5244\,755_{10}-2$	$+4.3655\,7101\,580_{10}-9$
$+1.3989\,3877\,246_{10}-9$	$-8.1585\,0815\,850_{10}-2$	$+3.9498\,6887\,642_{10}-9$
$-2.7683\,6240\,838_{10}-9$	$+1.3461\,5384\,615_{10}-1$	$+3.6004\,6570\,336_{10}-9$
$+2.4665\,7452\,585_{10}-9$	$-1.0769\,2307\,692_{10}-1$	$+3.3047\,1827\,681_{10}-9$
$-8.1851\,5052\,054_{10}-10$	$+3.3333\,3333\,333_{10}-2$	$+3.0520\,9013\,618_{10}-9$

"acc solve" with right-hand side (0, ..., 0, 360360, 0, 0)

$d^{(0)}$	$x^{(1)}=0+d^{(0)}$	$r^{(1)}=b-A\,x^{(1)}$
$+4.8510\,0010\,033_{10}+4$	$+4.8510\,0010\,033_{10}+4$	$-3.9145\,2026\,367_{10}+0$
$-1.9404\,0011\,270_{10}+6$	$-1.9404\,0011\,270_{10}+6$	$-6.4347\,3815\,918_{10}+0$
$+1.8711\,0013\,720_{10}+7$	$+1.8711\,0013\,720_{10}+7$	$-7.7243\,0419\,922_{10}-1$
$-7.2765\,0058\,982_{10}+7$	$-7.2765\,0058\,982_{10}+7$	$+2.4765\,7775\,879_{10}-1$
$+1.3340\,2511\,415_{10}+8$	$+1.3340\,2511\,415_{10}+8$	$+3.1143\,1884\,766_{10}-1$
$-1.1525\,9770\,196_{10}+8$	$-1.1525\,9770\,196_{10}+8$	$+2.9814\,1479\,492_{10}-1$
$+3.7837\,8034\,226_{10}+7$	$+3.7837\,8034\,226_{10}+7$	$+2.3474\,1210\,938_{10}-1$

$d^{(1)}$	$x^{(2)}=x^{(1)}+d^{(1)}$ *(rounded)*	$r^{(2)}=b-A\,x^{(2)}$
$-1.0032\,6554\,478_{10}-3$	$+4.8510\,0000\,000_{10}+4$	$+0.0000\,0000\,000$
$+1.1270\,1429\,771_{10}-1$	$-1.9404\,0000\,000_{10}+6$	$+0.0000\,0000\,000$
$-1.3720\,0943\,951_{10}-0$	$+1.8711\,0000\,000_{10}+7$	$+0.0000\,0000\,000$
$+5.8981\,9405\,087_{10}+0$	$-7.2765\,0000\,000_{10}+7$	$+0.0000\,0000\,000$
$-1.1415\,0403\,981_{10}+1$	$+1.3340\,2500\,000_{10}+8$	$+0.0000\,0000\,000$
$+1.0196\,2902\,556_{10}+1$	$-1.1525\,9760\,000_{10}+8$	$+0.0000\,0000\,000$
$-3.4226\,0782\,269_{10}+0$	$+3.7837\,8000\,000_{10}+7$	$+0.0000\,0000\,000$

acc inverse. First two solutions (lower triangle only)

1st inverse	2nd inverse
$+1.3597\,5135\,083_{10}-4;$	$+1.3597\,5135\,975_{10}-4;$
$-3.2634\,0327\,687_{10}-3;$	$-3.2634\,0326\,340_{10}-3;$
$+1.0442\,8907\,373_{10}-1;$	$+1.0442\,8904\,429_{10}-1;$
$+2.4475\,5247\,701_{10}-2;$	$+2.4475\,5244\,755_{10}-2;$
$-8.8111\,8918\,469_{10}-1;$	$-8.8111\,8881\,120_{10}-1;$
$+7.9300\,7038\,876_{10}-0;$	$+7.9300\,6993\,007_{10}+0;$
$-8.1585\,0829\,852_{10}-2;$	$-8.1585\,0815\,850_{10}-2;$
$+3.1328\,6729\,439_{10}+0;$	$+3.1328\,6713\,287_{10}+0;$
$-2.9370\,6313\,415_{10}+1;$	$-2.9370\,6293\,706_{10}+1;$
$+1.1188\,8120\,353_{10}+2;$	$+1.1188\,8111\,888_{10}+2;$
$+1.3461\,5387\,384_{10}-1;$	$+1.3461\,5384\,615_{10}-1;$
$-5.3846\,1569\,684_{10}+0;$	$-5.3846\,1538\,461_{10}+0;$
$+5.1923\,0807\,304_{10}+1;$	$+5.1923\,0769\,231_{10}+1;$
$-2.0192\,3093\,289_{10}+2;$	$-2.0192\,3076\,923_{10}+2;$
$+3.7019\,2339\,366_{10}+2;$	$+3.7019\,2307\,692_{10}+2;$
$-1.0769\,2310\,159_{10}-1;$	$-1.0769\,2307\,692_{10}-1;$
$+4.4307\,6950\,879_{10}+0;$	$+4.4307\,6923\,076_{10}+0;$
$-4.3615\,3880\,096_{10}+1;$	$-4.3615\,3846\,154_{10}+1;$
$+1.7230\,7706\,914_{10}+2;$	$+1.7230\,7692\,308_{10}+2;$
$-3.1984\,6182\,140_{10}+2;$	$-3.1984\,6153\,846_{10}+2;$
$+2.7913\,8486\,830_{10}+2;$	$+2.7913\,8461\,539_{10}+2;$
$+3.3333\,3341\,518_{10}-2;$	$+3.3333\,3333\,333_{10}-2;$
$-1.4000\,0009\,298_{10}+0;$	$-1.4000\,0000\,000_{10}+0;$
$+1.4000\,0011\,372_{10}+1;$	$+1.4000\,0000\,000_{10}+1;$
$-5.6000\,0048\,990_{10}+1;$	$-5.6000\,0000\,000_{10}+1;$
$+1.0500\,0009\,497_{10}+2;$	$+1.0500\,0000\,000_{10}+2;$
$-9.2400\,0084\,947_{10}+1;$	$-9.2399\,9999\,999_{10}+1;$
$+3.0800\,0028\,545_{10}+1;$	$+3.0800\,0000\,000_{10}+1;$

It should be appreciated that $-9.2399\,9999\,999_{10}+1$ is the correctly rounded decimal equivalent of the correctly rounded binary representation of $9.24_{10}+1$!

A second test was performed on the computer TR 4 at the Mathematisches Institut der Technischen Hochschule, München. The TR 4 has effectively a 38 binary digit mantissa; since it is not well adapted for the accumulation of inner-products, the latter were computed using true double-precision arithmetic. The results obtained generally confirmed those obtained on KDF 9 but as the rounding procedures are somewhat less satisfactory on TR 4 the accuracy of the first solutions was significantly poorer. However, in all cases the second solution was correct apart from the end figure and the third iteration served only to show that the second was correct. Because of the rounding procedures zero residuals are never obtained on TR 4 and this will be true of many other computers.

Acknowledgements. The authors wish to thank Dr. C. REINSCH of the Mathematisches Institut der Technischen Hochschule, München for a number of valuable comments and for his assistance in testing these algorithms. The work described here is part of the Research Programme of the National Physical Laboratory and this paper is published by permission of the Director of the Laboratory.

References

[1] MARTIN, R. S., G. PETERS, and J. H. WILKINSON: Symmetric decompositions of a positive definite matrix. Numerische Mathematik **7**, 362–383 (1965).

[2] WILKINSON, J. H.: Rounding errors in algebraic processes. London: Her Majesty's Stationary Office; Englewood Cliffs, N.J.: Prentice-Hall 1963.

[3] — The algebraic eigenvalue problem. London: Oxford University Press 1965.

National Physical Laboratory
Mathematics Division
Teddington, Middlesex (Great Britain)