

# 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  $\lambda_j$  is the  $j$ th eigenvalue of  $A$ . The left eigenvector  $u_j$  of  $A$  satisfies

$$u_j^H A = \lambda_j u_j^H$$

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 2.3.1.3 in How to Use the NAG Library and its Documentation 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:*  $n \geq 0$ .

5: **a**[ $dim$ ] – double *Input/Output*

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

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

**a**[ $(j-1) \times pda + i - 1$ ] when **order** = Nag\_ColMajor;  
**a**[ $(i-1) \times pda + j - 1$ ] 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:* **pda**  $\geq \max(1, n)$ .

7: **wr**[ $dim$ ] – double *Output*

8: **wi**[ $dim$ ] – double *Output*

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

*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

$\max(1, pdvl \times n)$  when **jobvl** = Nag\_LeftVecs;  
 1 otherwise.

Where **VL**( $i, j$ ) appears in this document, it refers to the array element

**vl**[ $(j-1) \times pdvl + i - 1$ ] when **order** = Nag\_ColMajor;  
**vl**[ $(i-1) \times pdvl + j - 1$ ] 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  $j$ th eigenvalue is real, then  $u_j = \mathbf{VL}(i, j)$ , for  $i = 1, 2, \dots, n$ . If the  $j$ th 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  $i = 1, 2, \dots, n$ .

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, **n**);  
 otherwise **pdvl**  $\geq$  1.

- 11: **vr**[*dim*] – double *Output*  
**Note:** the dimension, *dim*, of the array **vr** must be at least  
 max(1, **pdvr**  $\times$  **n**) when **jobvr** = Nag\_RightVecs;  
 1 otherwise.

Where **VR**(*i*, *j*) appears in this document, it refers to the array element

**vr**[(*j* – 1)  $\times$  **pdvr** + *i* – 1] when **order** = Nag\_ColMajor;  
**vr**[(*i* – 1)  $\times$  **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 *j*th eigenvalue is real, then  $v_j = \mathbf{VR}(i, j)$ , for  $i = 1, 2, \dots, \mathbf{n}$ . If the *j*th 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  $i = 1, 2, \dots, \mathbf{n}$ .

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, **n**);  
 otherwise **pdvr**  $\geq$  1.

- 13: **fail** – NagError \* *Input/Output*  
 The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation 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** =  $\langle value \rangle$ , **pdvl** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .  
 Constraint: if **jobvl** = Nag\_LeftVecs, **pdvl**  $\geq \max(1, \mathbf{n})$ ;  
 otherwise **pdvl**  $\geq 1$ .

On entry, **jobvr** =  $\langle value \rangle$ , **pdvr** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .  
 Constraint: if **jobvr** = Nag\_RightVecs, **pdvr**  $\geq \max(1, \mathbf{n})$ ;  
 otherwise **pdvr**  $\geq 1$ .

**NE\_INT**

On entry, **n** =  $\langle value \rangle$ .  
 Constraint: **n**  $\geq 0$ .

On entry, **pda** =  $\langle value \rangle$ .  
 Constraint: **pda**  $> 0$ .

On entry, **pdvl** =  $\langle value \rangle$ .  
 Constraint: **pdvl**  $> 0$ .

On entry, **pdvr** =  $\langle value \rangle$ .  
 Constraint: **pdvr**  $> 0$ .

**NE\_INT\_2**

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

**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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly.  
 See Section 2.7.5 in How to Use the NAG Library and its Documentation 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  $\epsilon$  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.

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.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#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
}
```

```

#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

```

---