NAG Library Function Document

nag_zgels (f08anc)

1 Purpose

nag_zgels (f08anc) solves linear least squares problems of the form

\[
\min_x \| b - Ax \|_2 \quad \text{or} \quad \min_x \| b - A^H x \|_2,
\]

where \( A \) is an \( m \) by \( n \) complex matrix of full rank, using a QR or LQ factorization of \( A \).

2 Specification

```c
#include <nag.h>
#include <nagf08.h>
void nag_zgels (Nag_OrderType order, Nag_TransType trans, Integer m,
            Integer n, Integer nrhs, Complex a[], Integer pda, Complex b[],
            Integer pdb, NagError *fail)
```

3 Description

The following options are provided:

1. If \( \text{trans} = \text{Nag_NoTrans} \) and \( m \geq n \): find the least squares solution of an overdetermined system, i.e., solve the least squares problem

\[
\min_x \| b - Ax \|_2.
\]

2. If \( \text{trans} = \text{Nag_NoTrans} \) and \( m < n \): find the minimum norm solution of an underdetermined system \( Ax = b \).

3. If \( \text{trans} = \text{Nag_ConjTrans} \) and \( m \geq n \): find the minimum norm solution of an undetermined system \( A^H x = b \).

4. If \( \text{trans} = \text{Nag_ConjTrans} \) and \( m < n \): find the least squares solution of an overdetermined system, i.e., solve the least squares problem

\[
\min_x \| b - A^H x \|_2.
\]

Several right-hand side vectors \( b \) and solution vectors \( x \) can be handled in a single call; they are stored as the columns of the \( m \) by \( r \) right-hand side matrix \( B \) and the \( n \) by \( r \) solution matrix \( X \).

4 References


5 Arguments

1: \( \text{order} \) – Nag_OrderType

Input

On entry: the \text{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
order = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

Constraint: order = Nag_RowMajor or Nag_ColMajor.

2: trans – Nag_TransType

On entry: if trans = Nag_NoTrans, the linear system involves $A$.
If trans = Nag_ConjTrans, the linear system involves $A^H$.

Constraint: trans = Nag_NoTrans or Nag_ConjTrans.

3: m – Integer

On entry: $m$, the number of rows of the matrix $A$.

Constraint: $m \geq 0$.

4: n – Integer

On entry: $n$, the number of columns of the matrix $A$.

Constraint: $n \geq 0$.

5: nrhs – Integer

On entry: $r$, the number of right-hand sides, i.e., the number of columns of the matrices $B$ and $X$.

Constraint: nrhs $\geq 0$.

6: a[dim] – Complex

Note: the dimension, dim, of the array a must be at least

$max(1, pd\times n)$ when order = Nag_ColMajor;
$max(1, m \times pd\) when order = Nag_RowMajor.$

The $(i,j)$th element of the matrix $A$ is stored in

$a[(j-1) \times pd\times i + i - 1]$ when order = Nag_ColMajor;
$a[(i-1) \times pd\times j + j - 1]$ when order = Nag_RowMajor.$

On entry: the $m$ by $n$ matrix $A$.

On exit: if $m \geq n$, a is overwritten by details of its $QR$ factorization, as returned by nag_zgeqrf (f08asc).
If $m < n$, a is overwritten by details of its $LQ$ factorization, as returned by nag_zgelqf (f08avc).

7: pda – Integer

On entry: the stride separating row or column elements (depending on the value of order) in the array a.

Constraints:

if order = Nag_ColMajor, pda $\geq max(1, m)$;
if order = Nag_RowMajor, pda $\geq max(1, n)$.

8: b[dim] – Complex

Note: the dimension, dim, of the array b must be at least

$max(1, pd\times nrhs)$ when order = Nag_ColMajor;
$max(1, max(1, m, n) \times pd\) when order = Nag_RowMajor.$
The \((i,j)\)th element of the matrix \(B\) is stored in
\[
\begin{align*}
  b[(j-1) \times \text{pdb} + i - 1] & \text{ when } \text{order} = \text{Nag\_ColMajor}; \\
  b[(i-1) \times \text{pdb} + j - 1] & \text{ when } \text{order} = \text{Nag\_RowMajor}.
\end{align*}
\]

On entry: the matrix \(B\) of right-hand side vectors, stored in rows or columns; \(b\) is \(m\) by \(r\) if \(\text{trans} = \text{Nag\_NoTrans}, \) or \(n\) by \(r\) if \(\text{trans} = \text{Nag\_ConjTrans}\).

On exit: \(b\) is overwritten by the solution vectors, \(x\), stored in rows or columns:

- if \(\text{trans} = \text{Nag\_NoTrans} \) and \(m \geq n\), or \(\text{trans} = \text{Nag\_ConjTrans} \) and \(m < n\), elements 1 to \(\min(m,n)\) in each column of \(b\) contain the least squares solution vectors; the residual sum of squares for the solution is given by the sum of squares of the modulus of elements \((\min(m,n)+1)\) to \(\max(m,n)\) in that column;
- otherwise, elements 1 to \(\max(m,n)\) in each column of \(b\) contain the minimum norm solution vectors.

9: \(\text{pdb} \) – Integer \hspace{1cm} \(\text{Input}\)

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

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

10: \(\text{fail} \) – NagError * \hspace{1cm} \(\text{Input/Output}\)

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

**NE\_ALLOC\_FAIL**

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE\_BAD\_PARAM**

On entry, argument \(\langle\text{value}\rangle\) had an illegal value.

**NE\_FULL\_RANK**

Diagonal element \(\langle\text{value}\rangle\) of the triangular factor of \(A\) is zero, so that \(A\) does not have full rank; the least squares solution could not be computed.

**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, \(\text{nrhs} = \langle\text{value}\rangle\).
Constraint: \(\text{nrhs} \geq 0\).

On entry, \(\text{pda} = \langle\text{value}\rangle\).
Constraint: \(\text{pda} > 0\).

On entry, \(\text{pdb} = \langle\text{value}\rangle\).
Constraint: \(\text{pdb} > 0\).
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 $nrhs = \langle \text{value} \rangle$.
Constraint: $pdb \geq \max(1, nrhs)$.

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

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.

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.

See Section 4.5 of Anderson et al. (1999) for details of error bounds.

nag_zgels (f08anc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zgels (f08anc) 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.

The real analogue of this function is nag_dgels (f08aac).

This example solves the linear least squares problem

$$\min_{x} \|b - Ax\|_2,$$

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

\[
b = \begin{pmatrix}
-2.09 + 1.93i \\
3.34 - 3.53i \\
-4.94 - 2.04i \\
0.17 + 4.23i \\
-5.19 - 3.63i \\
0.98 + 2.53i
\end{pmatrix}
\]

The square root of the residual sum of squares is also output.

\section{10.1 Program Text}
/* nag_zgels (f08anc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 23, 2011. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagf16.h>

int main(void)
{
    /* Scalars */
    double rnorm;
    Integer exit_status = 0, i, j, m, n, nrhs, pda, pdb;
    /* Arrays */
    Complex *a = 0, *b = 0;
    /* Nag Types */
    Nag_OrderType order;
    NagError fail;
    #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_zgels (f08anc) Example Program Results\n\n");
    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n"]);
    #else
    scanf("%*[\n"]);
    #endif
    #ifdef _WIN32
    scanf("%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%[\n"]", &m, &n, &nrhs);
    #else
    scanf("%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%[\n"]", &m, &n, &nrhs);
    #endif

Mark 25
```c
// Allocate memory */
if (!(a = NAG_ALLOC(m*n, Complex)) || 
    !(b = NAG_ALLOC(m*nrhs, Complex)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

#ifndef NAG_COLUMN_MAJOR
pda = m;
pdb = m;
#else
pda = n;
pdb = nrhs;
#endif

/* Read A and B from data file */
for (i = 1; i <= m; ++i)
    for (j = 1; j <= n; ++j)
#ifdef _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
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif

#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif

/* nag_zgels (f08anc). * Solve the least squares problem min( norm2(b - Ax) ) for x. */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zgels (f08anc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print solution */
printf("Least squares solution\n");
for (i = 1; i <= n; ++i) {
    for (j = 1; j <= nrhs; ++j)
        printf("(%7.4f, %7.4f)%s", B(i, j).re, B(i, j).im, j%4 == 0?"\n":" ");
        printf("\n");
}

/* nag_zge_norm (f16uac). * Compute and print estimate of the square root of the residual sum of squares. */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zgels (f08anc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_zge_norm (f16uac). * Compute and print estimate of the square root of the residual sum of squares. */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zgels (f08anc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
```

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zge_norm (f16uac).\n\n\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("\nSquare root of the residual sum of squares\n");
printf("%11.2e\n", rnorm);

END:
    NAG_FREE(a);
    NAG_FREE(b);
    return exit_status;
}
#undef A
#undef B

10.2 Program Data

nag_zgels (f08anc) Example Program Data

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>nrhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Values of m, n and nrhs

(-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)
(0.83, 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

(-2.09, 1.93)
(3.34, -3.53)
(-4.94, -2.04)
(0.17, 4.23)
(-5.19, 3.63)
(0.98, 2.53) :End of vector b

10.3 Program Results

nag_zgels (f08anc) Example Program Results

Least squares solution
(-0.5044, -1.2179)
(-2.4281, 2.8574)
(1.4872, -2.1955)
(0.4537, 2.6904)

Square root of the residual sum of squares
6.88e-02