NAG Library Function Document

nag_dgels (f08aac)

1 Purpose

nag_dgels (f08aac) solves linear least squares problems of the form

$$\min_x \|b - Ax\|_2 \quad \text{or} \quad \min_x \|b - A^{\mathrm{T}}x\|_2,$$

where A is an m by n real matrix of full rank, using a QR or LQ factorization of A.

2 Specification

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

```
void nag_dgels (Nag_OrderType order, Nag_TransType trans, Integer m,
     Integer n, Integer nrhs, double a[], Integer pda, double b[],
     Integer pdb, NagError *fail)
```

3 Description

The following options are provided:

1. If **trans** $=$ 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 **trans** $=$ Nag_NoTrans and $m < n$: find the minimum norm solution of an underdetermined system $Ax = b$.
3. If **trans** $=$ Nag_Trans and $m \geq n$: find the minimum norm solution of an undetermined system $A^{\mathrm{T}}x = b$.
4. If **trans** $=$ Nag_Trans and $m < n$: find the least squares solution of an overdetermined system, i.e., solve the least squares problem
$$\min_x \|b - A^{\mathrm{T}}x\|_2.$$

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

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia http://www.netlib.org/lapack/lug

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

5 Arguments

1: **order** – Nag_OrderType *Input*

On entry: the **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 *Input*

On entry: if **trans** = Nag_NoTrans, the linear system involves A.

If **trans** = Nag_Trans, the linear system involves A^{T}.

Constraint: **trans** = Nag_NoTrans or Nag_Trans.

3: **m** – Integer *Input*

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

Constraint: $\mathbf{m} \geq 0$.

4: **n** – Integer *Input*

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

Constraint: $\mathbf{n} \geq 0$.

5: **nrhs** – Integer *Input*

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

Constraint: $\mathbf{nrhs} \geq 0$.

6: **a**[*dim*] – double *Input/Output*

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

$\max(1, \mathbf{pda} \times \mathbf{n})$ when **order** = Nag_ColMajor;
$\max(1, \mathbf{m} \times \mathbf{pda})$ when **order** = Nag_RowMajor.

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

$\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1]$ when **order** = Nag_RowMajor.

On entry: the m by n matrix A.

On exit: if $\mathbf{m} \geq \mathbf{n}$, **a** is overwritten by details of its QR factorization, as returned by nag_dgeqrf (f08aec).

If $\mathbf{m} < \mathbf{n}$, **a** is overwritten by details of its LQ factorization, as returned by nag_dgelqf (f08ahc).

7: **pda** – Integer *Input*

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

Constraints:

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

8: **b**[*dim*] – double *Input/Output*

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

$\max(1, \mathbf{pdb} \times \mathbf{nrhs})$ when **order** = Nag_ColMajor;
$\max(1, \max(1, \mathbf{m}, \mathbf{n}) \times \mathbf{pdb})$ when **order** = Nag_RowMajor.

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

b$[(j-1) \times$ **pdb** $+ i - 1]$ when **order** = Nag_ColMajor;
b$[(i-1) \times$ **pdb** $+ j - 1]$ when **order** = Nag_RowMajor.

On entry: the matrix B of right-hand side vectors, stored in rows or columns; **b** is m by r if **trans** = Nag_NoTrans, or n by r if **trans** = Nag_Trans.

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

if **trans** = Nag_NoTrans and $m \geq n$, or **trans** = Nag_Trans 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: **pdb** – Integer *Input*

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

Constraints:

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

10: **fail** – NagError * *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 ⟨*value*⟩ had an illegal value.

NE_FULL_RANK

Diagonal element ⟨*value*⟩ 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** = ⟨*value*⟩.
Constraint: **m** ≥ 0.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **nrhs** = ⟨*value*⟩.
Constraint: **nrhs** ≥ 0.

On entry, **pda** = ⟨*value*⟩.
Constraint: **pda** > 0.

On entry, **pdb** = ⟨*value*⟩.
Constraint: **pdb** > 0.

NE_INT_2

On entry, **pda** $= \langle value \rangle$ and **m** $= \langle value \rangle$.
Constraint: **pda** $\geq \max(1, \mathbf{m})$.

On entry, **pda** $= \langle value \rangle$ and **n** $= \langle value \rangle$.
Constraint: **pda** $\geq \max(1, \mathbf{n})$.

On entry, **pdb** $= \langle value \rangle$ and **nrhs** $= \langle value \rangle$.
Constraint: **pdb** $\geq \max(1, \mathbf{nrhs})$.

NE_INT_3

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

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

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

8 Parallelism and Performance

nag_dgels (f08aac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dgels (f08aac) 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 total number of floating-point operations required to factorize A is approximately $\frac{2}{3}n^2(3m - n)$ if $m \geq n$ and $\frac{2}{3}m^2(3n - m)$ otherwise. Following the factorization the solution for a single vector x requires $O(\min(m^2, n^2))$ operations.

The complex analogue of this function is nag_zgels (f08anc).

10 Example

This example solves the linear least squares problem

$$\min_x \|b - Ax\|_2,$$

where

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} -2.67 \\ -0.55 \\ 3.34 \\ -0.77 \\ 0.48 \\ 4.10 \end{pmatrix}.$$

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

10.1 Program Text

```
/* nag_dgels (f08aac) 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 */
  double        *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_dgels (f08aac) 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, &nrhs);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[^\n]", &m, &n, &nrhs);
#endif

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

```
    }

#ifdef 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", &A(i, j));
#else
      scanf("%lf", &A(i, j));
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  for (i = 1; i <= m; ++i)
    for (j = 1; j <= nrhs; ++j)
#ifdef _WIN32
      scanf_s("%lf", &B(i, j));
#else
      scanf("%lf", &B(i, j));
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* nag_dgels (f08aac).
   * Solve the least squares problem min( norm2(b - Ax) ) for x.
   */
  nag_dgels(order, Nag_NoTrans, m, n, nrhs, a, pda, b, pdb, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dgels (f08aac).\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("%11.4f%s", B(i, j), j%7 == 0?"\n":" ");
    printf("\n");
  }

  /* nag_dge_norm (f16rac).
   * Compute and print estimate of the square root of the residual
   * sum of squares.
   */
  nag_dge_norm(order, Nag_FrobeniusNorm, m - n, 1, &B(n + 1,1), pdb, &rnorm,
               &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dge_norm (f16rac).\n%s\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_dgels (f08aac) Example Program Data

  6      4      1           :Values of m, n and nrhs

 -0.57  -1.28  -0.39   0.25
 -1.93   1.08  -0.31  -2.14
  2.30   0.24   0.40  -0.35
 -1.93   0.64  -0.66   0.08
  0.15   0.30   0.15  -2.13
 -0.02   1.03  -1.43   0.50 :End of matrix A

 -2.67
 -0.55
  3.34
 -0.77
  0.48
  4.10                      :End of vector b
```

10.3 Program Results

```
nag_dgels (f08aac) Example Program Results

Least squares solution
     1.5339
     1.8707
    -1.5241
     0.0392

Square root of the residual sum of squares
   2.22e-02
```