nag_real_svd (f02wec)

1. Purpose

nag_real_svd (f02wec) returns all, or part, of the singular value decomposition of a general real matrix.

2. Specification

```
#include <nag.h>
#include <nagf02.h>

void nag_real_svd(Integer m, Integer n, double a[], Integer tda, Integer ncolb,
     double b[], Integer tdb, Boolean wantq, double q[], Integer tdq,
     double sv[], Boolean wantp, double pt[], Integer tdpt, Integer *iter,
     double e[], Integer *failinfo, NagError *fail)
```

3. Description

The m by n matrix A is factorized as

$$A = QDP^T$$

where

$$\begin{array}{ll} D = \begin{pmatrix} S \\ 0 \end{pmatrix} & m > n \\ D = S, & m = n \\ D = (S\ 0) & m < n. \end{array}$$

Q is an m by m orthogonal matrix, P is an n by n orthogonal matrix and S is a $\min(m,n)$ by $\min(m,n)$ diagonal matrix with non-negative diagonal elements, $sv_1, sv_2, \ldots, sv_{\min(m,n)}$, ordered such that

$$sv_1 \geq sv_2 \geq \ldots \geq sv_{\min(m,n)} \geq 0.$$

The first $\min(m,n)$ columns of Q are the left-hand singular vectors of A, the diagonal elements of S are the singular values of A and the first $\min(m,n)$ columns of P are the right-hand singular vectors of A.

Either or both of the left-hand and right-hand singular vectors of A may be requested and the matrix C given by

$$C = Q^T B$$

where B is an m by *ncolb* given matrix, may also be requested.

The function obtains the singular value decomposition by first reducing A to upper triangular form by means of Householder transformations, from the left when $m \geq n$ and from the right when $m < n$. The upper triangular form is then reduced to bidiagonal form by Givens plane rotations and finally the QR algorithm is used to obtain the singular value decomposition of the bidiagonal form.

Good background descriptions to the singular value decomposition are given in Dongarra *et al*(1979), Hammarling (1985) and Wilkinson (1978). Note that this function is not based on the LINPACK routine SSVDC.

Note that if K is any orthogonal diagonal matrix such that

$$KK^T = I, \text{(so that } K \text{ has elements} + 1 \text{ or } -1 \text{ on the diagonal)}$$

then

$$A = (QK)D(PK)^T$$

is also a singular value decomposition of A.

4. Parameters

m

Input: the number of rows, m, of the matrix A.
Constraint: $\mathbf{m} \geq 0$.
When $\mathbf{m} = 0$ then an immediate return is effected.

n

Input: the number of columns, n, of the matrix A.
Constraint: $\mathbf{n} \geq 0$.
When $\mathbf{n} = 0$ then an immediate return is effected.

a[m][tda]

Input: the leading m by n part of the array **a** must contain the matrix A whose singular value decomposition is required.
Output: if $\mathbf{m} \geq \mathbf{n}$ and $\mathbf{wantq} = \mathbf{TRUE}$, then the leading m by n part of **a** will contain the first n columns of the orthogonal matrix Q.
If $\mathbf{m} < \mathbf{n}$ and $\mathbf{wantp} = \mathbf{TRUE}$, then the leading m by n part of **a** will contain the first m rows of the orthogonal matrix P^T.
If $\mathbf{m} \geq \mathbf{n}$ and $\mathbf{wantq} = \mathbf{FALSE}$ and $\mathbf{wantp} = \mathbf{TRUE}$, then the leading n by n part of **a** will contain the first n rows of the orthogonal matrix P^T.
Otherwise the contents of the leading m by n part of **a** are indeterminate.

tda

Input: the second dimension of the array **a** as declared in the function from which nag_real_svd is called.
Constraint: $\mathbf{tda} \geq \mathbf{n}$.

ncolb

Input: *ncolb*, the number of columns of the matrix B. When $\mathbf{ncolb} = 0$ the array **b** is not referenced.
Constraint: $\mathbf{ncolb} \geq 0$.

b[m][tdb]

Input: if $\mathbf{ncolb} > 0$, the leading m by *ncolb* part of the array **b** must contain the matrix to be transformed. If $\mathbf{ncolb} = 0$ the array **b** is not referenced and may be set to the null pointer, i.e., (double $*$)0.
Output: **b** is overwritten by the m by *ncolb* matrix $Q^T B$.

tdb

Input: the second dimension of the array **b** as declared in the function from which nag_real_svd is called.
Constraint: if $\mathbf{ncolb} > 0$ then $\mathbf{tdb} \geq \mathbf{ncolb}$.

wantq

Input: **wantq** must be **TRUE**, if the left-hand singular vectors are required. If $\mathbf{wantq} = \mathbf{FALSE}$, then the array **q** is not referenced.

q[m][tdq]

Output: if $\mathbf{m} < \mathbf{n}$ and $\mathbf{wantq} = \mathbf{TRUE}$, the leading m by m part of the array **q** will contain the orthogonal matrix Q. Otherwise the array **q** is not referenced and may be set to the null pointer, i.e., (double $*$)0.

tdq

Input: the second dimension of the array **q** as declared in the function from which nag_real_svd is called.
Constraint: if $\mathbf{m} < \mathbf{n}$ and $\mathbf{wantq} = \mathbf{TRUE}$, $\mathbf{tdq} \geq \mathbf{m}$.

sv[min(m,n)]

Output: the min(**m**,**n**) diagonal elements of the matrix S.

wantp

Input: **wantp** must be **TRUE** if the right-hand singular vectors are required. If $\mathbf{wantp} = \mathbf{FALSE}$, then the array **pt** is not referenced.

pt[n][tdpt]

Output: if $\mathbf{m} \geq \mathbf{n}$ and **wantq** and **wantp** are **TRUE**, the leading n by n part of the array **pt** will contain the orthogonal matrix P^T. Otherwise the array **pt** is not referenced and may be set to the null pointer, i.e., (double $*$)0.

tdpt

Input: the second dimension of the array **pt** as declared in the function from which nag_real_svd is called.
Constraint: if $\mathbf{m} \geq \mathbf{n}$ and **wantq** and **wantp** are **TRUE**, $\mathbf{tdpt} \geq \mathbf{n}$.

iter

Output: the total number of iterations taken by the QR algorithm.

e[min(m,n)-1]

Output: if the error **NE_QR_NOT_CONV** occurs the array **e** contains the super diagonal elements of matrix E in the factorisation of A according to $A = QEP^T$. See Section 5 for further details.

failinfo

Output: if the error **NE_QR_NOT_CONV** occurs **failinfo** contains the number of singular values which may not have been found correctly. See Section 5 for details.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_INT_ARG_LT

On entry, **m** must not be less than 0: $\mathbf{m} = \langle value \rangle$.
On entry, **n** must not be less than 0: $\mathbf{n} = \langle value \rangle$.
On entry, **ncolb** must not be less than 0: $\mathbf{ncolb} = \langle value \rangle$.

NE_2_INT_ARG_LT

On entry, $\mathbf{tda} = \langle value \rangle$ while $\mathbf{n} = \langle value \rangle$. These parameters must satisfy $\mathbf{tda} \geq \mathbf{n}$.
On entry, $\mathbf{tdb} = \langle value \rangle$ while $\mathbf{ncolb} = \langle value \rangle$. These parameters must satisfy $\mathbf{tdb} \geq \mathbf{ncolb}$.

NE_TDQ_LT_M

On entry, $\mathbf{tdq} = \langle value \rangle$ while $\mathbf{m} = \langle value \rangle$. When **wantq** is **TRUE** and $\mathbf{m} < \mathbf{n}$ then relationship $\mathbf{tdq} \geq \mathbf{m}$ must be satisfied.

NE_TDP_LT_N

On entry, $\mathbf{tdpt} = \langle value \rangle$ while $\mathbf{n} = \langle value \rangle$. When **wantq** and **wantp** are **TRUE** and $\mathbf{m} \geq \mathbf{n}$ then relationship $\mathbf{tdpt} \geq \mathbf{n}$ must be satisfied.

NE_QR_NOT_CONV

The QR algorithm has failed to converge in $\langle value \rangle$ iterations. Singular values 1,2,. . .,**failinfo** may not have been found correctly and the remaining singular values may not be the smallest. The matrix A will nevertheless have been factorized as $A = QEP^T$, where the leading $min(m,n)$ by $min(m,n)$ part of E is a bidiagonal matrix with **sv**[0], **sv**[1], . . ., **sv**[min(**m**,**n**-1)] as the diagonal elements and **e**[0], **e**[1], . . ., **e**[min(**m**,**n**-2)] as the superdiagonal elements.
This failure is not likely to occur.

NE_ALLOC_FAIL

Memory allocation failed.

6. Further Comments

6.1. Accuracy

The computed factors Q, D and P satisfy the relation

$$QDP^T = A + E$$

where $\|E\| \leq c\epsilon\|A\|$, ϵ being the ***machine precision***, c is a modest function of m and n and $\|.\|$ denotes the spectral (two) norm. Note that $\|A\| = sv_1$.

6.2. References

Dongarra J J, Moler C B, Bunch J R and Stewart G W (1979) *LINPACK Users' Guide* SIAM, Philadelphia.

Hammarling S (1985) The Singular Value Decomposition in Multivariate Statistics *ACM Signum Newsletter* **20** (3) 2–25.

Wilkinson J H (1978) Singular-value Decomposition – Basic Aspects *Numerical Software – Needs and Availability* D A H Jacobs (ed) Academic Press, London.

7. See Also

None.

8. Example

For this function two examples are presented, in Sections 8.1 and 8.2. In the example programs distributed to sites, there is a single example program for nag_real_svd, with a main function:

```
/* nag_real_svd(f02wec) Example Program
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagf02.h>

#define EX1_MMAX 20
#define EX1_NMAX 10

#define EX2_MMAX 10
#define EX2_NMAX 20

static void ex1(), ex2();

main()
{

  Vprintf("f02wec Example Program Results\n");
  Vscanf(" %*[^\n]");  /* Skip heading in data file */
  ex1();
  ex2();
  exit(EXIT_SUCCESS);
}
```

The code to solve the two example problems is given in the functions ex1 and ex2, in Sections 8.1.1 and 8.2.1 respectively.

8.1. Example 1

To find the singular value decomposition of the 5 by 3 matrix

$$A = \begin{pmatrix} 2.0 & 2.5 & 2.5 \\ 2.0 & 2.5 & 2.5 \\ 1.6 & -0.4 & 2.8 \\ 2.0 & -0.5 & 0.5 \\ 1.2 & -0.3 & -2.9 \end{pmatrix}$$

together with the vector $Q^T b$ for the vector

$$b = \begin{pmatrix} 1.1 \\ 0.9 \\ 0.6 \\ 0.0 \\ -0.8 \end{pmatrix}.$$

8.1.1. Program Text

```
static void ex1()
{
  Integer tda = EX1_NMAX;
  Integer tdpt = EX1_NMAX;

  double a[EX1_MMAX][EX1_NMAX], b[EX1_MMAX], e[EX1_NMAX-1];
  double pt[EX1_NMAX][EX1_NMAX], sv[EX1_NMAX], dummy[1];
  Integer i, j, m, n, iter, failinfo;
  Boolean wantp, wantq;
  static NagError fail;

  Vprintf("Example 1\n");
  Vscanf(" %*[^\n]");  /* Skip Example 1 heading */
  Vscanf(" %*[^\n]");

  Vscanf("%ld%ld", &m, &n);
  if (m > EX1_MMAX || n > EX1_NMAX)
    {
      Vprintf("m or n is out of range.\n");
      Vprintf("m = %2ld, n = %2ld\n", m, n);
    }
  else
    {
      Vscanf(" %*[^\n]");
      for (i = 0; i < m; ++i)
        for (j = 0; j < n; ++j)
          Vscanf("%lf", &a[i][j]);
      Vscanf(" %*[^\n]");
      for (i = 0; i < m; ++i)
        Vscanf("%lf", &b[i]);

      /* Find the SVD of A. */
      wantq = TRUE;
      wantp = TRUE;
      fail.print = TRUE;
      f02wec(m, n, (double *)a, tda, (Integer)1, b, (Integer)1, wantq,
             dummy, (Integer)1, sv, wantp, (double *)pt, tdpt, &iter,
             e, &failinfo, &fail);
      if (fail.code != NE_NOERROR) exit(EXIT_FAILURE);

      Vprintf("Singular value decomposition of A\n\n");
      Vprintf("Singular values\n");
      for (i = 0; i < n; ++i)
        Vprintf(" %8.4f", sv[i]);
      Vprintf("\n\n");
      Vprintf("Left-hand singular vectors, by column\n");
      for (i = 0; i < m; ++i)
        {
          for (j = 0; j < n; ++j)
            Vprintf(" %8.4f", a[i][j]);
          Vprintf("\n");
        }
      Vprintf("\n");
      Vprintf("Right-hand singular vectors, by column\n");
      for (i = 0; i < n; ++i)
        {
          for (j = 0; j < n; ++j)
            Vprintf(" %8.4f", pt[j][i]);
          Vprintf("\n");
        }
      Vprintf("\n");
      Vprintf("Vector Q'*B\n");
      for (i = 0; i < m; ++i)
        Vprintf(" %8.4f", b[i]);
      Vprintf("\n\n");
    }
}
```

8.1.2. Program Data

```
f02wec Example Program Data

Example 1
Values of m and n
  5     3

Matrix A
  2.0   2.5   2.5
  2.0   2.5   2.5
  1.6  -0.4   2.8
  2.0  -0.5   0.5
  1.2  -0.3  -2.9

Vector B
  1.1   0.9   0.6   0.0  -0.8
```

8.1.3. Program Results

```
f02wec Example Program Results
Example 1
Singular value decomposition of A

Singular values
   6.5616   3.0000   2.4384

Left-hand singular vectors, by column
   0.6011  -0.1961  -0.3165
   0.6011  -0.1961  -0.3165
   0.4166   0.1569   0.6941
   0.1688  -0.3922   0.5636
  -0.2742  -0.8629   0.0139

Right-hand singular vectors, by column
   0.4694  -0.7845   0.4054
   0.4324  -0.1961  -0.8801
   0.7699   0.5883   0.2471

Vector Q'*B
   1.6716   0.3922  -0.2276  -0.1000  -0.1000
```

8.2. Example 2

To find the singular value decomposition of the 3 by 5 matrix

$$A = \begin{pmatrix} 2.0 & 2.0 & 1.6 & 2.0 & 1.2 \\ 2.5 & 2.5 & -0.4 & -0.5 & -0.3 \\ 2.5 & 2.5 & -2.8 & 0.5 & -2.9 \end{pmatrix}.$$

8.2.1. Program Text

```
static void ex2()
{
  Integer tda = EX2_NMAX;
  Integer tdq = EX2_MMAX;

  double a[EX2_MMAX][EX2_NMAX], e[EX2_NMAX-1];
  double q[EX2_MMAX][EX2_MMAX], sv[EX2_MMAX], dummy[1];
  Integer i, j, m, n, iter, ncolb, failinfo;
  Boolean wantp, wantq;
  static NagError fail;

  Vprintf("\nExample 2\n");
  Vscanf(" %*[^\n]");  /* Skip Example 2 heading */
  Vscanf(" %*[^\n]");

  Vscanf("%ld%ld", &m, &n);
  if (m > EX2_MMAX || n > EX2_NMAX)
    {
```

```
        Vprintf("m or n is out of range.\n");
        Vprintf("m = %2ld, n = %2ld\n", m, n);
      }
    else
      {
        Vscanf(" %*[^\n]");
        for (i = 0; i < m; ++i)
          for (j = 0; j < n; ++j)
            Vscanf("%lf", &a[i][j]);

        /* Find the SVD of A. */
        wantq = TRUE;
        wantp = TRUE;
        ncolb = 0;
        fail.print = TRUE;
        f02wec(m, n, (double *)a, tda, ncolb, dummy, (Integer)1, wantq,
               (double *)q, tdq, sv, wantp, dummy, (Integer)1, &iter,
               e, &failinfo, &fail);
        if (fail.code != NE_NOERROR) exit(EXIT_FAILURE);

        Vprintf("Singular value decomposition of A\n\n\n");
        Vprintf("Singular values\n\n");
        for (i = 0; i < m; ++i)
          Vprintf(" %8.4f", sv[i]);
        Vprintf("\n\n");
        Vprintf("Left-hand singular vectors, by column\n\n");
        for (i = 0; i < m; ++i)
          {
            for (j = 0; j < m; ++j)
              Vprintf(" %8.4f", q[i][j]);
            Vprintf("\n");
          }
        Vprintf("Right-hand singular vectors, by column\n\n");
        for (i = 0; i < n; ++i)
          {
            for (j = 0; j < m; ++j)
              Vprintf(" %8.4f", a[j][i]);
            Vprintf("\n");
          }
      }
  }
```

8.2.2. Program Data

```
Example 2
Values of m and n
  3    5

Matrix A
  2.0   2.0   1.6   2.0   1.2
  2.5   2.5  -0.4  -0.5  -0.3
  2.5   2.5   2.8   0.5  -2.9
```

8.2.3. Program Results

```
Example 2
Singular value decomposition of A


Singular values

   6.5616   3.0000   2.4384

Left-hand singular vectors, by column

  -0.4694   0.7845  -0.4054
  -0.4324   0.1961   0.8801
  -0.7699  -0.5883  -0.2471
```

```
Right-hand singular vectors, by column

  -0.6011   0.1961   0.3165
  -0.6011   0.1961   0.3165
  -0.4166  -0.1569  -0.6941
  -0.1688   0.3922  -0.5636
   0.2742   0.8629  -0.0139
```