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

$$D = \begin{pmatrix} S \\ 0 \end{pmatrix} \text{ if } m > n,$n
$$D = S, \text{ if } m = n,$n
$$D = (S 0) \text{ if } m < n.$$n

$Q$ 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, $sv_1, sv_2, \ldots, sv_{\text{min}(m,n)}$, ordered such that

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

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

$$C = Q^TB$$

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

$$KK^T = I,$$n

(so that $K$ has elements $+1$ or $-1$ on the diagonal)

then

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

is also a singular value decomposition of $A$. 

[NP3275/5/pdf] 3.f02wec.1
4. Parameters

\( m \)
- Input: the number of rows, \( m \), of the matrix \( A \).
- Constraint: \( m \geq 0 \).
- When \( m = 0 \) then an immediate return is effected.

\( n \)
- Input: the number of columns, \( n \), of the matrix \( A \).
- Constraint: \( n \geq 0 \).
- When \( n = 0 \) 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 \( m \geq n \) and \( \text{wantq} = \text{TRUE} \), then the leading \( m \) by \( n \) part of \( a \) will contain the first \( n \) columns of the orthogonal matrix \( Q \).
- If \( m < n \) and \( \text{wantp} = \text{TRUE} \), then the leading \( m \) by \( n \) part of \( a \) will contain the first \( m \) rows of the orthogonal matrix \( P^T \).
- If \( m \geq n \) and \( \text{wantq} = \text{FALSE} \) and \( \text{wantp} = \text{TRUE} \), then the leading \( n \) by \( n \) part 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 \text{nag_real_svd} is called.
- Constraint: \( tda \geq n \).

\( ncolb \)
- Input: \( ncolb \), the number of columns of the matrix \( B \). When \( ncolb = 0 \) the array \( b \) is not referenced.
- Constraint: \( ncolb \geq 0 \).

\( b[m][tdb] \)
- Input: if \( ncolb > 0 \), the leading \( m \) by \( ncolb \) part of the array \( b \) must contain the matrix to be transformed. If \( ncolb = 0 \) the array \( b \) is not referenced and may be set to the null pointer, i.e., \((\text{double } *)0\).
- 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 \text{nag_real_svd} is called.
- Constraint: if \( ncolb > 0 \) then \( tdb \geq ncolb \).

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

\( q[m][tdq] \)
- Output: if \( m < n \) and \( \text{wantq} = \text{TRUE} \), 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., \((\text{double } *)0\).

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

\( sv[min(m,n)] \)
- Output: the \( \min(m,n) \) diagonal elements of the matrix \( S \).

\( \text{wantp} \)
- Input: \( \text{wantp} \) must be \text{TRUE} if the right-hand singular vectors are required. If \( \text{wantp} = \text{FALSE} \), then the array \( pt \) is not referenced.
6.1. Accuracy

The computed factors $Q$, $D$, and $P$ satisfy the relation

$$QDP^T = A + E$$

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


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:

```c
/* 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

```c
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
");
    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.
");
        Vprintf("m = %2ld, n = %2ld
", m, n);
    }
    else
    {
        for (i=0 ;i<m ; ++i)
            for (j=0 ;j<n ; ++j)
                Vscanf("%lf", &a[i][j]);
        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
");
        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.
");
Vprintf("m = %2ld, n = %2ld
", 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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.6011</td>
<td>0.1961</td>
<td>0.3165</td>
</tr>
<tr>
<td>-0.6011</td>
<td>0.1961</td>
<td>0.3165</td>
</tr>
<tr>
<td>-0.4166</td>
<td>-0.1569</td>
<td>-0.6941</td>
</tr>
<tr>
<td>-0.1688</td>
<td>0.3922</td>
<td>-0.5636</td>
</tr>
<tr>
<td>0.2742</td>
<td>0.8629</td>
<td>-0.0139</td>
</tr>
</tbody>
</table>