CS 133

HANDBOOK SERIES LINEAR ALGEBRA

SINGULAR VALUE DECOMPOSITION AND LEAST SQUARES SOLUTIONS

BY

G. H. GOLUB
AND
C. REINSCH

TECHNICAL REPORT NO. CS 133
MAY 1969

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

42

HANDBOOK SERIES LINEAR ALGEBRA

Singular Value Decomposition and Least Squares Solutions

Contributed by

G. H. Golub

C. Reinsch

Computer Science Department

Stanford University

HANDBOOK SERIES LINEAR ALGEBRA

SINGULAR VALUE DECOMPOSITION AND LEAST SQUARES SOLUTIONS

Contributed by

G. H. Golub[*] and C. Reinsch[+]

1. Theoretical Background

1.1 Introduction.

Let A be a real $m\times n$ matrix with $m \geq n$. It is well known (cf. [4]) that

$$A = U_{m\times n}\Sigma_{n\times n}V^T_{n\times n} \tag{1}$$

where $U^TU = I_n$, $VV^T = I_n$ and $\Sigma = \text{diag}(\sigma_1,\ldots,\sigma_n)$.

[*] The work of this author was in part supported by the National Science Foundation and Office of Naval Research.

[+] Case Western Reserve University, on leave from Technische Hochschule München.

The matrix U consists of n orthonormalized eigenvectors associated with the n largest eigenvalues of AA^T, and the matrix V consists of the orthornormalized eigenvectors of A^TA. The diagonal elements of Σ are the non-negative square roots of the eigenvalues of A^TA; they are called singular values. We shall assume that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0 .$$

Thus if rank $(A) = r$, $\sigma_{r+1} = \sigma_{r+2} = \cdots = \sigma_n = 0$. The decomposition (1) is called the singular value decomposition (SVD).

If the matrix U is not needed, it would appear that one could apply the usual diagonalization algorithms to the symmetric matrix A^TA which has to be formed explicitly. However, as in the case of linear least squares problems, the computation of A^TA involves unnecessary numerical inaccuracy. For example, let

$$A = \begin{bmatrix} 1 & 1 \\ \beta & 0 \\ 0 & \beta \end{bmatrix} ,$$

then $A^TA = \begin{bmatrix} 1+\beta^2 & 1 \\ 1 & 1+\beta^2 \end{bmatrix}$ so that

$$\sigma_1(A) = (2+\beta^2)^{1/2} , \quad \sigma_2(A) = |\beta| .$$

If $\beta^2 < \epsilon_0$, the machine precision, the computed A^TA has the form $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, and the best one may obtain from diagonalization is

$\tilde{\sigma}_1 = \sqrt{2}$, $\tilde{\sigma}_2 = 0$.

To compute the singular value decomposition of a given matrix A , Forsythe and Henrici [2], Hestenes [8], and Kogbetliantz [9] proposed methods based on plane rotations. Kublanovskaya [10] suggested a QR-type method. The program described below first uses Householder transformations to reduce A to bidiagonal form, and then the QR algorithm to find the eigenvalues of the bidiagonal matrix. The two phases properly combined produce the singular value decomposition of A .

1.2 Reduction to bidiagonal form.

It was shown in [6] how to construct two finite sequences of Householder transformations

$$P^{(k)} = I - 2x^{(k)}x^{(k)^T} \qquad (k = 1,2,\ldots,n)$$

and

$$Q^{(k)} = I - 2y^{(k)}y^{(k)^T} \qquad (k = 1,2,\ldots,n-2)$$

(where $x^{(k)^T}x^{(k)} = y^{(k)^T}y^{(k)} = 1$) such that

$$P^{(n)} \cdots P^{(1)} A Q^{(1)} \cdots Q^{(n-2)} =$$

$$= \left[\begin{array}{ccccccc} q_1 & e_2 & 0 & \cdot & \cdot & \cdot & 0 \\ & q_2 & e_3 & & & \bigcirc & \cdot \\ & & \cdot & \cdot & & & \cdot \\ & & & \cdot & \cdot & & \cdot \\ & \bigcirc & & & \cdot & \cdot & 0 \\ & & & & & \cdot & e_n \\ & & & & & & q_n \\ \hline & & & \bigcirc & & & \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \left.\right\} (m-n) \times n \end{array} = J^{(0)} \quad ,$$

an upper bidiagonal matrix. If we let $A^{(1)} = A$ and define

$$A^{(k+\frac{1}{2})} = P^{(k)} A^{(k)} \qquad (k = 1,2,\ldots,n)$$

$$A^{(k+1)} = A^{(k+\frac{1}{2})} Q^{(k)} \qquad (k = 1,2,\ldots,n-2)$$

then $P^{(k)}$ is determined such that

$$a_{ik}^{(k+\frac{1}{2})} = 0 \qquad (i = k+1,\ldots,m)$$

and $Q^{(k)}$ such that

$$a_{kj}^{(k+1)} = 0 \qquad (j = k+2,\ldots,n) \quad .$$

The singular values of $J^{(0)}$ are the same as those of A . Thus, if the singular value decomposition of

$$J^{(0)} = G \Sigma H^T$$

then

$$A = PG\Sigma H^T Q^T$$

so that $U = PG$, $V = QH$ with $P = P^{(1)} \ldots P^{(n)}$, $Q = Q^{(1)} \ldots Q^{(n-2)}$.

1.3 Singular value decomposition of the bidiagonal matrix.

By a variant of the QR algorithm, the matrix $J^{(0)}$ is iteratively diagonalized so that

$$J^{(0)} \to J^{(1)} \to \ldots \to \Sigma$$

where

$$J^{(i+1)} = S^{(i)^T} J^{(i)} T^{(i)} \quad ,$$

and $S^{(i)}$, $T^{(i)}$ are orthogonal. The matrices $T^{(i)}$ are chosen so that the sequence $M^{(i)} = J^{(i)^T} J^{(i)}$ converges to a diagonal matrix while the matrices $S^{(i)}$ are chosen so that all $J^{(i)}$ are of the bidiagonal form. In [7], another technique for deriving $\{S^{(i)}\}$ and $\{T^{(i)}\}$ is given but this is equivalent to the method described below.

For notational convenience, we drop the suffix and use the notation

$$J \equiv J^{(i)} \quad , \quad \bar{J} \equiv J^{(i+1)} \quad , \quad S \equiv S^{(i)} \quad , \quad T \equiv T^{(i)}$$

$$M \equiv J^T J \quad , \quad \bar{M} \equiv \bar{J}^T \bar{J} \quad .$$

The transition $J \to \bar{J}$ is achieved by application of Givens rotations to J alternately from the right and the left. Thus

$$\bar{J} = \underbrace{S_n^T S_{n-1}^T \ldots S_2^T}_{S^T} \, J \, \underbrace{T_2 T_3 \ldots T_n}_{T} \tag{2}$$

where

$$S_k = \begin{array}{cc}
\begin{matrix} \quad & & & & (k-1) & (k) & & & & \end{matrix} & \\
\begin{bmatrix}
1 & 0 & & & & & & & & \\
0 & \cdot & & & & & \bigcirc & & & \\
 & & \cdot & & & & & & & \\
 & & & \cdot & & & & & & \\
 & & & & 1 & & & & & \\
 & & & & & \cos\theta_k & -\sin\theta_k & & & \\
 & & & & & \sin\theta_k & \cos\theta_k & & & \\
 & & & & & & & 1 & & \\
 & \bigcirc & & & & & & & \cdot & \\
 & & & & & & & & & \cdot \\
 & & & & & & & & \cdot & 0 \\
 & & & & & & & & 0 & 1
\end{bmatrix} & \begin{matrix} \\ \\ \\ \\ \\ (k-1) \\ (k) \\ \\ \\ \\ \\ \\ \end{matrix}
\end{array}$$

and T_k is defined analogously to S_k with φ_k instead of θ_k.

Let the first angle, φ_2, be arbitrary while all the other angles are chosen so that $\bar{J}$ has the same form as J. Thus,

$$\left.\begin{array}{l}
T_2 \text{ annihilates nothing, generates an entry } (\bar{J})_{21}, \\
S_2^T \text{ annihilates } (\bar{J})_{21}, \text{ generates an entry } (\bar{J})_{13}, \\
T_3 \text{ annihilates } (\bar{J})_{13}, \text{ generates an entry } (\bar{J})_{32}, \\
\vdots \\
\text{and finally} \\
S_n^T \text{ annihilates } (\bar{J})_{n,n-1}, \text{ and generates nothing.}
\end{array}\right\} \qquad (3)$$

(See figure on next page.)

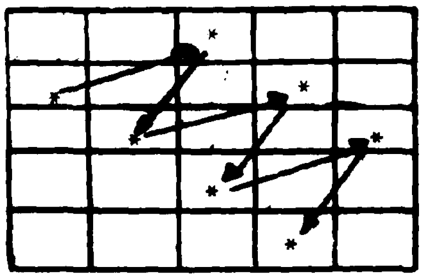

Figure 1

This process is frequently described as "chasing". Since $\bar{J} = S^TJT$,

$$\bar{M} = \bar{J}^T\bar{J} = T^TMT$$

and $\bar{M}$ is a tri-diagonal matrix just as M is. We show that the first angle, Θ_2 , which is still undetermined, can be chosen so that the transition $M \to \bar{M}$ is a QR transformation with a given shift s .

The usual QR algorithm with shifts is described as follows:

$$\left.\begin{aligned} (M-sI) &= T_sR_s \\ R_sT_s + sI &= \bar{M}_s \end{aligned}\right\} \tag{4}$$

where $T_s^TT_s = I$ and R_s is an upper triangular matrix. Thus $\bar{M}_s = T_s^TMT_s$. It has been shown by Francis [5] that it is not necessary to compute (4) explicitly but it is possible to perform the shift implicitly. Let T be for the moment an arbitrary matrix such that

$$\{T_s\}_{k,1} = \{T\}_{k,1} \qquad (k = 1,2,\ldots,n) \quad ,$$

(i.e., the elements of the first column of T_s are equal to the first column of T) and

$$T^TT = I \quad .$$

Then we have the following theorem (Francis):

i) $\tilde{M} = T^T M T$,

ii) $\tilde{M}$ is a tri-diagonal matrix,

iii) the sub-diagonal elements of $\tilde{M}$ are non-zero,

it follows that $\tilde{M} = D\tilde{M}_s D$ where D is a diagonal matrix whose diagonal elements are ± 1 .

Thus choosing T_2 in (3) such that its first column is proportional to that of $M-sI$, the same is true for the first column of the product $T = T_2 T_3 \dots T_n$ which therefore is identical to that of T_s . Hence, if the sub-diagonal of M does not contain any non-zero entry the conditions of the Francis theorem are fulfilled and T is therefore identical to T_s (up to a scaling of column ± 1). Thus the transition (2) is equivalent to the QR transformation of $J^T J$ with a given shift s .

The shift parameter s is determined by an eigenvalue of the lower 2x2 minor of M . Wilkinson [13] has shown that for this choice of s , the method converges globally and almost always cubically.

1.4 Test for convergence.

If $|e_n| \leq \delta$, a prescribed tolerance, then $|q_n|$ is accepted as a singular value, and the order of the matrix is dropped by one. If, however, $|e_k| \leq \delta$ for $k \neq n$, the matrix breaks into two, and the singular values of each block may be computed independently.

If $q_k = 0$, then at least one singular value must be equal to zero. In the absence of roundoff error, the matrix will break if a shift of zero is performed. Now, suppose at some stage

$$|q_k| \le \delta \quad .$$

At this stage an extra sequence of Givens rotations is applied from the left to J involving rows $(k, k+1)$, $(k, k+2)$, ..., (k, n) so that

$e_{k+1} = \{J\}_{k,k+1}$ is annihilated, but $\{J\}_{k,k+2}$, $\{J\}_{k+1,k}$ are generated,

$\{J\}_{k,k+2}$ is annihilated, but $\{J\}_{k,k+3}$, $\{J\}_{k+2,k}$ are generated,

$\vdots$

and finally

$\{J\}_{k,n}$ is annihilated, and $\{J\}_{n,k}$ is generated.

The matrix obtained thusly has the form

$$\bar{J} = \begin{array}{cc}
\begin{array}{ccccccccc} & & & & (k) & & & & \end{array} & \\
\left[\begin{array}{ccccccccc}
q_1 & e_2 & & & & & & O & \\
 & \vdots & \ddots & & & & & & \\
 & & \ddots & \ddots & & & & & \\
 & & & \ddots & e_k & & & & \\
 & & & & q_k & 0 & & & \\
 & & & & \delta_{k+1} & \bar{q}_{k+1} & \bar{e}_{k+2} & & \\
 & O & & & \vdots & & \ddots & \ddots & \\
 & & & & \vdots & & & \ddots & \bar{e}_n \\
 & & & & \delta_n & & & & \bar{q}_n
\end{array}\right] & \begin{array}{c} \\ \\ \\ \\ (k) \\ \\ \\ \\ \\ \end{array}
\end{array} \quad .$$

Note by orthogonality

$$\bar{q}_k^2 + \delta_{k+1}^2 + \dots + \delta_n^2 = q_k^2 \le \delta^2 \quad .$$

Thus choosing $\delta = \|J^{(0)}\|_\infty \epsilon_0$ (ϵ_0, the machine precision) ensures that all δ_k are less in magnitude than $\epsilon_0 \|J^{(0)}\|_\infty$. Elements of $\tilde{J}$ not greater than this are neglected. Hence $\tilde{J}$ breaks up into two parts which may be treated independently.

2. Applicability

There are a large number of applications of the singular value decomposition; an extensive list is given in [7]. Some of these are as follows:

2.1 Pseudoinverse (procedure SVD).

Let A be a real $m \times n$ matrix. An $n \times m$ matrix X is said to be the pseudoinverse of A if X satisfies the following four properties:

i) $AXA = A$

ii) $XAX = X$

iii) $(AX)^T = AX$

iv) $(XA)^T = XA$.

The unique solution is denoted by A^+ . It is easy to verify that if $A = U\Sigma V^T$, then $A^+ = V\Sigma^+ U^T$ where $\Sigma^+ = \text{diag}(\sigma_i^+)$ and

$$\sigma_i^+ = \begin{cases} 1/\sigma_i & \text{for } \sigma_i > 0 \\ 0 & \text{for } \sigma_i = 0 \end{cases} .$$

Thus the pseudoinverse may easily be computed from the output provided by the procedure SVD.

2.2 Solution of homogeneous equations (procedure SVD or procedure Minfit)

Let A be a matrix of rank r , and suppose we wish to solve

$$Ax_i = \Theta \qquad \text{for } i = r+1,\ldots,n$$

where Θ denotes the null vector.

Let

$$U = [u_1, u_2, \ldots, u_n] \text{ and } V = [v_1, v_2, \ldots, v_n] .$$

Then since $Au_i = \sigma_i v_i \quad (i = 1,2,\ldots,n)$,

$$Au_i = \theta \qquad \text{for} \quad i = r+1,\ldots,n$$

and $x_i = u_i$.

Here the procedure SVD or the procedure Minfit with $p = 0$ may be used for determining the solution. If the rank of A is known, then a modification of the algorithm of Businger and Golub [1] may be used.

2.3 Solutions of minimal length (procedure Minfit).

Let b be a given vector. Suppose we wish to determine a vector x so that

$$\|b - Ax\|_2 = \min \quad . \tag{5}$$

If the rank of A is less than n then there is no unique solution. Thus we require amongst all x which satisfy (5) that

$$\|\hat{x}\|_2 = \min.$$

and this solution is unique. It is easy to verify that

$$\hat{x} = A^+ b = V\Sigma^+ U^T b = V\Sigma^+ c \quad .$$

The procedure Minfit with $p > 0$ will yield the components for the solution to this problem.

2.4 A generalization of the least squares problem (procedure SVD)

Let A be a real $m \times n$ matrix of rank n and let b be a given vector. We wish to construct a vector x such that

$$(A + \Delta A)x = b + \Delta b$$

and

$$\Delta b^T \Delta b + K \text{ trace } (\Delta A^T \Delta A) = \min. \tag{6}$$

Here $K \geq 0$ is a given weight and the standard problem is obtained for $K \to \infty$. It can be shown that the solution is given by

$$x = (A^T A - \mu I)^{-1} A^T b$$

where the non-negative constant μ is determined as the smallest root of

$$b^T b - \mu K = b^T A (A^T A - \mu I)^{-1} A^T b \quad . \tag{7}$$

The minimum of (6) is given by μK. Using the decomposition $A = U \Sigma V^T$ and $c = U^T b$, equation (7) becomes

$$b^T b - \mu K = c^T \Sigma (\Sigma^2 - \mu I)^{-1} \Sigma c \quad . \tag{8}$$

A combination of bisection and Newton iteration may be used to determine μ in the interval $0 \leq \mu < \sigma_n^2$.

It is also possible to determine μ as a solution to a singular value problem using a technique used by Forsythe and Golub [7]. Consider the identity

$$\det \begin{bmatrix} X & Y \\ Z & W \end{bmatrix} = \det (X) \det (W - Z X^{-1} Y)$$

which is valid for any partitioned matrix with X and W square and $\det(X) \neq 0$. Thus (7) is equivalent to the determinantal equation

$$\det \begin{bmatrix} A^TA-\mu I & A^Tb \\ b^TA & b^Tb-\mu K \end{bmatrix} = 0 \quad .$$

A short manipulation shows that $\sqrt{\mu}$ is the smallest singular value of

$$G \equiv (A \, , \, \frac{1}{\sqrt{K}} b) \quad .$$

Once μ is determined, the solution x can be computed from the SVD of A. Thus

$$x = V(\Sigma - \mu\Sigma^{-1})^{-1} c \quad .$$

3. Formal Parameter List

3.1 Input to procedure SVD.

m	number of rows of A , $m \geq n$.
n	number of columns of A .
withu	true if U is desired, false otherwise.
withv	true if V is desired, false otherwise.
eps	a constant used in the test for convergence (see Section 5, (iii)); should not be smaller than the machine precision ϵ_0 , i.e., the smallest number for which $1+\epsilon_0 > 1$ in computer arithmetic.
tol	a machine dependent constant which should be set equal to β/ϵ_0 where β is the smallest positive number representable in the computer, see [11].
a[1:m,1:n]	represents the matrix A to be decomposed.

Output of procedure SVD.

q[1:n]	a vector holding the singular values of A , they are non-negative but not neccessarily ordered in decreasing sequence.
u[1:m,1:n]	represents the matrix U with orthonormalized columns, (if withu is true, otherwise u is used as a working storage).
v[1:n,1:n]	represents the orthogonal matrix V (if withv is true, otherwise v is not used).

3.2 **Input to procedure *Minfit*.**

m	number of rows of A .
n	number of columns of A .
p	number of columns of B , $p \geq 0$.
eps	same as for procedure *SVD*.
tol	same as for procedure *SVD*.
ab[1:max(m,n),1:n+p]	ab[i,j] represents $a_{i,j}$, $1 \leq i \leq m$, $1 \leq j \leq n$,
	ab[i,n+j] represents $b_{i,j}$, $1 \leq i \leq m$, $1 \leq j \leq p$.

Output to procedure *Minfit*.

ab[1:max(m,n),1:n+p]	ab[i,j] represents $v_{i,j}$, $1 \leq i \leq n$, $1 \leq j \leq n$,
	ab[i,n+j] represents $c_{i,j}$, $1 \leq i \leq \max(m,n)$, $1 \leq j \leq p$,
	viz. $C = U^T B$.
q[1:n]	same as for procedure *SVD*.

4. Algol Programs.

```
procedure  SVD (m,n,withu,withv,eps,tol) data: (a) result: (q,u,v);
   value    m,n,withu,withv,eps,tol;
   integer  m,n;
   Boolean  withu,withv;
   real     eps,tol;
   array    a,q,u,v;
comment
   Computation of the singular values and complete orthogonal decomposition
of a real rectangular matrix A,
            A = U diag(q) V^T,  U^T U = V^T V = I,
where the arrays a[1:m,1:n], u[1:m,1:n], v[1:n,1:n], q[1:n] represent
A, U, V, q respectively.  The actual parameters corresponding to a, u, v
may all be identical unless withu = withv = true.  In this case, the actual
parameters corresponding to u and v must differ.  m ≥ n is assumed;
begin
   integer  i,j,k,l,l1;
   real     c,f,g,h,s,x,y,z;
   array    e[1:n];
   for i:=1 step 1 until m do
   for j:=1 step 1 until n do u[i,j]:=a[i,j];
comment Householder's reduction to bidiagonal form;
```

```
g:=x:=0;
for i:=1 step 1 until n do
begin
   e[i]:=g; s:=0; l:=i+1;
   for j:=i step 1 until m do s:=s + u[j,i]↑2;
   if s < tol then g:=0 else
   begin
      f:=u[i,i]; g:= if f < 0 then sqrt(s) else -sqrt(s);
      h:=f*g-s; u[i,i]:=f-g;
      for j:=l step 1 until n do
      begin
         s:=0;
         for k:=i step 1 until m do s:=s + u[k,i]*u[k,j];
         f:=s/h;
         for k:=i step 1 until m do u[k,j]:=u[k,j] + f*u[k,i]
      end j
   end s;
   q[i]:=g; s:=0;
   for j:=l step 1 until n do s:=s + u[i,j]↑2;
   if s < tol then g:=0 else
   begin
      f:=u[i,i+1]; g:= if f < 0 then sqrt(s) else -sqrt(s);
      h:=f*g-s; u[i,i+1]:=f-g;
      for j:=l step 1 until n do e[j]:=u[i,j]/h;
      for j:=l step 1 until m do
```

```
          begin
             s:=0;
             for k:=l step 1 until n do s:=s + u[j,k]*u[i,k];
             for k:=l step 1 until n do u[j,k]:=u[j,k] + s*e[k]
          end j
       end s;
       y:=abs(q[i])+abs(e[i]); if y > x then x:=y
    end i;
comment  accumulation of right-hand transformations;
    if withv then for i:=n step -1 until 1 do
    begin
       if g ≠ 0 then
       begin
          h:=u[i,i+1]*g;
          for j:=l step 1 until n do v[j,i]:=u[i,j]/h;
          for j:=l step 1 until n do
          begin
             s:=0;
             for k:=l step 1 until n do s:=s + u[i,k]*v[k,j];
             for k:=l step 1 until n do v[k,j]:=v[k,j] + s*v[k,i]
          end j
       end g;
       for j:=l step 1 until n do v[i,j]:=v[j,i]:=0;
       v[i,i]:=1; g:=e[i]; l:=i
    end i;
```

```
comment  accumulation of left-hand transformations;
   if withu then for i:=n step -1 until 1 do
   begin
      l:=i+1; g:=q[i];
      for j:=l step 1 until n do u[i,j]:=0;
      if g ≠ 0 then
      begin
         h:=u[i,i]*g;
         for j:=l step 1 until n do
         begin
            s:=0;
            for k:=l step 1 until m do s:=s + u[k,i]*u[k,j];
            f:=s/h;
            for k:=i step 1 until m do u[k,j]:=u[k,j] + f*u[k,i]
         end j;
         for j:=i step 1 until m do u[j,i]:=u[j,i]/g
      end g
      else for j:=i step 1 until m do u[j,i]:=0;
      u[i,i]:=u[i,i] + 1
   end i;
comment  diagonalization of the bidiagonal form;
   eps:=eps*x;
   for k:=n step -1 until 1 do
   begin
      test f splitting:
         for l:=k step -1 until 1 do
```

```
            begin
               if abs(e[l]) ≤ eps then goto test f convergence;
               if abs(q[l-1]) ≤ eps then goto cancellation
            end l;
comment  cancellation of e[l] if l > 1;
   cancellation:
         c:=0; s:=1; l1:=l-1;
         for i:=l step 1 until k do
         begin
            f:=s*e[i]; e[i]:=c*e[i];
            if abs(f) ≤ eps then goto test f convergence;
            g:=q[i]; h:=q[i]:=sqrt(f*f + g*g); c:=g/h; s:=-f/h;
            if withu then for j:=1 step 1 until m do
            begin
               y:=u[j,l1]; z:=u[j,i];
               u[j,l1]:= y*c + z*s; u[j,i]:=-y*s + z*c
            end j
         end i;
   test f convergence:
         z:=q[k]; if l = k then goto convergence;
comment  shift from bottom 2*2 minor;
         x:=q[l]; y:=q[k-1]; g:=e[k-1]; h:=e[k];
         f:=((y-z)*(y+z) + (g-h)*(g+h)) / (2*h*y); g:=sqrt(f*f + 1);
         f:=((x-z)*(x+z) + h*(y/(if f < 0 then f-g else f+g) - h)) / x;
```

```
comment  next QR transformation;
      c:=s:=1;
      for i:=l+1 step 1 until k do
      begin
         g:=e[i]; y:=q[i]; h:=s*g; g:=c*g;
         e[i-1]:=z:=sqrt (f*f + h*h); c:=f/z; s:=h/z;
         f:=x*c + g*s; g:=-x*s + g*c; h:=y*s; y:=y*c;
         if withv then for j:=1 step 1 until n do
         begin
            x:=v[j,i-1]; z:=v[j,i];
            v[j,i-1]:=x*c + z*s; v[j,i]:=-x*s + z*c
         end j;
         q[i-1]:=z:=sqrt(f*f + h*h); c:=f/z; s:=h/z;
         f:=c*g + s*y; x:=-s*g + c*y;
         if withu then for j:=1 step 1 until m do
         begin
            y:=u[j,i-1]; z:=u[j,i];
            u[j,i-1]:=y*c + z*s; u[j,i]:=-y*s + z*c
         end j
      end i;
      e[l]:=0; e[k]:=f; q[k]:=x; goto test f splitting;
   convergence:
      if z < 0 then
      begin comment  q[k] is made non-negative;
         q[k]:=-z;
         if withv then for j:=1 step 1 until n do v[j,k]:=-v[j,k]
      end z
   end k
 end SVD;
```

```
procedure Minfit (m,n,p,eps,tol) trans: (ab) result: (q);
   value   m,n,p,eps,tol;
   integer m,n,p;
   real    eps,tol;
   array   ab,q;
comment
   Computation of the matrices diag(q), V, and C such that for given real
m*n matrix A and m*p matrix B
        $U_C^T A V = diag(q)$   and   $U_C^T B = C$   with orthogonal matrices $U_C$ and V.
The singular values and the matrices V and C may be used to determine $\bar{X}$
minimizing  (1)  $||AX-B||_F$  and  (2)  $||X||_F$  with the solution
        $\bar{X}$ = V * Pseudo-inverse of diag(q) * C.
The procedure can also be used to determine the complete solution of an
underdetermined linear system, i.e., rank(A) = m < n.
   The array q[1:n] represents the matrix diag(q), A and B together are to
be given as the first m rows of the array ab[1:max(m,n),1:n+p].  V is
returned in the first n rows and columns of ab while C is returned in the
last p columns of ab (if p > 0);
begin
   integer  i,j,k,l,l1,n1,np;
   real     c,f,g,h,s,x,y,z;
   array    e[1:n];
comment Householder's reduction to bidiagonal form;
```

```
g:=x:=0; np:=n+p;
for i:=1 step 1 until n do
begin
   e[i]:=g; s:=0; l:=i+1;
   for j:=i step 1 until m do s:=s + ab[j,i]↑2;
   if s < tol then g:=0 else
   begin
      f:=ab[i,i]; g:= if f < 0 then sqrt(s) else -sqrt(s);
      h:=f*g-s; ab[i,i]:=f-g;
      for j:=l step 1 until np do
      begin
         s:=0;
         for k:=i step 1 until m do s:=s + ab[k,i]*ab[k,j];
         f:=s/h;
         for k:=i step 1 until m do ab[k,j]:=ab[k,j] + f*ab[k,i]
      end j
   end s;
   q[i]:=g; s:=0;
   if i ≤ m then for j:=l step 1 until n do s:=s + ab[i,j]↑2;
   if s < tol then g:=0 else
   begin
      f:=ab[i,i+1]; g:= if f < 0 then sqrt(s) else -sqrt(s);
      h:=f*g-s; ab[i,i+1]:=f-g;
      for j:=l step 1 until n do e[j]:=ab[i,j]/h;
      for j:=l step 1 until m do
```

```
        begin
           s:=0;
           for k:=l step 1 until n do s:=s + ab[j,k]*ab[i,k];
           for k:=l step 1 until n do ab[j,k]:=ab[j,k] + s*e[k]
        end j
     end s;
     y:=abs(q[i]) + abs(e[i]); if y > x then x:=y
  end i;
comment accumulation of right-hand transformations;
  for i:=n step -1 until 1 do
  begin
     if g ≠ 0 then
     begin
        h:=ab[i,i+1]*g;
        for j:=l step 1 until n do ab[j,i]:=ab[i,j]/h;
        for j:=l step 1 until n do
        begin
           s:=0;
           for k:=l step 1 until n do s:=s + ab[i,k]*ab[k,j];
           for k:=l step 1 until n do ab[k,j]:=ab[k,j] + s*ab[k,i]
        end j
     end g;
     for j:=l step 1 until n do ab[i,j]:=ab[j,i]:=0;
     ab[i,i]:=1; g:=e[i]; l:=i
  end i;
```

```
   eps:=eps*x; n1:=n+1;
   for i:= n1 step 1 until m do
   for j:=n1 step 1 until np do ab[i,j]:=0;
comment diagonalization of the bidiagonal form;
   for k:=n step -1 until 1 do
   begin
      test f splitting:
         for l:=k step -1 until 1 do
         begin
            if abs(e[l]) ≤ eps then goto test f convergence;
            if abs(q[l-1]) ≤ eps then goto cancellation
         end l;
comment cancellation of e[l] if l > 1;
      cancellation:
         c:=0; s:=1; l1:=l-1;
         for i:=l step 1 until k do
         begin
            f:=s*e[i]; e[i]:=c*e[i];
            if abs(f) ≤ eps then goto test f convergence;
            g:=q[i]; q[i]:=h:=sqrt(f*f + g*g); c:=g/h; s:=-f/h;
            for j:=n1 step 1 until np do
            begin
               y:=ab[l1,j]; z:=ab[i,j];
               ab[l1,j]:=c*y + s*z; ab[i,j]:=-s*y + c*z
            end j
         end i;
```

```
    test f convergence:
        z:=q[k]; if l = k then goto convergence;
comment shift from bottom 2*2 minor;
        x:=q[l]; y:=q[k-1]; g:=e[k-1]; h:=e[k];
        f:=((y-z)*(y+z) + (g-h)*(g+h)) / (2*h*y); g:=sqrt(f*f + 1);
        f:=((x-z)*(x+z) + h*(y/(if f < 0 then f-g else f+g) - h)) / x;
comment next QR transformation;
        c:=s:=1;
        for i:=l+1 step 1 until k do
        begin
           g:=e[i]; y:=q[i]; h:=s*g; g:=c*g;
           e[i-1]:=z:=sqrt(f*f + h*h); c:=f/z; s:=h/z;
           f:=x*c + g*s; g:=-x*s + g*c; h:=y*s; y:=y*c;
           for j:=1 step 1 until n do
           begin
              x:=ab[j,i-1]; z:=ab[j,i];
              ab[j,i-1]:=x*c + z*s; ab[j,i]:=-x*s + z*c
           end j;
           q[i-1]:=z:=sqrt(f*f + h*h); c:=f/z; s:=h/z;
           f:=c*g + s*y; x:=-s*g + c*y;
           for j:=n1 step 1 until np do
           begin
              y:= ab[i-1,j]; z:=ab[i,j];
              ab[i-1,j]:=c*y + s*z; ab[i,j]:=-s*y + c*z
           end j
        end i;
        e[l]:=0; e[k]:=f; q[k]:=x; goto test f splitting;
        convergence:
          if z < 0 then
          begin comment  q[k] is made non-negative;
            q[k]:=-z;
            for j:=1 step 1 until n do ab[j,k]:=-ab[j,k]
          end z
      end k
    end Minfit;
```

5. Organizational and Notational Details

(i) The matrix U consists of the first n columns of an orthogonal matrix U_c. The following modification of the procedure SVD would produce U_c instead of U: After

comment accumulation of left-hand transformations;

insert a statement

```
if withu then for i:=n+1 step 1 until m do
begin
   for j:=n+1 step 1 until m do u[i,j]:=0;
   u[i,i]:=1
end i;
```

Moreover, replace n by m in the fourth and eighth line after that, i.e., write twice **for** j:=1 **step** 1 **until** m **do**.

(ii) $m \geq n$ is assumed for procedure SVD. This is no restriction; if $m < n$, store A^T, i.e., use an array at [1:n,1:m] where at[i,j] represents $a_{j,i}$ and call SVD(n,m,withv,withu,eps,tol,at,q,v,u) producing the $m*m$ matrix U and the $n*m$ matrix V. There is no restriction on the values of m and n for the procedure Minfit.

(iii) In the iterative part of the procedures an element of $J^{(i)}$ is considered to be negligible and is consequently replaced by zero if it is not larger in magnitude than ϵx where ϵ is the given tolerance and

$$x = \max_{1 \leq i \leq n} (|q_i| + |e_i|) \; .$$

The largest singular value σ_1 is bounded by $x/\sqrt{2} \leq \sigma_1 \leq x\sqrt{2}$.

(iv) A program organization was chosen which allows to save storage locations. To this end the actual parameters corresponding to $\underset{\sim}{a}$ and $\underset{\sim}{u}$ may be identical. In this event the original information stored in $\underset{\sim}{a}$ is overwritten by information on the reduction. This, in turn, is overwritten by $\underset{\sim}{u}$ if the latter is desired. Likewise, the actual parameters corresponding to $\underset{\sim}{a}$ and $\underset{\sim}{v}$ may agree. Then $\underset{\sim}{v}$ is stored in the upper part of $\underset{\sim}{a}$ if it is desired, otherwise $\underset{\sim}{a}$ is not changed. Finally, all three parameters $\underset{\sim}{a}$, $\underset{\sim}{u}$, and $\underset{\sim}{v}$ may be identical unless withu = withv = true.

This special feature, however, increases the number of multiplications needed to form U roughly by a factor $\underset{\sim}{m}/\underset{\sim}{n}$.

(v) Shifts are evaluated in a way as to reduce the danger of overflow or underflow of exponents.

(vi) The singular values as delivered in the array q are not necessarily ordered. Any sorting of them should be accompanied by the corresponding sorting of the columns of U and V , and of the rows of C .

(vii) The formal parameter list may be completed by the addition of a limit for the number of iterations to be performed, and by the addition of a failure exit to be taken if no convergence is reached after the specified number of iterations (f.e., 30 per singular value).

6. Numerical Properties

The stability of the Householder transformations has been demonstrated by Wilkinson [12]. In addition, he has shown that in the absence of roundoff the QR algorithm has global convergence and asymptotically is almost always cubically convergent.

The numerical experiments indicate that the average number of complete QR iterations on the bidiagonal matrix is usually less than two per singular value. Extra consideration must be given to the implicit shift technique which fails for a split matrix. The difficulties arise when there are small q_k's or e_k's . Using the techniques of Section 1.4, there can not be numerical instability since stable orthogonal transformations are used but under special circumstances there may be a slow down in the rate of convergence.

7. Test Results

Tests were carried out on the UNIVAC 1108 Computer of the Andrew R. Jennings Computing Center of Case Western Reserve University. Floating point numbers are represented by a normalized 27 bit mantissa and a 7 bit exponent to the radix 2, whence $\underset{\sim}{\text{eps}} = 1.5_{10}{-8}$, $\underset{\sim}{\text{tol}} = {}_{10}{-31}$. In the following, computed values are marked by a tilde and $m(A)$ denotes $\max|a_{i,j}|$.

First example:

$$A = \begin{bmatrix} 22 & 10 & 2 & 3 & 7 \\ 14 & 7 & 10 & 0 & 8 \\ -1 & 13 & -1 & -11 & 3 \\ -3 & -2 & 13 & -2 & 4 \\ 9 & 8 & 1 & -2 & 4 \\ 9 & 1 & -7 & 5 & -1 \\ 2 & -6 & 6 & 5 & 1 \\ 4 & 5 & 0 & -2 & 2 \end{bmatrix} \qquad B = \begin{bmatrix} -1 & 1 & 0 \\ 2 & -1 & 1 \\ 1 & 10 & 11 \\ 4 & 0 & 4 \\ 0 & -6 & -6 \\ -3 & 6 & 3 \\ 1 & 11 & 12 \\ 0 & -5 & -5 \end{bmatrix}$$

$$\sigma_1 = \sqrt{1248}\ , \quad \sigma_2 = 20\ , \quad \sigma_3 = \sqrt{384}\ , \quad \sigma_4 = \sigma_5 = 0\ .$$

The homogeneous system $Ax = \Theta$ has two linearly independent solutions. Six QR transformations were necessary to drop all off-diagonal elements below the internal tolerance $46.4_{10}{-8}$. Table 1 gives the singular values in the sequence as computed by procedures $\underset{\sim}{\text{SVD}}$ and $\underset{\sim}{\text{Minfit}}$. The accuracy of the achieved decomposition is characterized by

$$m(A - \tilde{U}\tilde{\Sigma}\tilde{V}^T) = 238_{10}{-8}\ , \quad m(\tilde{U}^T\tilde{U} - I) = 8.1_{10}{-8}\ , \quad m(\tilde{V}^T\tilde{V} - I) = 3.3_{10}{-8}\ .$$

__Table 1__

$\tilde{\sigma}_k$	$\sigma_k - \tilde{\sigma}_k$	
$0.96_{10}{-7}$	-9.6	
19.595916	191	
19.999999	143	$* \; 10^{-8}$
$1.97_{10}{-7}$	-19.7	
35.327038	518	

The computed solutions of the homogeneous system are given by the first and fourth column of the matrix $\tilde{V}$ (Table 2).

__Table 2__

$\tilde{v}_1$	$\tilde{v}_4$	$v_1 - \tilde{v}_1$	$v_4 - \tilde{v}_4$	
-0.4190 9545	0	-1.5	0 (Def.)	
0.4405 0912	0.4185 4806	1.7	0.6	
-0.0520 0457	0.3487 9006	1.2	-1.3	$* \; 10^{-8}$
0.6760 5915	0.2441 5305	1.0	0.3	
0.4129 7730	-0.8022 1713	1.3	-0.8	

Procedure Minfit was used to compute the solutions of the minimization problem of Section 2.3 corresponding to the three right-hand sides as given by the columns of the matrix B . Table 3 lists the exact solutions and the results obtained when the first and fourth value in Table 1 are replaced by zero.

Table 3

x_1	x_2	x_3	$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$
-1/12	0	-1/12	-0.0833 3333	$0.17_{10}-8$	-0.0833 3333
0	0	0	$-0.58_{10}-8$	$-1.09_{10}-8$	$-1.11_{10}-8$
1/4	0	1/4	0.2500 0002	$1.55_{10}-8$	0.2500 0003
-1/12	0	-1/12	-0.0833 3332	$0.74_{10}-8$	-0.0833 3332
1/12	0	1/12	0.0833 3334	$0.33_{10}-8$	0.0833 3334
	Residual				
0	8/5	8/5			

A second example is the $20*21$ matrix with entries

$$a_{i,j} = \begin{cases} 0 & \text{if } i > j \\ 21-i & \text{if } i = j \\ -1 & \text{if } i < j \end{cases} \qquad \begin{array}{l} 1 \le i \le 20 \\ 1 \le j \le 21 \end{array}$$

which has orthogonal rows and singular values $\sigma_{21-k} = \sqrt{k(k+1)}$, $k = 0,\ldots,20$. Theoretically, the Householder reduction should produce a matrix $J^{(o)}$ with diagonal $-20,0,\ldots,0$ and super-diagonal $-\sqrt{20},\sigma_2,\ldots,\sigma_{20}$. Under the influence of rounding errors a totally different matrix results. However, within working accuracy its singular values agree with those of the original matrix. Convergence is reached after 32 QR transformations and the $\tilde{\sigma}_k$, $k = 1,\ldots,20$ are correct within several units in the last digit, $\tilde{\sigma}_{21} = 1.61_{10}-11$.

A third example is obtained if the diagonal of the foregoing example is changed to

$$a_{i,i} = 1 \quad , \quad 1 \leq i \leq 20 \; .$$

This matrix has a cluster of singular values, σ_{10} to σ_{19} lying between 1.5 and 1.6 , $\sigma_{20} = \sqrt{2}$, $\sigma_{21} = 0$. Clusters, in general, have a tendency to reduce the number of required iterations; in this example, 26 iterations were necessary for convergence. $\tilde{\sigma}_{21} = 1.49_{10}{-8}$ is found in eighteenth position and the corresponding column of $\tilde{V}$ differs from the unique solution of the homogeneous system by less than $3.4_{10}{-8}$ in any component.

A second test was made by Dr. Peter Businger on the CDC 6600.

Acknowledgement: The authors wish to thank Dr. Peter Businger of Bell Telephone Laboratories for his stimulating comments.

References

[1] P. Businger and G. Golub, "Linear least squares solutions by Householder transformations", Num. Math., 7 (1965), 269-276.

[2] G. E. Forsythe and P. Henrici, "The cyclic Jacobi method for computing the principal values of a complex matrix", Proc. Amer. Math. Soc., 94 (1960), 1-23.

[3] G. E.Forsythe and G. Golub, "On the stationary values of a second-degree polynomial on the unit sphere", J. Soc. Indust. Appl. Math., 13 (1965), 1050-1068.

[4] G. E. Forsythe and C. B. Moler, Computer Solution of Linear Algebraic Systems, Prentice-Hall, Englewood Cliffs, New Jersey, 1967.

[5] J. Francis, "The QR transformation. A unitary analogue to the LR transformation", Comput. J., 4 (1961, 1962), 265-271.

[6] G. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix", J. SIAM. Numer. Anal., Ser. B, 2 (1965), 205-224.

[7] G. Golub, "Least squares, singular values, and matrix approximations", Aplikace Matematiky, 13 (1968), 44-51.

[8] M. R. Hestenes, "Inversion of matrices by biorthogonalization and related results", J. Soc. Indust. Appl. Math., 6 (1958), 51-90.

[9] E. G. Kogbetliantz, "Solution of linear equations by diagonalization of coefficients matrix", Quart. Appl. Math., 13 (1955), 123-132.

[10] V. N. Kublanovskaja, "Some algorithms for the solution of the complete problem of eigenvalues", V. Vyčisl. Mat. i. Mat. Fiz., 1 (1961), 555-570.

[11] R. S. Martin, C. Reinsch, and J. H. Wilkinson, "Householder's tridiagonalization of a symmetric matrix", Num. Math. 11 (1968), 181-195.

[12] J. Wilkinson, "Error analysis of transformations based on the use of matrices of the form $I-2ww^H$", Error in Digital Computation, Vol. II, L. B. Rall, ed., John Wiley and Sons, Inc., New York, 1965, 77-101.

[13] J. Wilkinson, "Global convergence of QR algorithm", Proceedings of IFIP Congress, 1968.

Keywords

Singular Value Decomposition

Least Squares

Pseudoinverse

QR algorithm

Unclassified
Security Classification

AD 687718

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Computer Science Department Stanford University Stanford, California 94305	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP ---

3. REPORT TITLE
HANDBOOK SERIES LINEAR ALGEBRA
SINGULAR VALUE DECOMPOSITION AND LEAST SQUARES SOLUTIONS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Manuscript for Publication (Technical Report)

5. AUTHOR(S) (First name, middle initial, last name)
G. H. Golub and C. Reinsch

6. REPORT DATE	7a. TOTAL NO. OF PAGES 38	7b. NO. OF REFS 13

8a. CONTRACT OR GRANT NO. N00014-67-A-0112-0029	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO. NR 044-211	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) none
d.	

10. DISTRIBUTION STATEMENT
Releasable without limitations on dissemination.

11. SUPPLEMENTARY NOTES ----	12. SPONSORING MILITARY ACTIVITY Office of Naval Research

13. ABSTRACT

Two Algol procedures are given which are useful in linear least squares problems. The first procedure computes the singular value decomposition by first reducing the rectangular matrix A to a bidiagonal matrix, and then computing the singular values of the bidiagonal matrix by a variant of the QR algorithm. The second procedure yields the components for the linear least squares solution when it is desirable to determine a vector $\underset{\sim}{x}$ for which $\|Ax-b\|_2 = \min$.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

Unclassified
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Singular Value Decomposition						
Least Squares						
Pseudoinverse						
QR algorithm						