NAG Library Function Document

nag_dgeev (f08nac)

1 Purpose

nag_dgeev (f08nac) computes the eigenvalues and, optionally, the left and/or right eigenvectors for an n by n real nonsymmetric matrix A.

2 Specification

```
#include <nag.h>
#include <nagf08.h>
void nag_dgeev (Nag_OrderType order, Nag_LeftVecsType jobvl,
     Nag_RightVecsType jobvr, Integer n, double a[], Integer pda,
     double wr[], double wi[], double vl[], Integer pdvl, double vr[],
     Integer pdvr, NagError *fail)
```

3 Description

The right eigenvector v_j of A satisfies

$$Av_j = \lambda_j v_j$$

where λ_j is the jth eigenvalue of A. The left eigenvector u_j of A satisfies

$$u_j^{\mathrm{H}} A = \lambda_j u_j^{\mathrm{H}}$$

where u_j^{H} denotes the conjugate transpose of u_j.

The matrix A is first reduced to upper Hessenberg form by means of orthogonal similarity transformations, and the QR algorithm is then used to further reduce the matrix to upper quasi-triangular Schur form, T, with 1 by 1 and 2 by 2 blocks on the main diagonal. The eigenvalues are computed from T, the 2 by 2 blocks corresponding to complex conjugate pairs and, optionally, the eigenvectors of T are computed and backtransformed to the eigenvectors of A.

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: **jobvl** – Nag_LeftVecsType *Input*

On entry: if **jobvl** = Nag_NotLeftVecs, the left eigenvectors of A are not computed.

If **jobvl** = Nag_LeftVecs, the left eigenvectors of A are computed.

Constraint: **jobvl** = Nag_NotLeftVecs or Nag_LeftVecs.

3: **jobvr** – Nag_RightVecsType *Input*

On entry: if **jobvr** = Nag_NotRightVecs, the right eigenvectors of A are not computed.

If **jobvr** = Nag_RightVecs, the right eigenvectors of A are computed.

Constraint: **jobvr** = Nag_NotRightVecs or Nag_RightVecs.

4: **n** – Integer *Input*

On entry: n, the order of the matrix A.

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

5: **a**[dim] – double *Input/Output*

Note: the dimension, *dim*, of the array **a** must be at least $\max(1, \mathbf{pda} \times \mathbf{n})$.

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 n by n matrix A.

On exit: **a** has been overwritten.

6: **pda** – Integer *Input*

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

Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

7: **wr**[dim] – double *Output*
8: **wi**[dim] – double *Output*

Note: the dimension, *dim*, of the arrays **wr** and **wi** must be at least $\max(1, \mathbf{n})$.

On exit: **wr** and **wi** contain the real and imaginary parts, respectively, of the computed eigenvalues. Complex conjugate pairs of eigenvalues appear consecutively with the eigenvalue having the positive imaginary part first.

9: **vl**[dim] – double *Output*

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

$\max(1, \mathbf{pdvl} \times \mathbf{n})$ when **jobvl** = Nag_LeftVecs;
1 otherwise.

Where $\mathbf{VL}(i,j)$ appears in this document, it refers to the array element

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

On exit: if **jobvl** = Nag_LeftVecs, the left eigenvectors u_j are stored one after another in **vl**, in the same order as their corresponding eigenvalues. If the jth eigenvalue is real, then $u_j = \mathbf{VL}(i,j)$, for $i = 1, 2, \ldots, \mathbf{n}$. If the jth and $(j+1)$st eigenvalues form a complex conjugate pair, then $u_j = \mathbf{VL}(i,j) + i \times \mathbf{VL}(i,j+1)$ and $u_{j+1} = \mathbf{VL}(i,j) - i \times \mathbf{VL}(i,j+1)$, for $i = 1, 2, \ldots, \mathbf{n}$.

If **jobvl** = Nag_NotLeftVecs, **vl** is not referenced.

10: **pdvl** – Integer *Input*

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

Constraints:

if **jobvl** = Nag_LeftVecs, **pdvl** $\geq \max(1, \mathbf{n})$;
otherwise **pdvl** ≥ 1.

11: **vr**[*dim*] – double *Output*

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

$\max(1, \mathbf{pdvr} \times \mathbf{n})$ when **jobvr** = Nag_RightVecs;
1 otherwise.

Where $\mathbf{VR}(i, j)$ appears in this document, it refers to the array element

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

On exit: if **jobvr** = Nag_RightVecs, the right eigenvectors v_j are stored one after another in **vr**, in the same order as their corresponding eigenvalues. If the jth eigenvalue is real, then $v_j = \mathbf{VR}(i, j)$, for $i = 1, 2, \ldots, \mathbf{n}$. If the jth and $(j+1)$st eigenvalues form a complex conjugate pair, then $v_j = \mathbf{VR}(i, j) + i \times \mathbf{VR}(i, j+1)$ and $v_{j+1} = \mathbf{VR}(i, j) - i \times \mathbf{VR}(i, j+1)$, for $i = 1, 2, \ldots, \mathbf{n}$.

If **jobvr** = Nag_NotRightVecs, **vr** is not referenced.

12: **pdvr** – Integer *Input*

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

Constraints:

if **jobvr** = Nag_RightVecs, **pdvr** $\geq \max(1, \mathbf{n})$;
otherwise **pdvr** ≥ 1.

13: **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_CONVERGENCE

The QR algorithm failed to compute all the eigenvalues, and no eigenvectors have been computed; elements ⟨*value*⟩ to **n** of **wr** and **wi** contain eigenvalues which have converged.

NE_ENUM_INT_2

On entry, **jobvl** = ⟨*value*⟩, **pdvl** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **jobvl** = Nag_LeftVecs, **pdvl** $\geq \max(1, \mathbf{n})$;
otherwise **pdvl** ≥ 1.

On entry, **jobvr** = ⟨*value*⟩, **pdvr** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **jobvr** = Nag_RightVecs, **pdvr** ≥ max(1, **n**);
otherwise **pdvr** ≥ 1.

NE_INT

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

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

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

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

NE_INT_2

On entry, **pda** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **pda** ≥ max(1, **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

The computed eigenvalues and eigenvectors are exact for a nearby matrix $(A + E)$, where

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the ***machine precision***. See Section 4.8 of Anderson *et al.* (1999) for further details.

8 Parallelism and Performance

nag_dgeev (f08nac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dgeev (f08nac) 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

Each eigenvector is normalized to have Euclidean norm equal to unity and the element of largest absolute value real and positive.

The total number of floating-point operations is proportional to n^3.

The complex analogue of this function is nag_zgeev (f08nnc).

10 Example

This example finds all the eigenvalues and right eigenvectors of the matrix

$$A = \begin{pmatrix} 0.35 & 0.45 & -0.14 & -0.17 \\ 0.09 & 0.07 & -0.54 & 0.35 \\ -0.44 & -0.33 & -0.03 & 0.17 \\ 0.25 & -0.32 & -0.13 & 0.11 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_dgeev (f08nac) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

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

int main(void)
{

  /* Scalars */
  Integer       i, j, n, pda, pdvr;
  Integer       exit_status = 0;

  /* Arrays */
  double        *a = 0, *vr = 0, *wi = 0, *wr = 0;
  double         dummy [1];

  /* Nag Types */
  NagError      fail;
  Nag_OrderType order;

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

  INIT_FAIL(fail);

  printf("nag_dgeev (f08nac) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n]", &n);
#else
  scanf("%"NAG_IFMT"%*[^\n]", &n);
#endif
  if (n < 0)
    {
      printf("Invalid n\n");
      exit_status = 1;
      goto END;
    }
```

```
  pda = n;
  pdvr = n;
  /* Allocate memory */
  if (!(a = NAG_ALLOC(n * n, double)) ||
      !(vr = NAG_ALLOC(n * n, double)) ||
      !(wi = NAG_ALLOC(n, double)) ||
      !(wr = NAG_ALLOC(n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read the matrix A from data file */
  for (i = 1; i <= n; ++i)
#ifdef _WIN32
    for (j = 1; j <= n; ++j) scanf_s("%lf", &A(i, j));
#else
    for (j = 1; j <= n; ++j) scanf("%lf", &A(i, j));
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* Compute the eigenvalues and right eigenvectors only of A
   * using nag_dgeev (f08nac).
   */
  nag_dgeev(order, Nag_NotLeftVecs, Nag_RightVecs, n, a, pda, wr, wi,
            dummy, 1, vr, pdvr, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dgeev (f08nac).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Print eigenvalues and right eigenvectors. */
  for (j = 0; j < n; ++j)
    {
      printf("\nEigenvalue %3"NAG_IFMT" = ", j+1);
      if (wi[j] == 0.0)
        printf("%13.4e\n", wr[j]);
      else
        printf(" (%13.4e, %13.4e)\n", wr[j], wi[j]);

      printf("\nEigenvector %2"NAG_IFMT"\n", j+1);
      if (wi[j] == 0.0)
        for (i = 0; i < n; ++i) printf("%17s%13.4e\n", "", VR(i, j));
      else if (wi[j] > 0.0)
        for (i = 0; i < n; ++i)
          printf("%18s(%13.4e, %13.4e)\n", "", VR(i, j), VR(i, j+1));
      else
        for (i = 0; i < n; ++i)
          printf("%18s(%13.4e, %13.4e)\n", "", VR(i, j-1), -VR(i, j));
      printf("\n");
    }

 END:
  NAG_FREE(a);
  NAG_FREE(vr);
  NAG_FREE(wi);
  NAG_FREE(wr);

  return exit_status;
}
#undef A
#undef VR
```

10.2 Program Data

```
nag_dgeev (f08nac) Example Program Data

  4                          : n

  0.35   0.45  -0.14  -0.17
  0.09   0.07  -0.54   0.35
 -0.44  -0.33  -0.03   0.17
  0.25  -0.32  -0.13   0.11 : matrix A
```

10.3 Program Results

```
nag_dgeev (f08nac) Example Program Results

Eigenvalue   1 =    7.9948e-01

Eigenvector  1
                   -6.5509e-01
                   -5.2363e-01
                    5.3622e-01
                   -9.5607e-02


Eigenvalue   2 =  (  -9.9412e-02,    4.0079e-01)

Eigenvector  2
                 (  -1.9330e-01,    2.5463e-01)
                 (   2.5186e-01,   -5.2240e-01)
                 (   9.7182e-02,   -3.0838e-01)
                 (   6.7595e-01,    0.0000e+00)


Eigenvalue   3 =  (  -9.9412e-02,   -4.0079e-01)

Eigenvector  3
                 (  -1.9330e-01,   -2.5463e-01)
                 (   2.5186e-01,    5.2240e-01)
                 (   9.7182e-02,    3.0838e-01)
                 (   6.7595e-01,   -0.0000e+00)


Eigenvalue   4 =   -1.0066e-01

Eigenvector  4
                    1.2533e-01
                    3.3202e-01
                    5.9384e-01
                    7.2209e-01
```