nag_real_general_eigensystem (f02bjc)

1. Purpose

nag_real_general_eigensystem (f02bjc) calculates all the eigenvalues and, if required, all the eigenvectors of the generalized eigenproblem \( Ax = \lambda Bx \) where \( A \) and \( B \) are real, square matrices, using the QZ algorithm.

2. Specification

```c
#include <nag.h>
#include <nagf02.h>

void nag_real_general_eigensystem(Integer n, double a[], Integer tda,
                                  double b[], Integer tdb, double tol, Complex alfa[],
                                  double beta[], Boolean wantv, double v[], Integer tdv,
                                  Integer iter[], NagError *fail)
```

3. Description

All the eigenvalues and, if required, all the eigenvectors of the generalized eigenproblem \( Ax = \lambda Bx \) where \( A \) and \( B \) are real, square matrices, are determined using the QZ algorithm. The QZ algorithm consists of four stages:

(a) \( A \) is reduced to upper Hessenberg form and at the same time \( B \) is reduced to upper triangular form.

(b) \( A \) is further reduced to quasi-triangular form while the triangular form of \( B \) is maintained.

(c) The quasi-triangular form of \( A \) is reduced to triangular form and the eigenvalues extracted.

This function does not actually produce the eigenvalues \( \lambda_j \), but instead returns \( \alpha_j \) and \( \beta_j \) such that

\[
\lambda_j = \alpha_j/\beta_j, \quad j = 1, 2, \ldots, n.
\]

The division by \( \beta_j \) becomes the responsibility of the user’s program, since \( \beta_j \) may be zero indicating an infinite eigenvalue. Pairs of complex eigenvalues occur with \( \alpha_j/\beta_j \) and \( \alpha_{j+1}/\beta_{j+1} \) complex conjugates, even though \( \alpha_j \) and \( \alpha_{j+1} \) are not conjugate.

(d) If the eigenvectors are required (\( \text{wantv} = \text{TRUE} \)), they are obtained from the triangular matrices and then transformed back into the original co-ordinate system.

4. Parameters

\( n \)

Input: \( n \), the order of the matrices \( A \) and \( B \).

Constraint: \( n \geq 1 \).

\( a[n][tda] \)

Input: the \( n \) by \( n \) matrix \( A \).

Output: the array is overwritten.

\( tda \)

Input: the second dimension of the array \( a \) as declared in the function from which nag_real_general_eigensystem is called.

Constraint: \( tda \geq n \).

\( b[n][tdb] \)

Input: the \( n \) by \( n \) matrix \( B \).

Output: the array is overwritten.

\( tdb \)

Input: the second dimension of the array \( b \) as declared in the function from which nag_real_general_eigensystem is called.

Constraint: \( tdb \geq n \).
tol
Input: the tolerance used to determine negligible elements. If \( \text{tol} > 0.0 \), an element will be considered negligible if it is less than \( \text{tol} \) times the norm of its matrix. If \( \text{tol} \leq 0.0 \), \textbf{machine precision} is used in place of \( \text{tol} \). A value of \( \text{tol} \) greater than \textbf{machine precision} may result in faster execution but less accurate results.

\text{alfa}[n]
Output: \( \alpha_j \), for \( j = 1, 2, \ldots, n \).

\text{beta}[n]
Output: \( \beta_j \), for \( j = 1, 2, \ldots, n \).

wantv
Input: \texttt{wantv} must be set to \texttt{TRUE} if the eigenvectors are required. If \texttt{wantv} is set to \texttt{FALSE} then the array \( \text{v} \) is not referenced.

\text{v}[n][tdv]
Output: if \texttt{wantv} = \texttt{TRUE}, then
(i) if the \texttt{j}th eigenvalue is real, the \texttt{j}th column of \( \text{v} \) contains its eigenvector;
(ii) if the \texttt{j}th and \((\texttt{j} + 1)\)th eigenvalues form a complex pair, the \texttt{j}th and \((\texttt{j} + 1)\)th columns of \( \text{v} \) contain the real and imaginary parts of the eigenvector associated with the first eigenvalue of the pair. The conjugate of this vector is the eigenvector for the conjugate eigenvalue.

Each eigenvector is normalised so that the component of largest modulus is real and the sum of squares of the moduli equal one.

If \texttt{wantv} = \texttt{FALSE}, \( \text{v} \) is not referenced and may be set to the null pointer, i.e., (double *)0.

\text{tdv}
Input: the second dimension of the array \( \text{v} \) as declared in the function from which \texttt{nag_real_general_eigensystem} is called.
Constraint: \( \text{tdv} \geq n \) if \texttt{wantv} = \texttt{TRUE}.

\text{iter}[n]
Output: \texttt{iter}[j − 1] contains the number of iterations needed to obtain the \texttt{j}th eigenvalue. Note that the eigenvalues are obtained in reverse order, starting with the \texttt{n}th.

fail
The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings
\texttt{NE_INT_ARG_LT}
On entry, \texttt{n} must not be less than 1: \texttt{n} = \langle value\rangle.

\texttt{NE_2_INT_ARG_LT}
On entry \texttt{tda} = \langle value\rangle while \texttt{n} = \langle value\rangle. These parameters must satisfy \texttt{tda} \geq \texttt{n}.
On entry \texttt{tdb} = \langle value\rangle while \texttt{n} = \langle value\rangle. These parameters must satisfy \texttt{tdb} \geq \texttt{n}.
On entry \texttt{tdv} = \langle value\rangle while \texttt{n} = \langle value\rangle. These parameters must satisfy \texttt{tdv} \geq \texttt{n}.

\texttt{NE_ITERATIONS_QZ}
More than \texttt{n} × 30 iterations are required to determine all the diagonal 1 by 1 or 2 by 2 blocks of the quasi-triangular form in the second step of the \texttt{QZ} algorithm. This failure occurs at the \texttt{i}th eigenvalue, \( i = \langle value\rangle \). \( \alpha_j \) and \( \beta_j \) are correct for \( j = i + 1, i + 2, \ldots, n \) but \( \text{v} \) does not contain any correct eigenvectors.

The value of \( i \) will be returned in member errnum of the NAG error structure provided \texttt{NAGERR_DEFAULT} is not used as the error parameter.

6. Further Comments
The time taken by the function is approximately proportional to \( n^3 \) and also depends on the value chosen for parameter \texttt{tol}.
6.1. Accuracy

The computed eigenvalues are always exact for a problem \((A + E)x = \lambda(B + F)x\) where \(\|E\|/\|A\|\) and \(\|F\|/\|B\|\) are both of the order of \(\max(\text{tol}, \varepsilon)\), \text{tol} being defined as in Section 4 and \(\varepsilon\) being the machine precision.

\textbf{Note:} interpretation of results obtained with the \textit{QZ} algorithm often requires a clear understanding of the effects of small changes in the original data. These effects are reviewed in Wilkinson (1979), in relation to the significance of small values of \(\alpha_j\) and \(\beta_j\). It should be noted that if \(\alpha_j\) and \(\beta_j\) are both small for any \(j\), it may be that no reliance can be placed on any of the computed eigenvalues \(\lambda_i = \alpha_i/\beta_i\). The user is recommended to study Wilkinson (1979) and, if in difficulty, to seek expert advice on determining the sensitivity of the eigenvalues to perturbations in the data.

6.2. References


7. See Also

None

8. Example

To find all the eigenvalues and eigenvectors of \(Ax = \lambda Bx\) where

\[
A = \begin{pmatrix}
3.9 & 4.3 & 4.3 & 4.4 \\
12.5 & 21.5 & 21.5 & 26.0 \\
-34.5 & -47.5 & -43.5 & -46.0 \\
-0.5 & 7.5 & 3.5 & 6.0
\end{pmatrix}
\]

and

\[
B = \begin{pmatrix}
1 & 1 & 1 & 1 \\
2 & 3 & 3 & 3 \\
-3 & -5 & -4 & -4 \\
1 & 4 & 3 & 4
\end{pmatrix}
\]

8.1. Program Text

\[
/* nag_real_general_eigensystem(f02bjc) Example Program */
/* Copyright 1991 Numerical Algorithms Group. */
/* * Mark 2, 1991. */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <math.h>
#include <nagf02.h>
#include <nagx02.h>
#define NMAX 8
#define TDA NMAX
#define TDB NMAX
#define TDZ NMAX

main()
{
    Integer i, j, k, n, ip, iter[NMAX];
    Complex alfa[NMAX];
}
double beta[NMAX], eps1;
double a[NMAX][TDA], b[NMAX][TDB], z[NMAX][TDZ];
Boolean matz;

Vprintf("f02bjc Example Program Results\n");
Vscanf("%*[\n"];
/* Skip heading in data file */
Vscanf("%ld", &n);
if (n>0 && n<=NMAX)
{
for (i=0; i<n; ++i)
  for (j=0; j<n; ++j)
    Vscanf("%lf", &a[i][j]);
for (i=0; i<n; ++i)
  for (j=0; j<n; ++j)
    Vscanf("%lf", &b[i][j]);
matz = TRUE;
eps1 = X02AJC;
f02bjc(n, (double *)a, (Integer)TDA, (double *)b, (Integer)TDB, eps1,
      alfa, beta, matz, (double *)z, (Integer)TDZ, iter,
      NAGERR_DEFAULT);

ip = 0;
for (i=0; i<n; ++i)
{
  if (i>0) i--; ++i)
    Vprintf("Eigensolution %4ld\n",i+1);
  Vprintf("alfa[%ld].re %7.3f",i,alfa[i].re);
  Vprintf("alfa[%ld].im %7.3f",i,alfa[i].im);
  Vprintf(" beta[%ld] %7.3f\n",i,beta[i]);
  if (beta[i] == 0.0)
    Vprintf("lambda is infinite");
else
  if (alfa[i].im == 0.0)
    { 
    Vprintf("lambda %7.3f\n",alfa[i].re/beta[i]);
    Vprintf("Eigenvector\n");
    for (j=0; j<n; ++j)
      Vprintf("%7.3f\n", z[j][i]);
    }
  else
    { 
    Vprintf("lambda %7.3f %7.3f", 
      alfa[i].re/beta[i], alfa[i].im/beta[i] );
    Vprintf("Eigenvector\n");
    k = (Integer)pow((double)-1, (double)(ip+2));
    for (j=0; j<n; ++j)
      { 
        Vprintf("%7.3f",z[j][i-ip]);
        Vprintf("%7.3f\n",k*Z[j][i-ip+1]);
      }
    ip = 1-ip;
    }
  Vprintf("Number of iterations (machine-dependent)\n");
  for (i=0; i<n; ++i)
    Vprintf("%2ld",iter[i]);
  Vprintf("\n");
  exit(EXIT_SUCCESS);
}
else
{ 
  Vprintf(stderr,"n is out of range: n = %4ld,\n",n);
  exit(EXIT_FAILURE);
}
8.2. Program Data

f02bjc Example Program Data

\[
\begin{array}{cccc}
3.9 & 12.5 & -34.5 & -0.5 \\
4.3 & 21.5 & -47.5 & 7.5 \\
4.4 & 26.0 & -46.0 & 6.0 \\
1.0 & 2.0 & -3.0 & 1.0 \\
1.0 & 3.0 & -5.0 & 4.0 \\
1.0 & 3.0 & -4.0 & 3.0 \\
1.0 & 3.0 & -4.0 & 4.0 \\
\end{array}
\]

8.3. Program Results

f02bjc Example Program Results

Eigensolution 1
\[\alpha[0].re = 3.801 \quad \alpha[0].im = 0.000 \quad \beta[0] = 1.900 \quad \lambda = 2.000\]
Eigenvector
\[0.996 \quad 0.006 \quad 0.063 \quad 0.063\]

Eigensolution 2
\[\alpha[1].re = 1.563 \quad \alpha[1].im = 2.084 \quad \beta[1] = 0.521 \quad \lambda = 3.000 \quad 4.000\]
Eigenvector
\[0.945 \quad 0.000 \quad 0.189 \quad 0.000 \quad 0.113 \quad -0.151 \quad 0.113 \quad -0.151\]

Eigensolution 3
\[\alpha[2].re = 3.030 \quad \alpha[2].im = -4.040 \quad \beta[2] = 1.010 \quad \lambda = 3.000 \quad -4.000\]
Eigenvector
\[0.945 \quad 0.000 \quad 0.189 \quad 0.000 \quad 0.113 \quad 0.151 \quad 0.113 \quad 0.151\]

Eigensolution 4
\[\alpha[3].re = 4.000 \quad \alpha[3].im = 0.000 \quad \beta[3] = 1.000 \quad \lambda = 4.000\]
Eigenvector
\[0.988 \quad 0.011 \quad -0.033 \quad 0.154\]

Number of iterations (machine-dependent)
\[0 \quad 0 \quad 5 \quad 0\]