NAG Library Function Document

nag_zggsvd (f08vnc)

1 Purpose

nag_zggsvd (f08vnc) computes the generalized singular value decomposition (GSVD) of an \( m \) by \( n \) complex matrix \( A \) and a \( p \) by \( n \) complex matrix \( B \).

2 Specification

```c
#include <nag.h>
#include <nagf08.h>

void nag_zggsvd (Nag_OrderType order, Nag_ComputeUType jobu,
                 Nag_ComputeVType jobv, Nag_ComputeQType jobq, Integer m, Integer n,
                 Integer p, Integer *k, Integer *l, Complex a[], Integer pda,
                 Complex b[], Integer pdb, double alpha[], double beta[], Complex u[],
                 Integer pdu, Complex v[], Integer pdv, Complex q[], Integer pdq,
                 Integer iwork[], NagError *fail)
```

3 Description

The generalized singular value decomposition is given by

\[
U^H A Q = D_1 \begin{pmatrix} 0 & R \end{pmatrix}, \quad V^H B Q = D_2 \begin{pmatrix} 0 & R \end{pmatrix},
\]

where \( U, V \) and \( Q \) are unitary matrices. Let \( (k + l) \) be the effective numerical rank of the matrix \( \begin{pmatrix} A \\ B \end{pmatrix} \), then \( R \) is a \( (k + l) \) by \( (k + l) \) nonsingular upper triangular matrix, \( D_1 \) and \( D_2 \) are \( m \) by \( (k + l) \) and \( p \) by \( (k + l) \) 'diagonal' matrices structured as follows:

if \( m - k - l \geq 0 \),

\[
D_1 = \begin{pmatrix} k & l \\ k & l & \ldots & \ldots & \ldots \\ & & & & \ldots \\ & & & & k & l \end{pmatrix},
\]

\[
D_2 = \begin{pmatrix} k & l \\ p & l & \ldots & \ldots & \ldots \\ & & & & k & l \end{pmatrix},
\]

\[
\begin{pmatrix} 0 & R \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix},
\]

where

\[
C = \text{diag}(\alpha_{k+1}, \ldots, \alpha_{k+l}),
\]

\[
S = \text{diag}(\beta_{k+1}, \ldots, \beta_{k+l}),
\]

and
\[ C^2 + S^2 = I. \]

\( R \) is stored as a submatrix of \( A \) with elements \( R_{ij} \) stored as \( A_{i,n-k-l+j} \) on exit.

If \( m-k-l < 0 \),

\[
D_1 = \begin{pmatrix}
  k & m-k & k+l-m \\
  m-k & 0 & 0 \\
  I & 0 & 0 \\
\end{pmatrix}
\]

\[
D_2 = \begin{pmatrix}
  m-k & k & m-k & k+l-m \\
  k+l-m & 0 & 0 & 0 \\
  p-l & 0 & 0 & 0 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
  0 & R \\
  \end{pmatrix} = \begin{pmatrix}
  k & m-k & k+l-m \\
  \end{pmatrix}
\]

where

\[
C = \text{diag}(\alpha_k, \ldots, \alpha_m),
\]

\[
S = \text{diag}(\beta_k, \ldots, \beta_m),
\]

and

\[ C^2 + S^2 = I. \]

\[
\begin{pmatrix}
  R_{11} & R_{12} & R_{13} \\
  0 & R_{22} & R_{23} \\
\end{pmatrix}
\]

is stored as a submatrix of \( A \) with \( R_{ij} \) stored as \( A_{i,n-k-l+j} \), and \( R_{13} \) is stored as a submatrix of \( B \) with \( (R_{13})_{ij} \) stored as \( B_{m-k+i,n+m-k-l+j} \).

The function computes \( C, S, R \) and, optionally, the unitary transformation matrices \( U, V \) and \( Q \).

In particular, if \( B \) is an \( n \) by \( n \) nonsingular matrix, then the GSVD of \( A \) and \( B \) implicitly gives the SVD of \( A \times B^{-1} \):

\[
AB^{-1} = UD_1D_2^{-1}V^H.
\]

If \( \begin{pmatrix} A \\ B \end{pmatrix} \) has orthonormal columns, then the GSVD of \( A \) and \( B \) is also equal to the CS decomposition of \( A \) and \( B \). Furthermore, the GSVD can be used to derive the solution of the eigenvalue problem:

\[
A^H Ax = \lambda B^H Bx.
\]

In some literature, the GSVD of \( A \) and \( B \) is presented in the form

\[
U^H AX = \begin{pmatrix} 0 & D_1 \end{pmatrix}, \quad V^H BX = \begin{pmatrix} 0 & D_2 \end{pmatrix},
\]

where \( U \) and \( V \) are orthogonal and \( X \) is nonsingular, and \( D_1 \) and \( D_2 \) are ‘diagonal’. The former GSVD form can be converted to the latter form by taking the nonsingular matrix \( X \) as

\[
X = Q \begin{pmatrix} I & 0 \\ 0 & R^{-1} \end{pmatrix}.
\]
4 References


5 Arguments

1: \texttt{order} – Nag_OrderType \textit{Input}

\textit{On entry:} the \texttt{order} argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by \texttt{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\textit{Constraint:} \texttt{order} = Nag_RowMajor or Nag_ColMajor.

2: \texttt{jobu} – Nag_ComputeUType \textit{Input}

\textit{On entry:} if \texttt{jobu} = Nag_AllU, the unitary matrix \( U \) is computed.

If \texttt{jobu} = Nag_NotU, \( U \) is not computed.

\textit{Constraint:} \texttt{jobu} = Nag_AllU or Nag_NotU.

3: \texttt{jobv} – Nag_ComputeVType \textit{Input}

\textit{On entry:} if \texttt{jobv} = Nag_ComputeV, the unitary matrix \( V \) is computed.

If \texttt{jobv} = Nag_NotV, \( V \) is not computed.

\textit{Constraint:} \texttt{jobv} = Nag_ComputeV or Nag_NotV.

4: \texttt{jobq} – Nag_ComputeQType \textit{Input}

\textit{On entry:} if \texttt{jobq} = Nag_ComputeQ, the unitary matrix \( Q \) is computed.

If \texttt{jobq} = Nag_NotQ, \( Q \) is not computed.

\textit{Constraint:} \texttt{jobq} = Nag_ComputeQ or Nag_NotQ.

5: \texttt{m} – Integer \textit{Input}

\textit{On entry:} \( m \), the number of rows of the matrix \( A \).

\textit{Constraint:} \( m \geq 0 \).

6: \texttt{n} – Integer \textit{Input}

\textit{On entry:} \( n \), the number of columns of the matrices \( A \) and \( B \).

\textit{Constraint:} \( n \geq 0 \).

7: \texttt{p} – Integer \textit{Input}

\textit{On entry:} \( p \), the number of rows of the matrix \( B \).

\textit{Constraint:} \( p \geq 0 \).

8: \texttt{k} – Integer * \textit{Output}

9: \texttt{l} – Integer * \textit{Output}

\textit{On exit:} \( k \) and \( l \) specify the dimension of the subblocks \( k \) and \( l \) as described in Section 3; \( (k + l) \) is the effective numerical rank of \( \begin{pmatrix} A \\ B \end{pmatrix} \).
10: \(\mathbf{a}[\text{dim}]\) – Complex

**Note:** the dimension, \(\text{dim}\), of the array \(\mathbf{a}\) must be at least

\[
\max(1, \text{pda} \times n) \quad \text{when order} = \text{Nag\_ColMajor};
\]
\[
\max(1, m \times \text{pda}) \quad \text{when order} = \text{Nag\_RowMajor}.
\]

The \((i,j)\)th element of the matrix \(\mathbf{A}\) is stored in

\[
\mathbf{a}[\{j - 1\} \times \text{pda} + i - 1] \quad \text{when order} = \text{Nag\_ColMajor};
\]
\[
\mathbf{a}[\{i - 1\} \times \text{pda} + j - 1] \quad \text{when order} = \text{Nag\_RowMajor}.
\]

*On entry:* the \(m\) by \(n\) matrix \(\mathbf{A}\).

*On exit:* contains the triangular matrix \(\mathbf{R}\), or part of \(\mathbf{R}\). See Section 3 for details.

11: \(\text{pda}\) – Integer

**Input**

*On entry:* the stride separating row or column elements (depending on the value of \(\text{order}\)) in the array \(\mathbf{a}\).

**Constraints:**

- if \(\text{order} = \text{Nag\_ColMajor}, \text{pda} \geq \max(1, m)\);
- if \(\text{order} = \text{Nag\_RowMajor}, \text{pda} \geq \max(1, n)\).

12: \(\mathbf{b}[\text{dim}]\) – Complex

**Input/Output**

**Note:** the dimension, \(\text{dim}\), of the array \(\mathbf{b}\) must be at least

\[
\max(1, \text{pdb} \times n) \quad \text{when order} = \text{Nag\_ColMajor};
\]
\[
\max(1, p \times \text{pdb}) \quad \text{when order} = \text{Nag\_RowMajor}.
\]

The \((i,j)\)th element of the matrix \(\mathbf{B}\) is stored in

\[
\mathbf{b}[\{j - 1\} \times \text{pdb} + i - 1] \quad \text{when order} = \text{Nag\_ColMajor};
\]
\[
\mathbf{b}[\{i - 1\} \times \text{pdb} + j - 1] \quad \text{when order} = \text{Nag\_RowMajor}.
\]

*On entry:* the \(p\) by \(n\) matrix \(\mathbf{B}\).

*On exit:* contains the triangular matrix \(\mathbf{R}\) if \(m - k - l < 0\). See Section 3 for details.

13: \(\text{pdb}\) – Integer

**Input**

*On entry:* the stride separating row or column elements (depending on the value of \(\text{order}\)) in the array \(\mathbf{b}\).

**Constraints:**

- if \(\text{order} = \text{Nag\_ColMajor}, \text{pdb} \geq \max(1, p)\);
- if \(\text{order} = \text{Nag\_RowMajor}, \text{pdb} \geq \max(1, n)\).

14: \(\text{alpha}[n]\) – double

**Output**

*On exit:* see the description of \(\text{beta}\).

15: \(\text{beta}[n]\) – double

**Output**

*On exit:* \(\text{alpha}\) and \(\text{beta}\) contain the generalized singular value pairs of \(\mathbf{A}\) and \(\mathbf{B}\), \(\alpha_i\) and \(\beta_i\);

\[
\text{ALPHA}(1 : k) = 1,
\]
\[
\text{BETA}(1 : k) = 0,
\]

and if \(m - k - l \geq 0\),

\[
\text{ALPHA}(k + 1 : k + l) = C,
\]
\[
\text{BETA}(k + 1 : k + l) = S,
\]

or if \(m - k - l < 0\),
\( \text{ALPHA}(k + 1 : m) = C, \)
\( \text{ALPHA}(m + 1 : k + 1) = 0, \)
\( \text{BETA}(k + 1 : m) = S, \)
\( \text{BETA}(m + 1 : k + 1) = 1, \) and
\( \text{ALPHA}(k + l + 1 : n) = 0, \)
\( \text{BETA}(k + l + 1 : n) = 0. \)

The notation \( \text{ALPHA}(k : n) \) above refers to consecutive elements \( \alpha[i - 1], \) for \( i = k, \ldots, n. \)

16: \( u[\text{dim}] \) – Complex

**Output**

**Note:** the dimension, \( \text{dim} \), of the array \( u \) must be at least

\[ \max(1, \text{pdu} \times m) \] when \( \text{jobu} = \text{Nag\_AllU} \);

1 otherwise.

The \((i,j)\)th element of the matrix \( U \) is stored in

\[ u[(j - 1) \times \text{pdu} + i - 1] \] when \( \text{order} = \text{Nag\_ColMajor} \);

\[ u[(i - 1) \times \text{pdu} + j - 1] \] when \( \text{order} = \text{Nag\_RowMajor} \).

**On exit:** if \( \text{jobu} = \text{Nag\_AllU} \), \( u \) contains the \( m \) by \( m \) unitary matrix \( U \).

If \( \text{jobu} = \text{Nag\_NotU} \), \( u \) is not referenced.

17: \( \text{pdu} \) – Integer

**Input**

**On entry:** the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( u \).

**Constraints:**

- if \( \text{jobu} = \text{Nag\_AllU} \), \( \text{pdu} \geq \max(1, m) \);
- otherwise \( \text{pdu} \geq 1. \)

18: \( v[\text{dim}] \) – Complex

**Output**

**Note:** the dimension, \( \text{dim} \), of the array \( v \) must be at least

\[ \max(1, \text{pdv} \times p) \] when \( \text{jobv} = \text{Nag\_ComputeV} \);

1 otherwise.

The \((i,j)\)th element of the matrix \( V \) is stored in

\[ v[(j - 1) \times \text{pdv} + i - 1] \] when \( \text{order} = \text{Nag\_ColMajor} \);

\[ v[(i - 1) \times \text{pdv} + j - 1] \] when \( \text{order} = \text{Nag\_RowMajor} \).

**On exit:** if \( \text{jobv} = \text{Nag\_ComputeV} \), \( v \) contains the \( p \) by \( p \) unitary matrix \( V \).

If \( \text{jobv} = \text{Nag\_NotV} \), \( v \) is not referenced.

19: \( \text{pdv} \) – Integer

**Input**

**On entry:** the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( v \).

**Constraints:**

- if \( \text{jobv} = \text{Nag\_ComputeV} \), \( \text{pdv} \geq \max(1, p) \);
- otherwise \( \text{pdv} \geq 1. \)
Note: the dimension, dim, of the array q must be at least
max(1, pdq × n) when jobq = Nag_ComputeQ;
1 otherwise.
The (i,j)th element of the matrix Q is stored in
q[(j-1) × pdq + i - 1] when order = Nag_ColMajor;
q[(i-1) × pdq + j - 1] when order = Nag_RowMajor.
On exit: if jobq = Nag_ComputeQ, q contains the n by n unitary matrix Q.
If jobq = Nag_NotQ, q is not referenced.

On entry: the stride separating row or column elements (depending on the value of order) in the array q.
Constraints:
if jobq = Nag_ComputeQ, pdq ≥ max(1, n);
otherwise pdq ≥ 1.

On exit: stores the sorting information. More precisely, the following loop will sort alpha such
that alpha[0] ≥ alpha[1] ≥ · · · ≥ alpha[n - 1].

On entry, argument ⟨value⟩ had an illegal value.
The Jacobi-type procedure failed to converge.
On entry, jobq = ⟨value⟩, pdq = ⟨value⟩ and n = ⟨value⟩.
Constraint: if jobq = Nag_ComputeQ, pdq ≥ max(1, n);
otherwise pdq ≥ 1.
On entry, jobu = ⟨value⟩, pdu = ⟨value⟩ and m = ⟨value⟩.
Constraint: if jobu = Nag_AllU, pdu ≥ max(1, m);
otherwise pdu ≥ 1.
On entry, jobv = ⟨value⟩, pdv = ⟨value⟩ and p = ⟨value⟩.
Constraint: if jobv = Nag_ComputeV, pdv ≥ max(1, p);
otherwise pdv ≥ 1.
NE_INT
On entry, \( m = \langle \text{value} \rangle \).
Constraint: \( m \geq 0 \).
On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 0 \).
On entry, \( p = \langle \text{value} \rangle \).
Constraint: \( p \geq 0 \).
On entry, \( pda = \langle \text{value} \rangle \).
Constraint: \( pda > 0 \).
On entry, \( pdb = \langle \text{value} \rangle \).
Constraint: \( pdb > 0 \).
On entry, \( pdq = \langle \text{value} \rangle \).
Constraint: \( pdq > 0 \).
On entry, \( pdu = \langle \text{value} \rangle \).
Constraint: \( pdu > 0 \).
On entry, \( pdv = \langle \text{value} \rangle \).
Constraint: \( pdv > 0 \).

NE_INT_2
On entry, \( pda = \langle \text{value} \rangle \) and \( m = \langle \text{value} \rangle \).
Constraint: \( pda \geq \max(1, m) \).
On entry, \( pda = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( pda \geq \max(1, n) \).
On entry, \( pdb = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( pdb \geq \max(1, n) \).
On entry, \( pdb = \langle \text{value} \rangle \) and \( p = \langle \text{value} \rangle \).
Constraint: \( pdb \geq \max(1, p) \).

NE_INTERNAL_ERROR
An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.
An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

7 Accuracy
The computed generalized singular value decomposition is nearly the exact generalized singular value decomposition for nearby matrices \((A + E)\) and \((B + F)\), where
\[
\|E\|_2 = O(\epsilon)\|A\|_2 \text{ and } \|F\|_2 = O(\epsilon)\|B\|_2,
\]
and \( \epsilon \) is the machine precision. See Section 4.12 of Anderson et al. (1999) for further details.

8 Parallelism and Performance
nag_zggsvd (f08vnc) is not threaded by NAG in any implementation.
nag_zggsvd (f08vnc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments

The diagonal elements of the matrix $R$ are real.

The real analogue of this function is nag_dggsvd (f08vac).

10 Example

This example finds the generalized singular value decomposition

$$A = U \Sigma_1 (0 \; R) Q^H, \quad B = V \Sigma_2 (0 \; R) Q^H,$$

where

$$A = \begin{pmatrix}
0.96 - 0.81i & -0.03 + 0.96i & -0.91 + 2.06i & -0.05 + 0.41i \\
-0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & -0.81 + 0.56i \\
0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\
0.37 + 0.38i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\
0.83 + 0.51i & 0.20 + 0.01i & -0.17 - 0.46i & 1.47 + 1.59i \\
1.08 - 0.28i & 0.20 - 0.12i & -0.07 + 1.23i & 0.26 + 0.26i
\end{pmatrix},$$

and

$$B = \begin{pmatrix}
1 & 0 & -1 & 0 \\
0 & 1 & 0 & -1
\end{pmatrix},$$

together with estimates for the condition number of $R$ and the error bound for the computed generalized singular values.

The example program assumes that $m \geq n$, and would need slight modification if this is not the case.

10.1 Program Text

/* nag_zggsvd (f08vnc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 9, 2009.
*/
#include <stdio.h>
#include <nag.h>
#include <nagf08.h>
#include <nagx04.h>
#include <nagf07.h>
#include <nagx02.h>

int main(void) {
  /* Scalars */
  double d, eps, rcond, serrbd;
  Integer exit_status = 0, i, irank, j, k, l, m, n, p,
            pda, pdb, pdq, pdu, pdv;
  NagError fail;
  Nag_OrderType order;
/* Arrays */
char *clabs = 0, *rlabs = 0;
Complex n; *a = 0, *b = 0, *q = 0, *u = 0, *v = 0;
double *alpha = 0, *beta = 0;
Integer *iwork = 0;

#ifdef NAG_COLUMN_MAJOR
#define A(I, J) a[(J-1)*pda + I - 1]
#define B(I, J) b[(J-1)*pdb + I - 1]
order = Nag_ColMajor;
#else
#define A(I, J) a[(I-1)*pda+J-1]
#define B(I, J) b[(I-1)*pdb +J-1]
order = Nag_RowMajor;
#endif

INIT_FAIL(fail);

printf("nag_zggsvd (f08vnc) Example Program Results\n\n");
/* Skip heading in data file */
#ifdef _WIN32
scanf_s("%*[\n] ");
#else
scanf("%*[\n] ");
#endif

#ifdef _WIN32
scanf_s("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &m, &n, &p);
#else
scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &m, &n, &p);
#endif
if (m <= 10 && n <= 10 && p <= 10)
{
/* Allocate memory */
 if (!((clabs = NAG_ALLOC(2, char)) || (rlabs = NAG_ALLOC(2, char)) ||
 (a = NAG_ALLOC(m*n, Complex)) ||
 (b = NAG_ALLOC(p*n, Complex)) ||
 (q = NAG_ALLOC(n*n, Complex)) ||
 (u = NAG_ALLOC(m*m, Complex)) ||
 (v = NAG_ALLOC(p*p, Complex)) ||
 (alpha = NAG_ALLOC(n, double)) ||
 (beta = NAG_ALLOC(n, double)) ||
 (iwork = NAG_ALLOC(n, Integer)) ))
 {
  printf("Allocation failure\n");
  exit_status = -1;
  goto END;
 }  
#ifdef NAG_COLUMN_MAJOR
 pda = m;
pdb = p;
pdq = n;
pdu = m;
pdv = p;
#else
 pda = n;
pdb = n;
pdq = n;
pdu = m;
pdv = p;
#endif
}
else
{
 printf("m and/or n too small\n");
goto END;
}

/* Read the m by n matrix A and p by n matrix B from data file */
for (i = 1; i <= m; ++i)
for (j = 1; j <= n; ++j)
#define _WIN32
    scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#else
    scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#endif
#define _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
for (i = 1; i <= p; ++i)
    for (j = 1; j <= n; ++j)
#define _WIN32
    scanf_s(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
#else
    scanf(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
#endif
#define _WIN32
    scanf("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
/* nag_zggsvd (f08vnc)
 * Compute the generalized singular value decomposition of (A, B)
 * (A = U*D1*(0 R)*(Q**H), B = V*D2*(0 R)*(Q**H), m.ge.n)
 */
nag_zggsvd(order, Nag_AllU, Nag_ComputeV, Nag_ComputeQ, m, n, p, &k, &l, a,
pda, b, pdb, alpha, beta, u, pdu, v, pdv, q, pdq, iwork, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zggsvd (f08vnc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print solution */
irank = k + 1;
printf("Number of infinite generalized singular values (K)\n");
printf("%5"NAG_IFMT"\n", k);
printf("Number of finite generalized singular values (L)\n");
printf("%5"NAG_IFMT"\n", l);
printf("Numerical rank of (A**H B**H)**H (K+L)\n");
printf("%5"NAG_IFMT"\n", irank);
printf("Finite generalized singular values\n");
for (j = k; j < irank; ++j)
{
    d = alpha[j] / beta[j];
    printf("%13.4e%"s, d, (j+1)%8 == 0 || (j+1) == irank?"\n": "");
}
printf("\n");
fflush(stdout);
nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m, m, u, pdu, Nag_BracketForm, "%13.4e",
"Orthogonal matrix U", Nag_IntegerLabels, 0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
printf("\n");
fflush(stdout);
nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag, p, p, v, pdv, Nag_BracketForm, "%13.4e",
...
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_ztrcon (f07tuc)
* estimate the reciprocal condition number of R
*/
nag_ztrcon(order, Nag_InfNorm, Nag_Upper, Nag_NonUnitDiag, irank, irank, &A(1, n - irank + 1), pda, &rcond, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ztrcon (f07tuc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* So long as irank = n, get the machine precision, eps, and compute the
* approximate error bound for the computed generalized singular values
*/
if (irank == n)
{
    eps = nag_machine_precision;
    serrbd = eps / rcond;
    printf("Error estimate for the generalized singular values\n"");
    printf("%11.1e", serrbd);
}
else
{
    printf("(A**H B**H)**H is not of full rank\n");
}
END:

NAG_FREE(clabs);
NAG_FREE(rlabs);
NAG_FREE(a);
NAG_FREE(b);
NAG_FREE(q);
NAG_FREE(u);
NAG_FREE(v);
NAG_FREE(alpha);
NAG_FREE(beta);
NAG_FREE(iwork);

return exit_status;
}

#define B
#define A

10.2 Program Data

nag_zggsvd (f08vnc) Example Program Data

<table>
<thead>
<tr>
<th>M</th>
<th>N</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Values of M, N and P

( 0.96,-0.81) (-0.03, 0.96) (-0.91, 2.06) (-0.05, 0.41)
(-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42) (-0.81, 0.56)
( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
( 0.37, 0.38) ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
( 0.81, 0.51) ( 0.20, 0.01) (-0.17,-0.46) ( 1.47, 1.59)
( 1.08,-0.28) ( 0.20,-0.12) (-0.07, 1.23) ( 0.26, 0.26)

End of matrix A

( 1.00, 0.00) ( 0.00, 0.00) (-1.00, 0.00) ( 0.00, 0.00)
( 0.00, 0.00) ( 1.00, 0.00) ( 0.00, 0.00) (-1.00, 0.00)

End of matrix B

10.3 Program Results

nag_zggsvd (f08vnc) Example Program Results

Number of infinite generalized singular values (K)
2
Number of finite generalized singular values (L)
2
Numerical rank of (A**H B**H)**H (K+L)
4

Finite generalized singular values
2.0720e+00 1.1058e+00

Orthogonal matrix U

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(-1.4039e-01, -2.6167e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(-1.4039e-01, -2.6167e-01)</td>
</tr>
<tr>
<td>(4.2764e-01, -6.2582e-01)</td>
<td>(8.6298e-02, -3.8174e-02)</td>
<td>(3.2595e-01, 1.6428e-01)</td>
<td>(3.5190e-01, -2.1251e-03)</td>
<td>(1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(3.2595e-01, 1.6428e-01)</td>
<td>(3.8163e-01, -1.8219e-01)</td>
<td>(1.3038e-02, -3.2595e-01)</td>
<td>(3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(1.5906e-01, -5.2151e-03)</td>
<td>(-2.8207e-01, 1.9732e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(1.7210e-01, -1.3038e-02)</td>
<td>(-5.0942e-01, -5.0319e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(2.6336e-01, -2.4772e-01)</td>
<td>(-1.0861e-01, 2.8474e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(2.5177e-01, -7.9789e-01)</td>
<td>(-5.0956e-02, -2.1750e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(3.1231e-01, -1.4565e-02)</td>
<td>(1.1979e-01, 1.6319e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(2.1598e-01, 1.8813e-01)</td>
<td>(-4.0163e-01, 2.6787e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(3.6483e-02, 1.9271e-01)</td>
<td>( 1.5906e-01, 1.5574e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(1.0906e-01, -1.2712e-01)</td>
<td>(-8.8159e-02, 5.6169e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(4.5947e-02, 1.4052e-04)</td>
<td>(-5.2773e-02, -2.2492e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
<tr>
<td>(8.0311e-02, 1.3605e-01)</td>
<td>(-3.8117e-02, -2.1907e-01)</td>
<td>(-3.2595e-01, 1.6428e-01)</td>
<td>(-3.5190e-01, -2.1251e-03)</td>
<td>(-1.3038e-02, -3.2595e-01)</td>
<td>(4.2764e-01, -6.2582e-01)</td>
</tr>
</tbody>
</table>
% 3 ( 5.9714e-02, -5.8974e-01) ( -1.3850e-01, -9.0941e-02)
% 4 ( -4.6443e-02, 3.0864e-01) ( -3.7354e-01, -5.5148e-01)
% 5 ( 5.7843e-01, -1.2439e-01) ( -1.8815e-02, -5.5686e-02)
% 6 ( 1.5763e-02, 4.7130e-02) ( 6.5007e-01, 4.9173e-03)

Orthogonal matrix V

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 9.8930e-01, 1.0471e-19)</td>
<td>( -1.1461e-01, 9.0250e-02)</td>
</tr>
</tbody>
</table>

Orthogonal matrix Q

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 7.0711e-01, 0.0000e+00)</td>
<td>( 0.0000e+00, 0.0000e+00)</td>
</tr>
<tr>
<td>( 0.0000e+00, 0.0000e+00)</td>
<td>( 7.0711e-01, 0.0000e+00)</td>
</tr>
</tbody>
</table>

Non singular upper triangular matrix R

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( -2.7118e+00, 0.0000e+00)</td>
<td>( -1.4390e+00, -1.0315e+00)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( -1.8583e+00, 0.0000e+00)</td>
<td>( -8.1044e-02, -6.3817e-02)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 3.2537e+00, 0.0000e+00)</td>
<td>( -2.1084e+00, 0.0000e+00)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Estimate of reciprocal condition number for R
1.3e-01

Error estimate for the generalized singular values
8.3e-16