NAG Library Function Document

nag_dgeesx (f08pbc)

1 Purpose

nag_dgeesx (f08pbc) computes the eigenvalues, the real Schur form \( T \), and, optionally, the matrix of Schur vectors \( Z \) for an \( n \) by \( n \) real nonsymmetric matrix \( A \).

2 Specification

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

void nag_dgeesx (Nag_OrderType order, Nag_JobType jobvs,
                Nag_SortEigValsType sort,
                Nag_Boolean (*select)(double wr, double wi),
                Nag_RCondType sense, Integer n, double a[], Integer pda,
                Integer *sdim, double wr[], double wi[], double vs[], Integer pdvs,
                double *rconde, double *rcondv, NagError *fail)
```

3 Description

The real Schur factorization of \( A \) is given by

\[
A = Z T Z^T,
\]

where \( Z \), the matrix of Schur vectors, is orthogonal and \( T \) is the real Schur form. A matrix is in real Schur form if it is upper quasi-triangular with 1 by 1 and 2 by 2 blocks. 2 by 2 blocks will be standardized in the form

\[
\begin{bmatrix}
  a & b \\
  c & a
\end{bmatrix}
\]

where \( bc < 0 \). The eigenvalues of such a block are \( a \pm \sqrt{bc} \).

Optionally, nag_dgeesx (f08pbc) also orders the eigenvalues on the diagonal of the real Schur form so that selected eigenvalues are at the top left; computes a reciprocal condition number for the average of the selected eigenvalues (\( rconde \)); and computes a reciprocal condition number for the right invariant subspace corresponding to the selected eigenvalues (\( rcondv \)). The leading columns of \( Z \) form an orthonormal basis for this invariant subspace.

For further explanation of the reciprocal condition numbers \( rconde \) and \( rcondv \), see Section 4.8 of Anderson et al. (1999) (where these quantities are called \( s \) and \( sep \) respectively).

4 References


5 Arguments

1: \textbf{order} \textemdash Nag\_OrderType \textemdash \textit{Input}

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

\textit{Constraint}: \textit{order} = Nag\_RowMajor or Nag\_ColMajor.

2: \textbf{jobvs} \textemdash Nag\_JobType \textemdash \textit{Input}

\textit{On entry}: if \textit{jobvs} = Nag\_DoNothing, Schur vectors are not computed.

If \textit{jobvs} = Nag\_Schur, Schur vectors are computed.

\textit{Constraint}: \textit{jobvs} = Nag\_DoNothing or Nag\_Schur.

3: \textbf{sort} \textemdash Nag\_SortEigValsType \textemdash \textit{Input}

\textit{On entry}: specifies whether or not to order the eigenvalues on the diagonal of the Schur form.

\textit{sort} = Nag\_NoSortEigVals

\hspace{1em} Eigenvalues are not ordered.

\textit{sort} = Nag\_SortEigVals

\hspace{1em} Eigenvalues are ordered (see \textit{select}).

\textit{Constraint}: \textit{sort} = Nag\_NoSortEigVals or Nag\_SortEigVals.

4: \textbf{select} \textemdash \text{function, supplied by the user} \textit{External Function}

If \textit{sort} = Nag\_SortEigVals, \textit{select} is used to select eigenvalues to sort to the top left of the Schur form.

If \textit{sort} = Nag\_NoSortEigVals, \textit{select} is not referenced and nag\_dgeesx (f08pbc) may be specified as NULLFN.

An eigenvalue \(w_r[j-1] + \sqrt{-1} \times w_i[j-1]\) is selected if \textit{select}(\(w_r[j-1], w_i[j-1]\)) is Nag\_TRUE. If either one of a complex conjugate pair of eigenvalues is selected, then both are. \textit{Note that a selected complex eigenvalue may no longer satisfy} \textit{select}(\(w_r[j-1], w_i[j-1]\)) = Nag\_TRUE after ordering, since ordering may change the value of complex eigenvalues (especially if the eigenvalue is ill-conditioned); in this case \textit{fail.errnum} is set to \(n + 2\).

The specification of \textit{select} is:

\begin{itemize}
  \item \text{Nag\_Boolean select (double wr, double wi)}
  \item \text{1: wr \textendash double \textit{Input}}
  \item \text{2: wi \textendash double \textit{Input}}
\end{itemize}

\textit{On entry}: the real and imaginary parts of the eigenvalue.

5: \textbf{sense} \textemdash Nag\_RCondType \textemdash \textit{Input}

\textit{On entry}: determines which reciprocal condition numbers are computed.

\textit{sense} = Nag\_NotRCond

\hspace{1em} None are computed.

\textit{sense} = Nag\_RCondEigVals

\hspace{1em} Computed for average of selected eigenvalues only.

\textit{sense} = Nag\_RCondEigVees

\hspace{1em} Computed for selected right invariant subspace only.
sense = Nag_RCondBoth  
  Computed for both.

If sense = Nag_RCondEigVals, Nag_RCondEigVecs or Nag_RCondBoth, sort = Nag_SortEigVals.

Constraint: sense = Nag_NotRCond, Nag_RCondEigVals, Nag_RCondEigVecs or Nag_RCondBoth.

6:  n – Integer  
  Input
  On entry: \( n \), the order of the matrix \( A \).
  Constraint: \( n \geq 0 \).

7:  a[\( dim \)] – double  
  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
  \[
  a[(j - 1) \times pda + i - 1] \quad \text{when } order = \text{Nag}_\text{ColMajor}; \\
  a[(i - 1) \times pda + j - 1] \quad \text{when } order = \text{Nag}_\text{RowMajor}.
  \]
  On entry: the \( n \) by \( n \) matrix \( A \).
  On exit: \( a \) is overwritten by its real Schur form \( T \).

8:  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 \text{max}(1, n) \).

9:  sdim – Integer *  
  Output
  On exit: if sort = Nag_NoSortEigVals, sdim = 0.
  If sort = Nag_SortEigVals, sdim = number of eigenvalues (after sorting) for which select is Nag_TRUE. (Complex conjugate pairs for which select is Nag_TRUE for either eigenvalue count as 2.)

10:  wr[\( dim \)] – double  
  Output
  Note: the dimension, \( dim \), of the array \( wr \) must be at least max(1, n).
  On exit: see the description of \( wi \).

11:  wi[\( dim \)] – double  
  Output
  Note: the dimension, \( dim \), of the array \( wi \) must be at least max(1, n).
  On exit: \( wr \) and \( wi \) contain the real and imaginary parts, respectively, of the computed eigenvalues in the same order that they appear on the diagonal of the output Schur form \( T \). Complex conjugate pairs of eigenvalues will appear consecutively with the eigenvalue having the positive imaginary part first.

12:  vs[\( dim \)] – double  
  Output
  Note: the dimension, \( dim \), of the array \( vs \) must be at least
  \[
  \text{max}(1, pdvs \times n) \quad \text{when } jobvs = \text{Nag}_\text{Schur}; \\
  1 \quad \text{otherwise}.
  \]
  The \( i \)th element of the \( j \)th vector is stored in
  \[
  vs[(j - 1) \times pdvs + i - 1] \quad \text{when } order = \text{Nag}_\text{ColMajor}; \\
  vs[(i - 1) \times pdvs + j - 1] \quad \text{when } order = \text{Nag}_\text{RowMajor}.
  \]
On exit: if jobvs = Nag_Schur, vs contains the orthogonal matrix Z of Schur vectors.
If jobvs = Nag_DoNothing, vs is not referenced.

13: pdvs – Integer
   On entry: the stride used in the array vs.
   Constraints:
   if jobvs = Nag_Schur, pdvs ≥ max(1, n);
   otherwise pdvs ≥ 1.

14: rconde – double *
   On exit: if sense = Nag_RCondEigVals or Nag_RCondBoth, contains the reciprocal condition number for the average of the selected eigenvalues.
   If sense = Nag_NotRCond or Nag_RCondEigVecs, rconde is not referenced.

15: rcondv – double *
   On exit: if sense = Nag_RCondEigVecs or Nag_RCondBoth, rcondv contains the reciprocal condition number for the selected right invariant subspace.
   If sense = Nag_NotRCond or Nag_RCondEigVals, rcondv is not referenced.

16: fail – NagError *
   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.

NE_ENUM_INT_2
   On entry, jobvs = ⟨value⟩, pdvs = ⟨value⟩ and n = ⟨value⟩.
   Constraint: if jobvs = Nag_Schur, pdvs ≥ max(1, n);
   otherwise pdvs ≥ 1.

NE_INT
   On entry, n = ⟨value⟩.
   Constraint: n ≥ 0.
   On entry, pda = ⟨value⟩.
   Constraint: pda > 0.
   On entry, pdvs = ⟨value⟩.
   Constraint: pdvs > 0.
NE_INT_2
On entry, pda = \langle value \rangle and n = \langle value \rangle.
Constraint: pda \geq \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.

NE_SCHUR_REORDER
The eigenvalues could not be reordered because some eigenvalues were too close to separate (the problem is very ill-conditioned).

NE_SCHUR_REORDER_SELECT
After reordering, roundoff changed values of some complex eigenvalues so that leading eigenvalues in the Schur form no longer satisfy select = Nag_TRUE. This could also be caused by underflow due to scaling.

7 Accuracy
The computed Schur factorization satisfies
\[ A + E = ZTZ^T, \]
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_dgeesx (f08pbc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dgeesx (f08pbc) 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 is proportional to \( n^3 \).
The complex analogue of this function is nag_zgeesx (f08ppc).
10 Example

This example finds the Schur factorization of the matrix

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

such that the real positive eigenvalues of \( A \) are the top left diagonal elements of the Schur form, \( T \). Estimates of the condition numbers for the selected eigenvalue cluster and corresponding invariant subspace are also returned.

10.1 Program Text

/* nag_dgeesx (f08pbc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* * Mark 25, 2014. */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagf16.h>
#include <nagx02.h>
#include <nagx04.h>

#ifdef __cplusplus
extern "C" {
#endif
static Nag_Boolean NAG_CALL select_fun(const double wr, const double wi);
#ifdef __cplusplus
}
#endif

int main(void)
{

  /* Scalars */
  double alpha, anorm, beta, eps, norm, rconde, rcondv;
  Integer i, j, n, pda, pdc, pdd, pdvs, sdim;
  Integer exit_status = 0;

  /* Arrays */
  double *a = 0, *c = 0, *d = 0, *vs = 0, *wi = 0, *wr = 0;

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

  #ifdef NAG_COLUMN_MAJOR
  #define A(I, J) a[(J-1)*pda +I-1]
  #else
  #define A(I, J) a[(I-1)*pda+J-1]
  #endif
  INIT_FAIL(fail);
  printf("nag_dgeesx (f08pbc) Example Program Results\n\n");

  /* Skip heading in data file */
  #ifdef _WIN32
  scanf_s("%*[^
");
  #endif

  ...
#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;
    return exit_status;
}
pda = n;
pdc = n;
pdd = n;
pdvs = n;
/* Allocate memory */
if (!(a = NAG_ALLOC(n * n, double)) ||
    !(c = NAG_ALLOC(n * n, double)) ||
    !(d = NAG_ALLOC(n * n, double)) ||
    !(vs = 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 in the matrix A */
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
/* Copy A to D: nag_dge_copy (f16qfc),
   real valued general matrix copy. */
    nag_dge_copy(order, Nag_NoTrans, n, n, a, pda, d, pdd, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dge_copy (f16qfc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* nag_dge_norm (f16rac): Find norm of matrix A for use later
   in relative error test. */
    nag_dge_norm(order, Nag_OneNorm, n, n, a, pda, &anorm, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dge_norm (f16rac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* nag_gen_real_mat_print (x04cac): Print Matrix A. */
    fflush(stdout);
    nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_nonUnitDiag, n, n, a,
                           "Matrix A", 0, &fail);
    printf("\n");
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Find the Schur factorization of A using nag_dgeesx (f08pbc). */
if (fail.code != NE_NOERROR) {
    printf("Error from nag_dgeesx (f08pbc).\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Reconstruct A from Schur Factorization Z*T*Trans(Z) where T is upper triangular and stored in A. This can be done using the following steps:
* i. C = Z*T (nag_dgemm, f16yac),
* ii. D = D-C*trans(Z) (nag_dgemm, f16yac).
*/
alpha = 1.0;
beta = 0.0;
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dgemm (f16yac).\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_dge_norm (f16rac): Find norm of difference matrix D and print warning if it is too large relative to norm of A. */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dge_norm (f16rac).\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Get the machine precision, using nag_machine_precision (x02ajc) */
eps = nag_machine_precision;
if (norm > pow(eps,0.8)*MAX(anorm,1.0))
{
    printf("||A-(Z*T*Z\'T)||/||A|| is larger than expected.\n   \n   Schur factorization has failed.\n\n   exit_status = 1;
   goto END;
}

/* Print details on eigenvalues */
printf("Number of eigenvalues for which select is true = %4"NAG_IFMT"\n\n", f08pbc
if (fail.code == NE_SCHUR_REORDER_SELECT) {
    printf(" ** Note that rounding errors mean that leading eigenvalues in the"
" Schur form\n no longer satisfy select(lambda) = Nag_TRUE\n\n");
} else {
    printf("The selected eigenvalues are:\n\n");
    for (i=0;i<sdim;i++)
        printf("%3"NAG_IFMT" (%13.4e, %13.4e)\n", i+1, wr[i], wi[i]);
}

/* Print out the reciprocal condition numbers */
printf("\nReciprocal of projection norm onto the invariant subspace\n\n"%26sfor the selected eigenvalues rconde = %8.1e\n\n", "Reciprocal condition number for the invariant subspace rcondv = "
"%8.1e\n\n", rconde);

END:
NAG_FREE(a);
NAG_FREE(c);
NAG_FREE(d);
NAG_FREE(vs);
NAG_FREE(wi);
NAG_FREE(wr);
return exit_status;
}

static Nag_Boolean NAG_CALL select_fun(const double ar, const double ai)
{
    /* Boolean function select for use with nag_dgees (f08pac)
    * Returns the value Nag_TRUE if the eigenvalue is real and positive
    */
    return (ar>0.0 && ai==0.0 ? Nag_TRUE : Nag_FALSE);
}

10.2 Program Data

nag_dgeesx (f08pbc) 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_dgeesx (f08pbc) Example Program Results

Matrix A
     1     2     3     4
  1  0.3500  0.4500 -0.1400 -0.1700
  2  0.0900  0.0700 -0.5400  0.3500
  3 -0.4400 -0.3300 -0.0300  0.1700
  4  0.2500 -0.3200 -0.1300  0.1100
Number of eigenvalues for which select is true = 1

The selected eigenvalues are:
1 ( 7.9948e-01, 0.0000e+00)

Reciprocal of projection norm onto the invariant subspace
for the selected eigenvalues rconde = 9.9e-01

Reciprocal condition number for the invariant subspace rcondv = 8.2e-01

Approximate asymptotic error bound for selected eigenvalues = 1.3e-16

Approximate asymptotic error bound for the invariant subspace = 1.6e-16