NAG Library Function Document

nag_zgeev (f08nnc)

1 Purpose

nag_zgeev (f08nnc) computes the eigenvalues and, optionally, the left and/or right eigenvectors for an n by n complex nonsymmetric matrix A.

2 Specification

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

```
void nag_zgeev (Nag_OrderType order, Nag_LeftVecsType jobvl,
     Nag_RightVecsType jobvr, Integer n, Complex a[], Integer pda,
     Complex w[], Complex vl[], Integer pdvl, Complex 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 unitary similarity transformations, and the QR algorithm is then used to further reduce the matrix to upper triangular Schur form, T, from which the eigenvalues are computed. Optionally, the eigenvectors of T are also computed and backtransformed to those 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** ≥ 0.

5: **a**[dim] – Complex *Input/Output*

Note: the dimension, *dim*, of the array **a** must be at least max(1, **pda** × **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: **w**[dim] – Complex *Output*

Note: the dimension, *dim*, of the array **w** must be at least max(1, **n**).

On exit: contains the computed eigenvalues.

8: **vl**[dim] – Complex *Output*

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

max(1, **pdvl** × **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; that is $u_j =$ **VL**(i, j), for $i = 1, 2, \ldots,$ **n**.

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

9: **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** ≥ 1.

10: **vr**[*dim*] – Complex *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

$\mathbf{vr}[(j-1) \times \mathbf{pdvr} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{vr}[(i-1) \times \mathbf{pdvr} + j - 1]$ 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; that is $v_j = \mathbf{VR}(i,j)$, for $i = 1, 2, \ldots, \mathbf{n}$.

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

11: **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, $\mathbf{pdvr} \geq \max(1, \mathbf{n})$;
otherwise $\mathbf{pdvr} \geq 1$.

12: **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 **w** contain eigenvalues which have converged.

NE_ENUM_INT_2

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

On entry, **jobvr** = ⟨*value*⟩, **pdvr** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **jobvr** = Nag_RightVecs, $\mathbf{pdvr} \geq \max(1, \mathbf{n})$;
otherwise $\mathbf{pdvr} \geq 1$.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: $\mathbf{n} \geq 0$.

On entry, **pda** = ⟨*value*⟩.
Constraint: $\mathbf{pda} > 0$.

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

8 Parallelism and Performance

nag_zgeev (f08nnc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zgeev (f08nnc) 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 real analogue of this function is nag_dgeev (f08nac).

10 Example

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

$$A = \begin{pmatrix} -3.97 - 5.04i & -4.11 + 3.70i & -0.34 + 1.01i & 1.29 - 0.86i \\ 0.34 - 1.50i & 1.52 - 0.43i & 1.88 - 5.38i & 3.36 + 0.65i \\ 3.31 - 3.85i & 2.50 + 3.45i & 0.88 - 1.08i & 0.64 - 1.48i \\ -1.10 + 0.82i & 1.81 - 1.59i & 3.25 + 1.33i & 1.57 - 3.44i \end{pmatrix}.$$

10.1 Program Text

```
/* nag_zgeev (f08nnc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <nag.h>
#include <nagx04.h>
#include <nag_stdlib.h>
#include <nagf08.h>

int main(void)
{
  /* Scalars */
  Integer exit_status = 0, i, j, n, pda, pdvr;

  /* Arrays */
  Complex *a = 0, *vr = 0, *w = 0;
  Complex 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_zgeev (f08nnc) 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 "%*[^\n]", &n);
#else
  scanf("%" NAG_IFMT "%*[^\n]", &n);
#endif

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

  /* Read the matrix A from data file */
  for (i = 1; i <= n; ++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

  /* Compute the eigenvalues and right eigenvectors of A
     using nag_zgeev (f08nnc). */
  nag_zgeev(order, Nag_NotLeftVecs, Nag_RightVecs, n, a, pda, w, dummy, 1,
            vr, pdvr, &fail);

  if (fail.code != NE_NOERROR) {
    printf("Error from nag_zgeev (f08nnc).\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 (w[j].im == 0.0)
      printf("%13.4e\n", w[j].re);
    else
      printf(" (%13.4e, %13.4e)\n", w[j].re, w[j].im);

    printf("\nEigenvector %2" NAG_IFMT "\n", j + 1);
    for (i = 0; i < n; ++i)
      printf("%18s(%13.4e, %13.4e)\n", "", VR(i, j).re, VR(i, j).im);
    printf("\n");
  }

END:
  NAG_FREE(a);
  NAG_FREE(vr);
  NAG_FREE(w);

  return exit_status;
}

#undef A
```

10.2 Program Data

```
nag_zgeev (f08nnc) Example Program Data

  4                                                                 : n

(-3.97, -5.04)  (-4.11,  3.70)  (-0.34,  1.01)  ( 1.29, -0.86)
( 0.34, -1.50)  ( 1.52, -0.43)  ( 1.88, -5.38)  ( 3.36,  0.65)
( 3.31, -3.85)  ( 2.50,  3.45)  ( 0.88, -1.08)  ( 0.64, -1.48)
(-1.10,  0.82)  ( 1.81, -1.59)  ( 3.25,  1.33)  ( 1.57, -3.44) : matrix A
```

10.3 Program Results

```
nag_zgeev (f08nnc) Example Program Results


Eigenvalue   1 =  (   -6.0004e+00,   -6.9998e+00)

Eigenvector  1
                  (    8.4572e-01,    0.0000e+00)
                  (   -1.7723e-02,    3.0361e-01)
                  (    8.7521e-02,    3.1145e-01)
                  (   -5.6147e-02,   -2.9060e-01)
```

```
Eigenvalue   2 =  (  -5.0000e+00,    2.0060e+00)

Eigenvector  2
                  (  -3.8655e-01,    1.7323e-01)
                  (  -3.5393e-01,    4.5288e-01)
                  (   6.1237e-01,    0.0000e+00)
                  (  -8.5928e-02,   -3.2836e-01)


Eigenvalue   3 =  (   7.9982e+00,   -9.9637e-01)

Eigenvector  3
                  (  -1.7297e-01,    2.6690e-01)
                  (   6.9242e-01,    0.0000e+00)
                  (   3.3240e-01,    4.9598e-01)
                  (   2.5039e-01,   -1.4655e-02)


Eigenvalue   4 =  (   3.0023e+00,   -3.9998e+00)

Eigenvector  4
                  (  -3.5614e-02,   -1.7822e-01)
                  (   1.2637e-01,    2.6663e-01)
                  (   1.2933e-02,   -2.9657e-01)
                  (   8.8982e-01,    0.0000e+00)
```