NAG Toolbox for MATLAB

f02sd

1 Purpose

f02sd finds the eigenvector corresponding to a given real eigenvalue for the generalized problem $Ax = \lambda Bx$, or for the standard problem $Ax = \lambda x$, where A and B are real band matrices.

2 Syntax

```
[a, b, vec, d, ifail] = f02sd(ma1, mb1, a, b, sym, relep, rmu, d, 'n',
n)
```

3 Description

Given an approximation μ to a real eigenvalue λ of the generalized eigenproblem $Ax = \lambda Bx$, this function attempts to compute the corresponding eigenvector by inverse iteration.

The function first computes lower and upper triangular factors, L and U, of $A - \mu B$, using Gaussian elimination with interchanges, and then solves the equation $Ux = e$, where $e = (1, 1, 1, \ldots, 1)^{\mathrm{T}}$ – this is the first half iteration.

There are then three possible courses of action depending on the input value of $\mathbf{d}(1)$.

1. $\mathbf{d}(1) = 0$.

   This setting should be used if λ is an ill-conditioned eigenvalue (provided the matrix elements do not vary widely in order of magnitude). In this case it is essential to accept only a vector found after one half iteration, and μ must be a very good approximation to λ. If acceptable growth is achieved in the solution of $Ux = e$, then the normalized x is accepted as the eigenvector. If not, columns of an orthogonal matrix are tried in turn in place of e. If none of these give acceptable growth, the function fails, indicating that μ was not a sufficiently good approximation to λ.

2. $\mathbf{d}(1) > 0$.

   This setting should be used if μ is moderately close to an eigenvalue which is not ill-conditioned (provided the matrix elements do not differ widely in order of magnitude). If acceptable growth is achieved in the solution of $Ux = e$, the normalized x is accepted as the eigenvector. If not, inverse iteration is performed. Up to 30 iterations are allowed to achieve a vector and a correction to μ which together give acceptably small residuals.

3. $\mathbf{d}(1) < 0$.

   This setting should be used if the elements of A and B vary widely in order of magnitude. Inverse iteration is performed, but a different convergence criterion is used.

See Section 8.3 for further details.

Note that the bandwidth of the matrix A must not be less than the bandwidth of B. If this is not so, either A must be filled out with zeros, or matrices A and B may be reversed and $1/\mu$ supplied as an approximation to the eigenvalue $1/\lambda$. Also it is assumed that A and B each have the same number of subdiagonals as superdiagonals. If this is not so, they must be filled out with zeros. If A and B are **both** symmetric, only the upper triangles need be supplied.

4 References

Peters G and Wilkinson J H 1979 Inverse iteration, ill-conditioned equations and Newton's method *SIAM Rev.* **21** 339–360

Wilkinson J H 1965 *The Algebraic Eigenvalue Problem* Oxford University Press, Oxford

Wilkinson J H 1972 Inverse iteration in theory and practice *Symposia Mathematica Volume X* 361–379 Istituto Nazionale di Alta Matematica, Monograf, Bologna

Wilkinson J H 1974 Notes on inverse iteration and ill-conditioned eigensystems *Acta Univ. Carolin. Math. Phys.* **1–2** 173–177

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: **ma1 – int32 scalar**

The value $m_A + 1$, where m_A is the number of nonzero lines on each side of the diagonal of A. Thus the total bandwidth of A is $2m_A + 1$.

Constraint: $1 \leq$ **ma1** $\leq$ **n**.

2: **mb1 – int32 scalar**

If **mb1** ≤ 0, B is assumed to be the unit matrix. Otherwise **mb1** must specify the value $m_B + 1$, where m_B is the number of nonzero lines on each side of the diagonal of B. Thus the total bandwidth of B is $2m_B + 1$.

Constraint: **mb1** $\leq$ **ma1**.

3: **a(lda,n) – double array**

lda, the first dimension of the array, must be at least $2 \times$ **ma1** $- 1$.

The n by n band matrix A. The m_A subdiagonals must be stored in the first m_A rows of the array; the diagonal in the $(m_A + 1)$th row; and the m_A superdiagonals in rows $m_A + 2$ to $2m_A + 1$. Each row of the matrix must be stored in the corresponding column of the array. For example, if $n = 6$ and $m_A = 2$ the storage scheme is:

$$\begin{array}{cccccc} * & * & a_{31} & a_{42} & a_{53} & a_{64} \\ * & a_{21} & a_{32} & a_{43} & a_{54} & a_{65} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \\ a_{12} & a_{23} & a_{34} & a_{45} & a_{56} & * \\ a_{13} & a_{24} & a_{35} & a_{46} & * & * \end{array}.$$

Elements of the array marked $*$ need not be set. The following code assigns the matrix elements within the band to the correct elements of the array:

```
for j=1:n
  for i=max(1,j-ma1+1):min(n,j+ma1-1)
    a(i-j+ma1,j) = matrix(i,j);
  end
end
```

If **sym** = **true** (see below) (i.e., both A and B are symmetric), only the lower triangle of A need be stored in the first **ma1** rows of the array.

4: **b(ldb,n) – double array**

ldb, the first dimension of the array, must be at least

if **sym** = **false**, **ldb** $\geq 2 \times$ **mb1** $- 1$;
if **sym** = **true**, **ldb** $\geq$ **mb1**.

.

If **mb1** > 0, **b** must contain the n by n band matrix B, stored in the same way as A. If **sym** = **true**, only the lower triangle of B need be stored in the first **mb1** rows of the array.

If **mb1** ≤ 0, the array is not used.

5: **sym – logical scalar**

If **sym** = **true**, both A and B are assumed to be symmetric and only their upper triangles need be stored. Otherwise **sym** must be set to **false**.

6: **relep – double scalar**

The relative error of the coefficients of the given matrices A and B. If the value of **relep** is less than the ***machine precision***, the ***machine precision*** is used instead.

7: **rmu – double scalar**

μ, an approximation to the eigenvalue for which the corresponding eigenvector is required.

8: **d(30) – double array**

d(1) must be set to indicate the type of problem (see Section 3):

d$(1) > 0.0$

Indicates a well-conditioned eigenvalue.

d$(1) = 0.0$

Indicates an ill-conditioned eigenvalue.

d$(1) < 0.0$

Indicates that the matrices have elements varying widely in order of magnitude.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **a**, **b**, **vec**. (An error is raised if these dimensions are not equal.)

n, the order of the matrices A and B.

Constraint: **n** ≥ 1.

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldb, iwork, work, lwork

5.4 Output Parameters

1: **a(lda,n) – double array**

Details of the factorization of $A - \bar{\lambda}B$, where $\bar{\lambda}$ is an estimate of the eigenvalue.

2: **b(ldb,n) – double array**

Elements in the top-left corner, and in the bottom right corner if **sym** = **false**, are set to zero; otherwise the array is unchanged.

3: **vec(n) – double array**

The eigenvector, normalized so that the largest element is unity, corresponding to the improved eigenvalue **rmu** + **d**(30).

4: **d(30) – double array**

If **d**$(1) \neq 0.0$ on entry, the successive corrections to μ are given in **d**(i), for $i = 1, 2, \ldots, k$, where $k + 1$ is the total number of iterations performed. The final correction is also given in the last position, **d**(30), of the array. The remaining elements of **d** are set to zero.

If $\mathbf{d}(1) = 0.0$ on entry, no corrections to μ are computed and $\mathbf{d}(i)$ is set to 0.0, for $i = 1, 2, \ldots, 30$. Thus in all three cases the best available approximation to the eigenvalue is $\mathbf{rmu} + \mathbf{d}(30)$.

5: **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:

$\mathbf{ifail} = 1$

On entry, $\mathbf{n} < 1$,
or $\mathbf{ma1} < 1$,
or $\mathbf{ma1} > \mathbf{n}$,
or $\mathbf{lda} < 2 \times \mathbf{ma1} - 1$,
or $\mathbf{ldb} < \mathbf{mb1}$ when $\mathbf{sym} = \mathbf{true}$,
or $\mathbf{ldb} < 2 \times \mathbf{mb1} - 1$ when $\mathbf{sym} = \mathbf{false}$ (**ldb** is not checked if $\mathbf{mb1} \leq 0$).

$\mathbf{ifail} = 2$

On entry, $\mathbf{ma1} < \mathbf{mb1}$. Either fill out **a** with zeros, or reverse the roles of **a** and **b**, and replace **rmu** by its reciprocal, i.e., solve $Bx = \lambda^{-1}Ax$.

$\mathbf{ifail} = 3$

On entry, $\mathbf{lwork} < 2 \times \mathbf{n}$ when $\mathbf{d}(1) = 0.0$,
or $\mathbf{lwork} < \mathbf{n} \times (\mathbf{ma1} + 1)$ when $\mathbf{d}(1) \neq 0.0$.

$\mathbf{ifail} = 4$

A is null. If B is nonsingular, all the eigenvalues are zero and any set of **n** orthogonal vectors forms the eigensolution.

$\mathbf{ifail} = 5$

B is null. If A is nonsingular, all the eigenvalues are infinite, and the columns of the unit matrix are eigenvectors.

$\mathbf{ifail} = 6$

On entry, A and B are both null. The eigensolution is arbitrary.

$\mathbf{ifail} = 7$

$\mathbf{d}(1) \neq 0.0$ on entry and convergence is not achieved in 30 iterations. Either the eigenvalue is ill-conditioned or **rmu** is a poor approximation to the eigenvalue. See Section 8.3.

$\mathbf{ifail} = 8$

$\mathbf{d}(1) = 0.0$ on entry and no eigenvector has been found after $\min(\mathbf{n}, 5)$ back-substitutions. **rmu** is not a sufficiently good approximation to the eigenvalue.

$\mathbf{ifail} = 9$

$\mathbf{d}(1) < 0.0$ on entry and **rmu** is too inaccurate for the solution to converge.

7 Accuracy

The eigensolution is exact for some problem

$$(A+E)x = \mu(B+F)x,$$

where $\|E\|, \|F\|$ are of the order of $\eta(\|A\| + \mu\|B\|)$, where η is the value used for **relep**.

8 Further Comments

8.1 Timing

The time taken by f02sd is approximately proportional to $n(2m_A+1)^2$ for factorization, and to $n(2m_A+1)$ for each iteration.

8.2 Storage

The storage of the matrices A and B is designed for efficiency on a paged machine.

This function will work with full matrices but it will do so inefficiently, particularly in respect of storage requirements.

8.3 Algorithmic Details

Inverse iteration is performed according to the rule

$$(A-\mu B)y_{r+1} = Bx_r$$

$$x_{r+1} = \frac{1}{\alpha_{r+1}} y_{r+1}$$

where α_{r+1} is the element of y_{r+1} of largest magnitude.

Thus:

$$(A-\mu B)x_{r+1} = \frac{1}{\alpha_{r+1}} Bx_r.$$

Hence the residual corresponding to x_{r+1} is very small if $|\alpha_{r+1}|$ is very large (see Peters and Wilkinson 1979). The first half iteration, $Uy_1 = e$, corresponds to taking $L^{-1}PBx_0 = e$.

If μ is a very accurate eigenvalue, then there should always be an initial vector x_0 such that one half iteration gives a small residual and thus a good eigenvector. If the eigenvalue is ill-conditioned, then second and subsequent iterated vectors may not be even remotely close to an eigenvector of a neighbouring problem (see pages 374–376 of Wilkinson 1972 and Wilkinson 1974). In this case it is essential to accept only a vector obtained after one half iteration.

However, for well-conditioned eigenvalues, there is no loss in performing more than one iteration (see page 376 of Wilkinson 1972), and indeed it will be necessary to iterate if μ is not such a good approximation to the eigenvalue. When the iteration has converged, y_{r+1} will be some multiple of x_r, $y_{r+1} = \beta_{r+1}x_r$, say.

Therefore

$$(A-\mu B)\beta_{r+1}x_r = Bx_r,$$

giving

$$\left(A - \left(\mu + \frac{1}{\beta_{r+1}}\right)B\right)x_r = 0.$$

Thus $\mu + \frac{1}{\beta_{r+1}}$ is a better approximation to the eigenvalue. β_{r+1} is obtained as the element of y_{r+1} which corresponds to the element of largest magnitude, $+1$, in x_r. The function terminates when

$\left\| \left(A - \left(\mu + \frac{1}{\beta_r} \right) B \right) x_r \right\|$ is of the order of the ***machine precision*** relative to $\|A\| + |\mu| \|B\|$.

If the elements of A and B vary widely in order of magnitude, then $\|A\|$ and $\|B\|$ are excessively large and a different convergence test is required. The function terminates when the difference between successive corrections to μ is small relative to μ.

In practice one does not necessarily know if the given problem is well-conditioned or ill-conditioned. In order to provide some information on the condition of the eigenvalue or the accuracy of μ in the event of failure, successive values of $\frac{1}{\beta_r}$ are stored in the vector **d** when $\mathbf{d}(1)$ is nonzero on input. If these values appear to be converging steadily, then it is likely that μ was a poor approximation to the eigenvalue and it is worth trying again with $\mathbf{rmu} + \mathbf{d}(30)$ as the initial approximation. If the values in **d** vary considerably in magnitude, then the eigenvalue is ill-conditioned.

A discussion of the significance of the singularity of A and/or B is given in relation to the QZ algorithm in Wilkinson 1979.

9 Example

```
ma1 = int32(3);
mb1 = int32(2);
a = [0, 0, 0, 0, 0;
     0, -1, -1, -1, -1;
     1, 2, 3, 4, 5;
     1, 1, 1, 1, 0;
     2, 2, 2, 0, 0];
b = [0, 1, 2, 2, 1;
     5, 4, 3, 2, 1;
     1, 2, 2, 1, 0];
sym = false;
relep = 0;
rmu = -12.33;
d = zeros(30, 1);
d(1) = 1;
[aOut, bOut, vec, dOut, ifail] = f02sd(ma1, mb1, a, b, sym, relep, rmu,
d)
```

```
aOut =
    0.0160    0.0204    0.0423    0.0883   68.1366
   13.3300   25.2983   28.6600   17.3300         0
    2.0000    2.0000   13.3300         0         0
         0         0         0         0         0
         0         0         0         0         0
bOut =
     0     1     2     2     1
     5     4     3     2     1
     1     2     2     1     0
vec =
   -0.0572
    0.3951
   -0.8427
    1.0000
   -0.6540
dOut =
   -0.0094
   -0.0094
   -0.0094
   -0.0094
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
   -0.0094
ifail =
           0
```