

CS 73

AD 662883

28

LEAST SQUARES, SINGULAR VALUES AND MATRIX APPROXIMATIONS

BY

①

GENE H. GOLUB

AN ALGOL PROCEDURE FOR COMPUTING THE
SINGULAR VALUE DECOMPOSITION

BY

PETER BUSINGER

TECHNICAL REPORT NO. CS73
JULY 31, 1967

DTIC
F
DEC 22 1967
E

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY



1967
for public release and sale; its
distribution is unlimited.

Reproduced by the
CLEARINGHOUSE
for Federal Scientific & Technical
Information Springfield Va. 22151

Least Squares, Singular Values
and Matrix Approximations *

Gene H. Golub

0. Let A be a real, $m \times n$ matrix (for notational convenience we assume that $m \geq n$). It is well known (cf. [6]) that

$$(0.1) \quad A = U\Sigma V^T, \quad ,$$

where $U U^T = I_m$, $V V^T = I_n$ and

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_n \\ \dots & & \\ 0 & & \end{pmatrix} \}^{(m-n) \times n}.$$

The matrix U consists of the orthonormalized eigenvectors of $A A^T$, and the matrix V consists of the orthonormalized eigenvectors of $A^T A$. The diagonal elements of Σ are the non-negative square roots of the eigenvalues of $A^T A$; they are called the singular values or principal values of A . Throughout this note, we assume

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0 \quad .$$

Thus if $\text{rank}(A) = r$, $\sigma_{r+1} = \sigma_{r+2} = \dots = \sigma_n = 0$. The decomposition (0.1) is called the singular value decomposition.

* To be presented at the conference on "Basic Problems of Numerical Mathematics" to be held in Libbie, Czechoslovakia, September 11, 1967 through September 15, 1967. This work was in part supported by NSF and ONR.

1. The singular value decomposition plays an important role in a number of least squares problems, and we will illustrate this with some examples.

Throughout this discussion, we use the euclidean or Frobenius norm of a matrix, viz. $\|A\| = (\sum_{i,j} |a_{ij}|^2)^{1/2}$.

A) Let U_n be the set of all $n \times n$ orthogonal matrices. For an arbitrary $n \times n$ real matrix A , determine $Q \in U_n$ such that

$$\|A - Q\| \leq \|A - X\| \quad \text{for any } X \in U_n.$$

It has been shown by Fan and Hoffman [2] that if $A = U \Sigma V^T$, then $Q = U V^T$.

B) An important generalization of problem A occurs in factor analysis.

For arbitrary $n \times n$ real matrices A and B , determine $Q \in U_n$ such that

$$\|A - BQ\| \leq \|A - BX\| \quad \text{for any } X \in U_n.$$

It has been shown by Green [5] and by Schönemann [9] that if

$$B^T A = U \Sigma V^T, \quad \text{then } Q = U V^T.$$

C) Let $\mathcal{M}_{m,n}^{(k)}$ be the set of all $m \times n$ matrices of rank k . Assume $A \in \mathcal{M}_{m,n}^{(r)}$. Determine $B \in \mathcal{M}_{m,n}^{(k)}$ ($k \leq r$) such that

$$\|A - B\| \leq \|A - X\| \quad \text{for all } X \in \mathcal{M}_{m,n}^{(k)}.$$

It has been shown by Eckart and Young [1] that if

$$(1.1) \quad A = U \Sigma V^T, \quad \text{then } B = U \Omega_k V^T,$$

where

$$(1.2) \quad \Omega_k = \begin{pmatrix} \sigma_1 & & & 0 & & \\ & \sigma_2 & & & & \\ & & \ddots & & & \\ 0 & & & & \sigma_k & \\ \hline & & & & & 0 \end{pmatrix}.$$

Note that

$$(1.3) \quad \|A - B\| = \|\Sigma - \Omega_k\| = (\sigma_{k+1}^2 + \dots + \sigma_r^2)^{1/2} \quad .$$

D) An $n \times m$ matrix X is said to be the pseudo-inverse of an $m \times n$ matrix A if X satisfies the following four properties:

- i) $AXA = A$
- ii) $XAX = X$
- iii) $(AX)^T = AX$
- iv) $(XA)^T = XA \quad .$

We denote the pseudo-inverse by A^+ . We wish to determine A^+ numerically.

It can be shown [8] that A^+ can always be determined and is unique.

It is easy to verify that

$$(1.4) \quad A^+ = V \Lambda U^T$$

where

$$\Lambda = \begin{pmatrix} \frac{1}{\sigma_1} & & & 0 & & \\ & \frac{1}{\sigma_2} & & & & \\ & & \ddots & & & \\ 0 & & & & \frac{1}{\sigma_r} & \\ \hline & & & & & 0 \end{pmatrix}_{n \times m}$$

In recent years there have been a number of algorithms proposed for computing the pseudo-inverse of a matrix. These algorithms usually depend upon a knowledge of the rank of the matrix or upon some suitably chosen parameter. For example in the latter case, if one uses (1.4) to compute the pseudo-inverse, then after one has computed the singular value decomposition numerically it is necessary to determine which of the singular values are zero by testing against some tolerance.

Alternatively, suppose we know that the given matrix A can be represented as

$$A = B + \delta B ,$$

where δB is a matrix of perturbations and

$$\|\delta B\| \leq \eta .$$

Now, we wish to construct a matrix \hat{B} such that

$$\|A - \hat{B}\| \leq \eta$$

and

$$\text{rank}(\hat{B}) = \text{minimum} .$$

This can be accomplished with the aid of the solution to problem (C). Let

$$B_k = U_k V^T \quad \text{as in equation (1.2).}$$

Then using (1.3),

$$\hat{B} = B_p$$

if

$$(\sigma_{p+1}^2 + \sigma_{p+2}^2 + \dots + \sigma_n^2)^{1/2} \leq \eta$$

and

$$(\sigma_p^2 + \sigma_{p+1}^2 + \dots + \sigma_n^2)^{1/2} > \eta$$

Since $\text{rank}(\hat{B}) = p$ by construction,

$$\hat{B}^+ = V \Omega^+ U^T$$

Thus, we take \hat{B}^+ as our approximation to A^+ .

E) Let A be a given matrix, and let \tilde{b} be a known vector.

Determine a vector \tilde{x} such that for

$$\begin{cases} \|\tilde{b} - A\tilde{x}\|_2 = \min. \\ \text{and } \|\tilde{x}\|_2 = \min., \end{cases}$$

where $\|y\|_2 = [\sum y_i^2]^{1/2}$ for any vector y . It is easy to verify that $\tilde{x} = A^+ \tilde{b}$.

A norm is said to be unitarily invariant if $\|AU\| = \|VA\| = \|A\|$ when $U^*U = I$ and $V^*V = I$. Fan and Hoffman [2] have shown that the solution to problem (A) is the same for all unitarily invariant norms and Mirsky [7] has proved a similar result for the solution to problem (C).

2. In [4] it was shown by Golub and Kahan that it is possible to construct a sequence of orthogonal matrices $\{P^{(k)}\}_{k=1}^n, \{Q^{(k)}\}_{k=1}^{n-1}$ via Householder transformation so that

$$P^{(n)} P^{(n-1)} \dots P^{(1)} A Q^{(1)} Q^{(2)} \dots Q^{(n-1)} = P^T A Q = J$$

and J is an $m \times n$ bi-diagonal matrix of the form

$$J = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \cdot & 0 \\ \alpha_2 & \beta_2 & \cdot & 0 \\ \vdots & \vdots & \ddots & \beta_{n-1} \\ \hline 0 & & & \alpha_n \\ \hline 0 & & & \end{bmatrix}_{(m-n) \times n}.$$

The singular values of J are the same as those of A . Thus if the singular value decomposition of

$$J = X\Sigma Y^T,$$

then

$$A = P\Sigma Y Q^T$$

so that $U = PX$, $V = QY$.

A number of algorithms were proposed in [4] for computing the singular value decomposition of J . We now describe a new algorithm, based on the QR algorithm of Francis [3], for computing the singular value decomposition of J .

Let

$$K = \begin{bmatrix} 0 & \alpha_1 & & & & \\ \alpha_1 & 0 & \beta_1 & & & \\ \beta_1 & 0 & \alpha_2 & & & \\ \alpha_2 & \cdot & \cdot & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ 0 & & & \cdot & \cdot & \alpha_n \\ \hline \alpha_n & 0 & & & & \end{bmatrix}_{2n \times 2n}.$$

It can be shown [4] that K is a symmetric, tri-diagonal matrix whose eigenvalues are \pm singular values of J . One of the most effective methods of computing the eigenvalues of a tri-diagonal matrix is the QR algorithm of Francis, which proceeds as follows:

Begin with the given matrix $K = K_0$. Compute the factorization

$$K_0 = M_0 R_0$$

where $M_0^T M_0 = I$ and R_0 is an upper triangular matrix, and then multiply the matrices in reverse order so that

$$K_1 = R_0 M_0 = M_0^T K M_0 .$$

Now one treats K_1 in the same fashion as the matrix K_0 , and a sequence of matrices is obtained by continuing ad infinitum. Thus

$$K_i = M_i R_i \quad \text{and}$$

$$K_{i+1} = R_i M_i = M_{i+1} R_{i+1} ,$$

so that

$$\begin{aligned} K_{i+1} &= M_i^T K M_i \\ &= M_{i-1}^T \dots M_0^T K M_0 \dots M_i . \end{aligned}$$

The method has the advantage that K_i remains tri-diagonal throughout the computation.

For suitably chosen shift parameters s_i , we can accelerate the convergence of the QR method by computing

$$(2.1) \quad \left\{ \begin{array}{l} (K_i - s_i I) = M_i R_i \\ R_i M_i + s_i I = K_{i+1} \end{array} \right. .$$

Unfortunately, the shift parameter s_i may destroy the zeroes on the diagonal of K .

Since the eigenvalues of K always occur in pairs, it would seem more appropriate to compute the QR decomposition of

$$(K_i - s_i I)(K_i + s_i I) = K_i^2 - s_i^2 I$$

so that

$$M_i R_i = K_i^2 - s_i^2 I \quad .$$

It has been shown by Francis that it is not necessary to compute (2.1) explicitly but it is possible to perform the shift implicitly. Let

$$\{N_i\}_{k,1} = \{M_i\}_{k,1} \quad k = 1, 2, \dots, 2n \quad .$$

(i.e., the elements of the first column of N are equal to the elements of the first column of M) and

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

Then if

- i) $T_{i+1} = N_i^T K_i N_i \quad ,$
- ii) T_{i+1} is a tri-diagonal matrix,
- iii) K_i is non-singular,
- iv) the sub-diagonal elements of T_{i+1} are positive,

it follows that $T_{i+1} = K_{i+1} \quad .$

The calculation proceeds quite simply. Dropping the iteration counter i , let

Then $\cos \theta_1$ is chosen so that

$$\{z_1(\kappa^2 - s^2 I)\}_{k,1} = 0 \quad \text{for} \quad k = 2, 3, \dots, 2n .$$

Then the matrix

$$z_1 K z_1 = \begin{bmatrix} 0 & \alpha'_1 & 0 & d_1 \\ \alpha'_1 & 0 & \beta'_1 & \\ 0 & \beta'_1 & \ddots & \alpha'_2 \\ d_1 & \alpha'_2 & \ddots & \beta_2 \\ & \beta_2 & \ddots & \ddots \\ & & \ddots & \ddots & \alpha_n \\ & & & \alpha_n & 0 \end{bmatrix};$$

and

$$T = Z_{2n-2} \dots Z_1 K Z_1 \dots Z_{2n-2} ,$$

where Z_2, \dots, Z_{2n-2} are constructed so that T is tri-diagonal. The product of all the orthogonal transformations which gives the singular values yields the matrix of orthogonal eigenvectors of K . For ease of notation let us write

$$\gamma_{2j-1} = \alpha_j \quad j = 1, 2, \dots, n$$

$$\gamma_{2j} = \beta_j \quad j = 1, 2, \dots, n-1 .$$

Then explicitly, the calculation goes as follows: Dropping the iteration counter i ,

$$\gamma_0 = \gamma_1^2 - s^2 \quad , \quad d_0 = \gamma_1 \gamma_2 .$$

For $j = 0, 1, \dots, 2n-3$,

$$r_j = (\gamma_j^2 + d_j^2)^{1/2}$$

$$\sin \theta_j = d_j / r_j \quad , \quad \cos \theta_j = \gamma_j / r_j \quad ,$$

$$\gamma_j = r_j$$

$$\bar{\gamma}_{j+1} = \tilde{\gamma}_{j+1} \cos \theta_j + \hat{\gamma}_{j+2} \sin \theta_j \quad ,$$

$$\tilde{\gamma}_{j+2} = \tilde{\gamma}_{j+1} \sin \theta_j - \hat{\gamma}_{j+2} \cos \theta_j \quad ,$$

$$\hat{\gamma}_{j+3} = -\gamma_{j+3} \cos \theta_j \quad ,$$

$$d_{j+1} = \gamma_{j+3} \sin \theta_j \quad .$$

In the actual computation, no additional storage is required for

$$\{\bar{\gamma}_j, \tilde{\gamma}_j, \hat{\gamma}_j\}$$

since they may overwrite $\{\gamma_j\}$. Furthermore, only one element of storage need be reserved for $\{d_j\}$. When $|\gamma_{2n-2}|$ is sufficiently small, $|\gamma_{2n-1}|$ is taken as a singular value and n is replaced by $n-1$.

Now let us define

where $\cos \theta_p$ is defined as above. It has been pointed out to the author by J. H. Wilkinson that the above iteration is equivalent to forming

$$\hat{J} = w_{2n-2} \dots w_4 w_2 J w_1 w_3 \dots w_{2n-3}$$

where J is again a bi-diagonal matrix. Thus,

$$x = \prod_i (w_2^{(i)} w_4^{(i)} \dots w_{2n-2}^{(i)})$$

$$Y = \prod_i (w_1^{(i)} w_3^{(i)} \dots w_{2n-3}^{(i)})$$

An ALGOL procedure embodying these techniques will soon be published by Dr. Peter Businger and the author.

References

An extensive list of references on singular values is given in [4].

- [1] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank", Psychometrika, 1 (1936), pp. 211-218.
- [2] K. Fan and A. Hoffman, "Some metric inequalities in the space of matrices", Proc. Amer. Math. Soc., 6 (1955), pp. 111-116.
- [3] J. Francis, "The QR transformation. A unitary analogue to the LR transformation", Comput. J., 4 (1961, 1962), pp. 265-271.
- [4] G. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix", J. SIAM Numer. Anal. Ser. B, 2 (1965), pp. 205-224.
- [5] B. Green, "The orthogonal approximation of an oblique structure in factor analysis", Psychometrika, 17 (1952), pp. 429-440.
- [6] C. Lanczos, Linear Differential Operators, Van Nostrand, London, 1961, Chap. 3.
- [7] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms", Quart. J. Math. Oxford (2), 11 (1960), pp. 50-59.
- [8] R. Penrose, "A generalized inverse for matrices", Proc. Cambridge Philos. Soc., 51 (1955), pp. 406-413.
- [9] P. Schönemann, "A generalized solution of the orthogonal procrustes problem", Psychometrika, 31 (1966), pp. 1-10.

**AN ALGOL PROCEDURE FOR COMPUTING THE
SINGULAR VALUE DECOMPOSITION**

by

PETER BUSINGER *

*** Computing Center, University of Texas, Austin, Texas.**

BLANK PAGE

```
procedure singular values decomposition
  (a, m, n, u desired, vt desired, eta) results: (sigma, u, vt) ;
value m, n, u desired, vt desired, eta ;
real array a, sigma, u, vt ;
integer m, n ;
boolean u desired, vt desired ;
real eta ;
comment Householder's and the QR method are used to find all singular
  values sigma[i], (i=1, 2, ..., n) of the given matrix a[1:m, 1:n],
  (m≥n). The orthogonal matrices u[1:m, 1:m] and vt[1:n, 1:n] which
  effect the singular values decomposition a=u sigma vt are computed
  individually depending on whether u desired or vt desired. The input
  parameter eta is the relative machine precision ;
begin
procedure Householder bidiagonalization
  (a, m, n, u desired, vt desired) results: (alpha, beta, u, vt) ;
value m, n, u desired, vt desired ;
real array a, alpha, beta, u, vt ;
integer m, n ;
boolean u desired, vt desired ;
comment Householder transformations applied in turn on the left and
  the right reduce the given matrix a[1:m, 1:n], (m≥n) to upper bi-
  diagonal form J. The diagonal elements of J are returned as alpha[i],
  (i=1, 2, ..., n), the superdiagonal elements as beta[i], (i=1, 2,
  ..., n-1), beta[n]=0. The orthogonal matrices u[1:m, 1:m] and
  vt[1:n, 1:n] which effect the decomposition a=u J vt are computed
  individually depending on whether u desired or vt desired ;
```

```
begin
  real procedure inner product (i, m, n, a, b, c) ;
  value m, n, c ; real a, b, c ; integer i, m, n ;
  begin
    for i := m step 1 until n do c=c+a×b ; inner product=c
  end inner product ;
  real s, b ;
  integer i, j, k ;
  if u desired then
    for i:=1 step 1 until m do
    begin
      u[i,i]:=1.0 ;
      for j:=i+1 step 1 until m do u[i,j]:=u[j,i]:=0.0
    end i ;
  if vt desired then
    for i:=1 step 1 until n do
    begin
      vt[i,i]:=1.0 ;
      for j:=i+1 step 1 until n do vt[i,j]:=vt[j,i]:=0.0
    end i ;
  for k:=1 step 1 until n do
  begin
    s:=inner product(i, k, m, a[i,k], a[i,k], 0.0) ;
    alpha[k]:=if a[k,k]<0.0 then sqrt(s) else -sqrt(s) ;
    if s<0.0 then
      begin comment transformation on the left ;
      b:=s-a[k,k]×alpha[k] ;
```

```
a[k,k]:=a[k,k]-alpha[k] ;
for j:=k+1 step 1 until n do
begin
  s:=inner product(i, k, m, a[i,k], a[i,j], 0.0)/b ;
  for i:=k step 1 until m do
    a[i,j]:=a[i,j]-a[i,k]*s
  end j ;
  if u desired then
    for i:=1 step 1 until m do
    begin
      s:=inner product(j, k, m, u[i,j], a[j,k], 0.0)/b ;
      for j:=k step 1 until m do
        u[i,j]:=u[i,j]-s*a[j,k]
    end i
  end transformation on the left ;
  if k<n-2 then
  begin
    s:=inner product(j, k+1, n, a[k,j], a[k,i], 0.0) ;
    beta[k]:=if a[k,k+1]<0.0 then sqrt(s) else -sqrt(s) ;
    if s<0.0 then
      begin comment transformation on the right ;
        b:=s-a[k,k+1]*beta[k] ;
        a[k,k+1]:=a[k,k+1]-beta[k] ;
        for i:=k+1 step 1 until m do
        begin
          s:=inner product(j, k+1, n, a[k,j], a[i,j], 0.0)/b ;
          for j:=k+1 step 1 until n do
            a[i,j]:=a[i,j]-a[k,j]*s
        end i ;
      end comment transformation on the right ;
    end if s<0.0 then
  end if k<n-2 then
end if k<n-2 then
```

```
if vt desired then
  for j:=1 step 1 until n do
    begin
      s:=inner product(i, k+1, n, a[k,i], vt[i,j], 0.0)/b ;
      for i:=k+1 step 1 until n do
        vt[i,j]:=vt[i,j]-a[i,k]*s
    end j
  end transformation on the right
end k from 1 to n-2
  else beta[k]:=if k=n then 0.0 else a[k,n]
end k
end Householder bi-diagonalization ;
procedure QR diagonalization
  (gamma, m, n, u desired, vt desired, eta) result: (sigma)
  transients: (u, vt) ;
  value m, n, u desired, vt desired, eta ;
  real array gamma, sigma, u, vt ;
  integer m, n ;
  Boolean u desired, vt desired ;
  comment The QR algorithm diagonalizes the given symmetric tridiagonal
  matrix T of order 2n by 2n whose diagonal elements are zero and
  whose super- and subdiagonal elements are gamma[i], (i=1, 2, ...,
  2n-1), gamma[0]=gamma[2n]=0. If u desired then the odd numbered
  rotations of the QR algorithm are also applied to u[1:m, 1:m] from
  the right. If vt desired then the even numbered rotations are also
  applied to vt[1:n, 1:n] from the left. The input parameter eta is
  the relative machine precision. The nonnegative eigenvalues of T
```

```
are returned as sigma[i], (i=1, 2, ..., n) ;

begin
  real kappa, d, r, sinphi, cosphi, g0, g1, g2, g3, epsilon, rho ;
  integer i, j, k, s, s0, t, t0, t2 ;
  s:=s0:=t0:=0 ; t:=2*n ;
  kappa:=g1:=abs(gamma[1]) ;
  for i:=2 step 1 until t do
    begin comment find the infinity norm of the tridiagonal matrix T ;
    g2:=abs(gamma[i]) ; d:=g1+g2 ; if d>kappa then kappa:=d ;
    g1:=g2
  end i ;
  epsilon:=eta*kappa ;

inspect:
  comment scan for lower block limit t ;
  gamma[s]:=gamma[t]:=0.0 ;
  for i:=t-2 while abs(gamma[i])<=epsilon do
    begin comment pick up computed value ;
    t2:=t-2 ; sigma[t2]:=abs(gamma[t-1]) ;
    if gamma[t-1]<0.0  $\wedge$  vt desired then
      for j:=1 step 1 until n do vt[t2,j]:=-vt[t2,j] ;
    t:=i ; gamma[t]:=0.0 ;
    if t=0 then go to return
  end ;
  s:=t-4 ; comment scan for upper block limit s ;
  for i:=s-2 while abs(gamma[s])>epsilon do s:=i ;
  comment did block limits s, t change ;
```

```
if s≠s0\&t≠t0 then
begin
zero shift:
    gamma[s]:=gamma[s+1] : d:=gamma[s+2] : go to QR sweep
end zero shift ;
comment does matrix break ;
if abs(gamma[s+1]×gamma[s+2])≤epsilon then go to zero shift ;
for i:=s+1 step 2 until t-1 do
    if abs(gamma[i])≤epsilon then go to zero shift ;
comment did bottom value settle down ;
if abs(abs(gamma[t-1])-rho)>0.1×abs(gamma[t-1]) then
    go to zero shift ;
comment determine the origin shift kappa ;
g0:=gamma[t-1]↑2+gamma[t-2]↑2+gamma[t-3]↑2 ;
g1:=gamma[t-1]↑2×gamma[t-2]↑2 ;
g2:=0.5×(g0+sqrt(g0↑2-4.0×g1)) ;
g3:=g1/g2 ;
kappa:=if abs(gamma[t-1]↑2-g2)<abs(gamma[t-1]↑2-g3) then g2 else g3
gamma[s]:=gamma[s+1]↑2-kappa ; d:=gamma[s+1]×gamma[s+2] ;
QR sweep:
comment save previous block limits and bottom element ;
s0:=s ; t0:=t ; rho:=abs(gamma[t-1]) ;
for i:=s step 1 until t-3 do
begin
comment does matrix break ;
if d=0.0 then go to inspect ;
```

```
g0:=gamma[1] ; g1:=gamma[1+1] ;
g2:=gamma[1+2] ; g3:= gamma[1+3] ;
r:=sqrt(g0↑2+d↑2) ;
sinphi:=d/r ; cosphi:=g0/r ;
gamma[1]:=r ;
gamma[1+1]:=g1*cosphi+g2*sinphi ;
gamma[1+2]:=g1*sinphi-g2*cosphi ;
gamma[1+3]:=-g3*cosphi ;
d:=g3*sinphi ;
if u desired  $\vee$  vt desired then
begin
k:=1÷2 ;
if i=2×k  $\wedge$  vt desired then
for j:=1 step 1 until n do
begin
g1:=vt[k+1,j] ; g2:=vt[k+2,j] ;
vt[k+1,j]:=g1*cosphi+g2*sinphi ;
vt[k+2,j]:=g1*sinphi-g2*cosphi
end j ;
if i≠2×k  $\wedge$  u desired then
for j:=1 step 1 until m do
begin
g1:=u[j,k+1] ; g2:=u[j,k+2] ;
u[j,k+1]:=g1*cosphi+g2*sinphi ;
u[j,k+2]:=g1*sinphi-g2*cosphi
end j
```

```
    end if u desired or vt desired
    end i ;
    go to inspect ;
return;
end QR diagonalization ;
real array alpha, beta[1:n], gamma[0:2xn] ;
integer i, j ;
Householder bidiagonalization
    (a, m, n, u desired, vt desired, alpha, beta, u, vt) ;
for i:=1 step 1 until n do
    begin
        gamma[2x1-1]:=alpha[1] ; gamma[2x1]:=beta[1]
    end i ;
    gamma[0]:=gamma[2xn]:=0.0 ;
QR diagonalization
    (gamma, m, n, u desired, vt desired, eta, sigma, u, vt)
end singular values decomposition
```