NAG Toolbox for MATLAB

f02gj

1 Purpose

f02gj calculates all the eigenvalues and, if required, all the eigenvectors of the complex generalized eigenproblem $Ax = \lambda Bx$ where A and B are complex, square matrices, using the QZ algorithm.

2 Syntax

```
[ar, ai, br, bi, alfr, alfi, beta, vr, vi, iter, ifail] = f02gj(ar, ai,
br, bi, eps1, matv, 'n', n)
```

3 Description

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

1. A is reduced to upper Hessenberg form (with real, nonnegative subdiagonal elements) and at the same time B is reduced to upper triangular form.
2. A is further reduced to triangular form while the triangular form of B is maintained and the diagonal elements of B are made real and nonnegative.

   f02gj does not actually produce the eigenvalues λ_j, but instead returns α_j and β_j such that

   $$\lambda_j = \alpha_j / \beta_j, \qquad j = 1, 2, \ldots, n.$$

   The division by β_j becomes the responsibility of your program, since β_j may be zero, indicating an infinite eigenvalue.
3. If the eigenvectors are required (**matv** = **true**), they are obtained from the triangular matrices and then transferred back into the original co-ordinate system.

4 References

Moler C B and Stewart G W 1973 An algorithm for generalized matrix eigenproblems *SIAM J. Numer. Anal.* **10** 241–256

Ward R C 1975 The combination shift *QZ* algorithm *SIAM J. Numer. Anal.* **12** 835–853

Wilkinson J H 1979 Kronecker's canonical form and the *QZ* algorithm *Linear Algebra Appl.* **28** 285–303

5 Parameters

5.1 Compulsory Input Parameters

1: **ar(ldar,n) – double array**

ldar, the first dimension of the array, must be at least **n**.

The real parts of the elements of the n by n complex matrix A.

2: **ai(ldai,n) – double array**

ldai, the first dimension of the array, must be at least **n**.

The imaginary parts of the elements of the n by n complex matrix A.

3: **br(ldbr,n) – double array**

ldbr, the first dimension of the array, must be at least **n**.

The real parts of the elements of the n by n complex matrix B.

4: **bi(ldbi,n) – double array**

ldbi, the first dimension of the array, must be at least **n**.

The imaginary parts of the elements of the n by n complex matrix B.

5: **eps1 – double scalar**

A tolerance used to determine negligible elements.

eps1 > 0.0

An element will be considered negligible if it is less than **eps1** times the norm of its matrix.

eps1 ≤ 0.0

machine precision is used for **eps1**.

A positive value of **eps1** may result in faster execution but less accurate results.

6: **matv – logical scalar**

Must be set **true** if the eigenvectors are required, otherwise **false**.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **ar**, **ai**, **br**, **bi**, **alfr**, **alfi**, **beta**, **vr**, **vi**, **iter**. (An error is raised if these dimensions are not equal.)

n, the order of the matrices A and B.

Constraint: $\mathbf{n} \geq 1$.

5.3 Input Parameters Omitted from the MATLAB Interface

ldar, ldai, ldbr, ldbi, ldvr, ldvi

5.4 Output Parameters

1: **ar(ldar,n) – double array**

The array is overwritten.

2: **ai(ldai,n) – double array**

The array is overwritten.

3: **br(ldbr,n) – double array**

The array is overwritten.

4: **bi(ldbi,n) – double array**

The array is overwritten.

5: **alfr(n) – double array**
6: **alfi(n) – double array**

The real and imaginary parts of α_j, for $j = 1, 2, \ldots, n$.

7: **beta(n) – double array**

β_j, for $j = 1, 2, \ldots, n$.

8: **vr(ldvr,n) – double array**

If **matv** = **true**, the jth column of **vr** contains the real parts of the eigenvector corresponding to the jth eigenvalue. The eigenvectors are normalized so that the sum of squares of the moduli of the components is equal to 1.0 and the component of largest modulus is real.

If **matv** = **false**, **vr** is not used.

9: **vi(ldvi,n) – double array**

If **matv** = **true**, the jth column of **vi** contains the imaginary parts of the eigenvector corresponding to the jth eigenvalue.

If **matv** = **false**, **vi** is not used.

10: **iter(n) – int32 array**

iter(j) contains the number of iterations needed to obtain the jth eigenvalue. Note that the eigenvalues are obtained in reverse order, starting with the nth.

11: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= i$

More than $30 \times$ **n** iterations have been performed altogether in the second step of the *QZ* algorithm; **ifail** is set to the index i of the eigenvalue at which the failure occurs. On soft failure, α_j and β_j are correct for $j = i + 1, i + 2, \ldots, n$, but the arrays **vr** and **vi** do not contain any correct eigenvectors.

7 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 $\max(\mathbf{eps1}, \epsilon)$, **eps1** being defined as in Section 5 and ϵ being the ***machine precision***.

Note: interpretation of results obtained with the *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 α_j and β_j. It should be noted that if α_j and β_j are **both** small for any j, it may be that no reliance can be placed on **any** of the computed eigenvalues $\lambda_i = \alpha_i/\beta_i$. You are 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.

8 Further Comments

The time taken by f02gj is approximately proportional to n^3 and also depends on the value chosen for parameter **eps1**.

9 Example

```
ar = [-21.1, 53.5, -34.5, 7.5;
```

```
     -0.46, -3.5, -15.5, -10.5;
     4.3, 39.7, -68.5, -7.5;
     5.5, 14.4, -32.5, -19];
ai = [-22.5, -50.5, 127.5, 0.5;
     -7.78, -37.5, 58.5, -1.5;
     -5.5, -17.1, 12.5, -3.5;
     4.4, 43.3, -46, -32.5];
br = [1, 1.6, -3, 0;
     0.8, 3, -4, -2.4;
     1, 2.4, -4, 0;
     0, -1.8, 0, 4];
bi = [-5, 1.2, 0, -1;
     -0.6, -5, 3, -3.2;
     0, 1.8, -5, -3;
     1, 2.4, -4, -5];
eps1 = 1.111307226797642e-16;
matv = true;
[arOut, aiOut, brOut, biOut, alfr, alfi, beta, vr, vi, iter, ifail] =
f02gj(ar, ai, br, bi, eps1, matv)
```

```
arOut =
   19.0329   94.0531  -55.7774   41.0012
         0   11.8818   -3.6587    1.3068
         0         0   10.9609   23.8168
         0         0         0   21.8722
aiOut =
  -57.0986   45.7511   86.5965  108.1735
         0  -29.7045  -27.6162  -16.3832
         0         0   -3.6536   11.2315
         0         0         0  -27.3403
brOut =
    6.3443    0.6432    1.1695    2.1117
         0    5.9409   -1.1487   -1.9021
         0         0    3.6536    0.0295
    0.0000         0         0    5.4681
biOut =
         0   -3.2210    3.1860    6.8785
         0         0    0.1900    1.0432
         0         0         0   -0.0222
         0         0         0         0
alfr =
   19.0329
   11.8818
   10.9609
   21.8722
alfi =
  -57.0986
  -29.7045
   -3.6536
  -27.3403
beta =
    6.3443
    5.9409
    3.6536
    5.4681
vr =
    0.9449    0.9961    0.9449    0.9875
    0.1512    0.0046    0.1512    0.0088
    0.1134    0.0626    0.1134   -0.0329
   -0.1512    0.0000    0.1512    0.0000
vi =
         0         0         0         0
   -0.1134   -0.0034   -0.1134   -0.0066
    0.1512   -0.0000   -0.1512    0.0000
    0.1134    0.0626    0.1134    0.1536
iter =
           0
           1
           5
           0
```

```
ifail =
           0
```