NAG Toolbox for MATLAB

d02ka

1 Purpose

d02ka finds a specified eigenvalue of a regular second-order Sturm–Liouville system defined on a finite range, using a Pruefer transformation and a shooting method.

2 Syntax

```
[bcond, elam, delam, ifail] = d02ka(xl, xr, coeffn, bcond, k, tol, elam, delam, monit)
```

3 Description

d02ka finds a specified eigenvalue $\tilde{\lambda}$ of a Sturm–Liouville system defined by a self-adjoint differential equation of the second-order

$$\left(p(x)y'\right)' + q(x;\lambda)y = 0, \qquad a < x < b$$

together with boundary conditions of the form

$$a_2 y(a) = a_1 p(a) y'(a)$$

$$b_2 y(b) = b_1 p(b) y'(b)$$

at the two, finite, end points a and b. The functions p and q, which are real-valued, are defined by user-supplied (sub)program **coeffn**.

For the theoretical basis of the numerical method to be valid, the following conditions should hold on the coefficient functions:

(a) $p(x)$ must be nonzero and of constant sign throughout the closed interval $[a, b]$;

(b) $\dfrac{\partial q}{\partial \lambda}$ must be of constant sign and nonzero throughout the open interval (a, b) and for all relevant values of λ, and must not be identically zero as x varies, for any relevant value λ;

(c) p and q should (as functions of x) have continuous derivatives, preferably up to the fourth-order, on $[a, b]$. The differential equation code used will integrate through mild discontinuities, but probably with severely reduced efficiency. Therefore, if p and q violate this condition, d02kd should be used.

The eigenvalue $\tilde{\lambda}$ is determined by a shooting method based on a Pruefer transformation of the differential equations. Providing certain assumptions are met, the computed value of $\tilde{\lambda}$ will be correct to within a mixed absolute/relative error specified by **tol**. d02ka is a driver function for the more complicated function d02kd whose specification provides more details of the techniques used.

A good account of the Sturm–Liouville systems, with some description of Pruefer transformations, is given in Chapter X of Birkhoff and Rota 1962. The best introduction to the use of Pruefer transformations for the numerical solution of eigenvalue problems arising from physics and chemistry is given in Bailey 1966.

4 References

Bailey P B 1966 Sturm–Liouville eigenvalues via a phase function *SIAM J. Appl. Math.* **14** 242–249

Birkhoff G and Rota G C 1962 *Ordinary Differential Equations* Ginn & Co., Boston and New York

5 Parameters

5.1 Compulsory Input Parameters

1: **xl – double scalar**

2: **xr – double scalar**

The left- and right-hand end points a and b respectively, of the interval of definition of the problem.

Constraint: **xl** $<$ **xr**.

3: **coeffn – string containing name of m-file**

coeffn must compute the values of the coefficient functions $p(x)$ and $q(x;\lambda)$ for given values of x and λ. Section 3 gives the conditions which p and q must satisfy.

Its specification is:

```
[p, q, dqdl] = coeffn(x, elam, jint)
```

Input Parameters

1: **x – double scalar**

The current value of x.

2: **elam – double scalar**

The current trial value of the eigenvalue parameter λ.

3: **jint – int32 scalar**

This parameter is included for compatibility with the more complex function d02kd (which is called by d02ka).

Need not be set.

Output Parameters

1: **p – double scalar**

The value of $p(x)$ for the current value of x.

2: **q – double scalar**

The value of $q(x;\lambda)$ for the current value of x and the current trial value of λ.

3: **dqdl – double scalar**

The value of $\dfrac{\partial q}{\partial \lambda}(x;\lambda)$ for the current value of x and the current trial value of λ. However **dqdl** is only used in error estimation and, in the rare cases where it may be difficult to evaluate, an approximation (say to within 20%) will suffice.

4: **bcond(3,2) – double array**

bcond$(1,1)$ and **bcond**$(2,1)$ must contain the numbers a_1, a_2 specifying the left-hand boundary condition in the form

$$a_2 y(a) = a_1 p(a) y'(a)$$

where $|a_2| + |a_1 p(a)| \neq 0$.

bcond$(1,2)$ and **bcond**$(2,2)$ must contain b_1, b_2 such that

$$b_2 y(b) = b_1 p(b) y'(b)$$

where $|b_2| + |b_1 p(b)| \neq 0$.

Note the occurrence of $p(a)$, $p(b)$ in these formulae.

5: **k – int32 scalar**

The index, k, of the desired eigenvalue when these are ordered:

$$\lambda_0 < \lambda_1 < \cdots < \lambda_k \cdots .$$

Constraint: **k** ≥ 0.

6: **tol – double scalar**

The tolerance parameter which determines the accuracy of the computed eigenvalue. The error estimate held in **delam** on exit will satisfy the 'mixed absolute/relative error test'

$$\textbf{delam} \leq \textbf{tol} \times \max(1.0, |\textbf{elam}|), \tag{1}$$

where **elam** has its exit value; **delam** will usually be somewhat smaller than the right-hand side of (1) but not several orders of magnitude smaller.

Constraint: **tol** > 0.0.

7: **elam – double scalar**

An initial estimate of the eigenvalue.

8: **delam – double scalar**

An indication of the scale of the problem in the λ-direction. **delam** holds the initial 'search step' (positive or negative). Its value is not critical, but the first two trial evaluations are made at **elam** and **elam** + **delam**, so the function will work most efficiently if the eigenvalue lies between these values. A reasonable choice (if a closer bound is not known) is about half the distance between adjacent eigenvalues in the neighbourhood of the one sought. Often there will be a problem, similar to the one in hand but with known eigenvalues, which will help one to choose initial values for **elam** and **delam**.

If **delam** $= 0.0$ on entry, it is given the default value of $0.25 \times \max(1.0, |\textbf{elam}|)$.

9: **monit – string containing name of m-file**

monit is called by d02ka at the end of each iteration for λ and allows you to monitor the course of the computation by printing out the parameters (see Section 9 for an example). **The parameters must not be altered by the function.**

If no monitoring is required, the dummy (sub)program **d02kay** may be used. (**d02kay** is included in the NAG Fortran Library.

Its specification is:

```
[] = monit(nit, iflag, elam, finfo)
```

Input Parameters

1: **nit – int32 scalar**

15 minus the number of iterations used so far in the search for $\tilde{\lambda}$. (Up to 15 iterations are permitted.)

2: **iflag – int32 scalar**

Describes what phase the computation is in.

iflag < 0

An error occurred during the computation at this iteration; an error exit from d02ka will follow.

iflag $= 1$

The function is trying to bracket the eigenvalue $\tilde{\lambda}$.

iflag $= 2$

The function is converging to the eigenvalue $\tilde{\lambda}$ (having already bracketed it).

Normally, the iteration will terminate after a sequence of iterates with **iflag** $= 2$, but occasionally the bracket on $\tilde{\lambda}$ thus determined will not be sufficiently small and the iteration will be repeated with tighter accuracy control.

3: **elam – double scalar**

The current trial value of $\tilde{\lambda}$.

4: **finfo**$(\mathbf{15})$ **– double array**

Information about the behaviour of the shooting method, and diagnostic information in the case of errors. It should not normally be printed in full if no error has occurred (that is, if **iflag** ≥ 0), though the first few components may be of interest to you. In case of an error (**iflag** < 0) all the components of **finfo** should be printed.

The contents of **finfo** are as follows:

finfo(1)

The current value of the 'miss-distance' or 'residual' function $f(\lambda)$ on which the shooting method is based. $f\left(\tilde{\lambda}\right) = 0$ in theory. This is set to zero if **iflag** < 0.

finfo(2)

An estimate of the quantity $\delta\lambda$ defined as follows. Consider the perturbation in the miss-distance $f(\lambda)$ that would result if the local error in the solution of the differential equation were always positive and equal to its maximum permitted value. Then $\delta\lambda$ is the perturbation in λ that would have the same effect on $f(\lambda)$. Thus, at the zero of $f(\lambda), |\delta\lambda|$ is an approximate bound on the perturbation of the zero (that is the eigenvalue) caused by errors in numerical solution. If $\delta\lambda$ is very large then it is possible there has been a programming error in user-supplied (sub)program **coeffn** such that q is independent of λ. If this is the case, an error exit with **ifail** $= 5$ should follow. **finfo**(2) is set to zero if **iflag** < 0.

finfo(3)

The number of internal iterations, using the same value of λ and tighter accuracy tolerances, needed to bring the accuracy (that is, the value of $\delta\lambda$) to an acceptable value. Its value should normally be 1.0, and should almost never exceed 2.0.

finfo(4)

The number of calls to user-supplied (sub)program **coeffn** at this iteration.

finfo(5)

The number of successful steps taken by the internal differential equation solver at this iteration.

finfo(6)

The number of unsuccessful steps used by the internal integrator at this iteration.

finfo(7)

The number of successful steps at the maximum step size taken by the internal integrator at this iteration.

finfo(8)

Not used.

finfo(9) to **finfo**(15)

Set to zero, unless **iflag** < 0 in which case they hold the following values describing the point of failure:

finfo(9)

1 or 2 depending on whether integration was in a forward or backward direction at the time of failure.

finfo(10)

The value of the independent variable, x, the point at which an error occurred.

finfo(11), **finfo**(12), **finfo**(13)

The current values of the Pruefer dependent variables β, ϕ and ρ respectively. See Section 3 of the document for d02ke for a description of these variables.

finfo(14)

The local-error tolerance being used by the internal integrator at the point of failure.

finfo(15)

The last integration mesh point.

Output Parameters

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **bcond**(**3,2**) **– double array**

bcond$(3,1)$ and **bcond**$(3,2)$ hold values σ_l, σ_r estimating the sensitivity of the computed eigenvalue to changes in the boundary conditions. These values should only be of interest if the boundary conditions are, in some sense, an approximation to some 'true' boundary conditions. For example, if the range [**xl**, **xr**] should really be $[0, \infty]$ but instead **xr** has been given a large value and the boundary conditions at infinity applied at **xr**, then the sensitivity parameter σ_r may be of interest. Refer to Section 8.5 of the document for d02kd, for the actual meaning of σ_r and σ_l.

2: **elam – double scalar**

The final computed estimate, whether or not an error occurred.

3: **delam – double scalar**

If **ifail** $= 0$, **delam** holds an estimate of the absolute error in the computed eigenvalue, that is $\left|\tilde{\lambda} - \textbf{elam}\right| \simeq \textbf{delam}$, where $\tilde{\lambda}$ is the true eigenvalue.

With **ifail** $\neq 0$, **delam** may hold an estimate of the error, or its initial value, depending on the value of **ifail**. See Section 6 for further details.

4: **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 $= 1$

On entry, **k** < 0,
or **tol** ≤ 0.

ifail $= 2$

On entry, $a_1 = p(a)a_2 = 0$,
or $b_1 = p(b)b_2 = 0$,

(the array **bcond** has been set up incorrectly).

ifail $= 3$

At some point between **xl** and **xr** the value of $p(x)$ computed by user-supplied (sub)program **coeffn** became zero or changed sign. See the last call of (sub)program **monit** for details.

ifail $= 4$

After 15 iterations the eigenvalue had not been found to the required accuracy.

ifail $= 5$

The bracketing phase (with parameter **iflag** of the (sub)program **monit** equal to 1) failed to bracket the eigenvalue within ten iterations. This is caused by an error in formulating the problem (for example, q is independent of λ), or by very poor initial estimates of **elam**, **delam**.

On exit **elam** and **elam** + **delam** give the end points of the interval within which no eigenvalue was located by the function.

ifail $= 6$

To obtain the desired accuracy the local error tolerance was set so small at the start of some sub-interval that the differential equation solver could not choose an initial step size large enough to make significant progress. See the last call of (sub)program **monit** for diagnostics.

ifail $= 7$

At some point the step size in the differential equation solver was reduced to a value too small to make significant progress (for the same reasons as with **ifail** $= 6$). This could be due to pathological behaviour of $p(x)$ and $q(x;\lambda)$ or to an unreasonable accuracy requirement or to the current value of λ making the equation 'stiff'. See the last call of (sub)program **monit** for details.

ifail $= 8$

tol is too small for the problem being solved and the ***machine precision*** being used. The local value of **elam** should be a very good approximation to the eigenvalue $\tilde{\lambda}$.

ifail = 9

c05az, called by d02ka, has terminated with the error exit corresponding to a pole of the matching function. This error exit should not occur, but if it does, try solving the problem again with a smaller value for **tol**.

Note: error exits with **ifail** = 2, 3, 6, 7 or 9 are caused by the inability to set up or solve the differential equation at some iteration and will be immediately preceded by a call of (sub)program **monit**, giving diagnostic information.

ifail = 10 (**d02kdy**)
ifail = 11 (c05az)
ifail = 12 (d02kd)

A serious error has occurred in an internal call to an interpolation function. Check all (sub)program calls and array dimensions. Seek expert help.

7 Accuracy

The absolute accuracy of the computed eigenvalue is usually within a mixed absolute/relative bound defined by **tol** (as defined above).

8 Further Comments

The time taken by d02ka depends on the complexity of the coefficient functions, whether they or their derivatives are rapidly changing, the tolerance demanded, and how many iterations are needed to obtain convergence. The amount of work per iteration is roughly doubled when **tol** is divided by 16. To make the most economical use of the function, one should try to obtain good initial values for **elam** and **delam**.

See Section 8 of the document for d02kd for a discussion of the technique used.

9 Example

d02ka_coeffn.m

```
function [p, q, dqdl] = coeffn(x, elam, jint)
  p = 1;
  dqdl = 1;
  q = elam - 10*cos(2.0*x);
```

d02ka_monit.m

```
function monit(nit, iflag, elam, finfo)
  if (nit == 14)
    fprintf('\nOutput from Monit\n');
  end

  fprintf('%2d   %d   %6.3f %+6.3f %+6.3g %+6.3f %+6.3f\n', ...
     nit, iflag, elam, finfo(1), finfo(2), finfo(3), finfo(4));
```

```
xl = 0;
xr = 3.141592653589793;
bcond = [1, 1;
     0, 0;
     0, -0.04564094560555453];
k = int32(4);
tol = 1e-05;
elam = 15;
delam = 4;
[bcondOut, elamOut, delamOut, ifail] = ...
```

```
        d02ka(xl,  xr,  'd02ka_coeffn',  bcond,  k,  tol,  elam,  delam,
'd02ka_monit')
```

```
Output from Monit
14   1   15.000 -0.322 -0.000108 +1.000 +206.000
13   1   15.000 -0.322 -5.67e-05 +2.000 +234.000
12   1   19.000 +0.257 -6.69e-05 +1.000 +226.000
11   2   17.225 +0.018 -6.75e-05 +1.000 +226.000
10   2   17.089 -0.001 -6.43e-05 +1.000 +226.000
 9   2   17.097 +0.000 -6.41e-05 +1.000 +226.000
 8   2   17.097 -0.000 -6.41e-05 +1.000 +226.000
bcondOut =
    1.0000    1.0000
         0         0
   -0.9064    0.9064
elamOut =
   17.0966
delamOut =
   1.0685e-04
ifail =
           0
```