NAG Toolbox

nag_ode_sl2_reg_finite (d02ka)

1 Purpose

nag_ode_sl2_reg_finite (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] = nag_ode_sl2_reg_finite(xl, xr, coeffn, bcond, k, tol, elam, delam, monit)
[bcond, elam, delam, ifail] = d02ka(xl, xr, coeffn, bcond, k, tol, elam, delam, monit)
```

3 Description

nag_ode_sl2_reg_finite (d02ka) finds a specified eigenvalue $\tilde{\lambda}$ of a Sturm–Liouville system defined by a self-adjoint differential equation of the second-order

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

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 **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 must not change sign throughout the closed interval $[a, b]$;

(b) $\dfrac{\partial q}{\partial \lambda}$ must not change sign and must be 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; and,

(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, nag_ode_sl2_breaks_vals (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**. nag_ode_sl2_reg_finite (d02ka) is a driver function for the more complicated function nag_ode_sl2_breaks_vals (d02kd) whose specification provides more details of the techniques used.

A good account of the theory of Sturm–Liouville systems, with some description of Pruefer transformations, is given in Chapter X of Birkhoff and Rota (1962). An 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** – REAL (KIND=nag_wp)

2: **xr** – REAL (KIND=nag_wp)

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

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

3: **coeffn** – SUBROUTINE, supplied by the user.

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

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

Input Parameters

1: **x** – REAL (KIND=nag_wp)

The current value of x.

2: **elam** – REAL (KIND=nag_wp)

The current trial value of the eigenvalue argument λ.

3: **jint** – INTEGER

This argument is included for compatibility with the more complex function nag_ode_sl2_breaks_vals (d02kd) (which is called by nag_ode_sl2_reg_finite (d02ka)).

Need not be set.

Output Parameters

1: **p** – REAL (KIND=nag_wp)

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

2: **q** – REAL (KIND=nag_wp)

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

3: **dqdl** – REAL (KIND=nag_wp)

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**$(\mathbf{3},\mathbf{2})$ – REAL (KIND=nag_wp) 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** – INTEGER

k, the index of the required eigenvalue when the eigenvalues are ordered

$$\lambda_0 < \lambda_1 < \lambda_2 < \cdots < \lambda_k < \cdots.$$

Constraint: **k** ≥ 0.

6: **tol** – REAL (KIND=nag_wp)

The tolerance argument which determines the accuracy of the computed eigenvalue. The error estimate held in **delam** on exit satisfies the mixed absolute/relative error test

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

where **elam** is the final estimate of the eigenvalue. **delam** is usually somewhat smaller than the right-hand side of (1) but not several orders of magnitude smaller.

Constraint: **tol** > 0.0.

7: **elam** – REAL (KIND=nag_wp)

An initial estimate of the eigenvalue $\tilde{\lambda}$.

8: **delam** – REAL (KIND=nag_wp)

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. In practice, there will often 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** – SUBROUTINE, supplied by the NAG Library or the user.

monit is called by nag_ode_sl2_reg_finite (d02ka) at the end of each iteration for λ and allows you to monitor the course of the computation by printing out the arguments (see Section 10 for an example).

If no monitoring is required, the dummy (sub)program nag_ode_sl2_reg_finite_dummy_monit (d02kay) may be used. (nag_ode_sl2_reg_finite_dummy_monit (d02kay) is included in the NAG Toolbox.)

```
monit(nit, iflag, elam, finfo)
```

Input Parameters

1: **nit** – INTEGER

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

2: **iflag** – INTEGER

Describes what phase the computation is in.

iflag < 0
An error occurred in the computation at this iteration; an error exit from nag_ode_sl2_reg_finite (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** – REAL (KIND=nag_wp)

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

4: **finfo**(**15**) – REAL (KIND=nag_wp) 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. $f(\tilde{\lambda}) = 0$ in theory. This is set to zero if **iflag** < 0.

finfo(2)
An estimate of the quantity $\partial\lambda$ 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 $\partial\lambda$ is the perturbation in λ that would have the same effect on $f(\lambda)$. Thus, at the zero of $f(\lambda)$, $|\partial\lambda|$ is an approximate bound on the perturbation of the zero (that is the eigenvalue) caused by errors in numerical solution. If $\partial\lambda$ is very large then it is possible that there has been a programming error in **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 $\partial\lambda$) 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 **coeffn** at this iteration.

finfo(5)
The number of successful steps taken by the internal differential equation solver at this iteration. A step is successful if it is used to advance the integration.

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 the error occurred.

finfo(11), **finfo**(12), **finfo**(13)
The current values of the Pruefer dependent variables β, ϕ and ρ respectively. See Section 3 in nag_ode_sl2_breaks_funs (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.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **bcond**(**3**, **2**) – REAL (KIND=nag_wp) 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 argument σ_r may be of interest. Refer to Section 9.5 in nag_ode_sl2_breaks_vals (d02kd), for the actual meaning of σ_r and σ_l.

2: **elam** – REAL (KIND=nag_wp)

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

3: **delam** – REAL (KIND=nag_wp)

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.

If **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** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **k** < 0,
or **tol** ≤ 0.0.

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 **coeffn** became zero or changed sign. See the last call of **monit** for details.

ifail $= 4$

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

ifail $= 5$

The 'bracketing' phase (with **iflag** of the **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** and **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 subinterval that the differential equation solver could not choose an initial step size large enough to make significant progress. See the last call of **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 equations 'stiff'. See the last call of **monit** for details.

ifail $= 8$ (*warning*)

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 $= 9$

nag_roots_contfn_brent_rcomm (c05az), called by nag_ode_sl2_reg_finite (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 **monit** giving diagnostic information.

ifail $= 10$
ifail $= 11$
ifail $= 12$

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

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

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 nag_ode_sl2_reg_finite (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 9 in nag_ode_sl2_breaks_vals (d02kd) for a discussion of the technique used.

9 Example

This example finds the fourth eigenvalue of Mathieu's equations

$$y'' + (\lambda - 2q\cos 2x)y = 0$$

with boundary conditions

$$y'(0) = y'(\pi) = 0$$

and $q = 5$. We use a starting value **elam** $= 15.0$ and a step **delam** $= 4.0$. We illustrate the effect of varying **tol** by choosing **tol** $= 1.0\text{e}{-5}$ and $1.0\text{e}{-6}$ (note the change in the output value of the error estimate **delam**). The range of integration and initial estimates are read from a data file.

9.1 Program Text

```
function d02ka_example

fprintf('d02ka example results\n\n');

xl = 0;
xr = pi;
bcond = [1, 1;
         0, 0;
         0, 0];
k = nag_int(4);
```

```
elam_in = 15;
delam_in = 4;

for tol_ind=-5:-1:-6
  tol = 10^tol_ind;

  [bcond, elam, delam, ifail] = ...
    d02ka(...
           xl, xr, @coeffn, bcond, k, tol, elam_in, delam_in, @monit);

  fprintf('\nCalculation with tol = %12.4e\n\n',tol);
  fprintf('Final results\n\n');
  fprintf('K = %3d  elam = %7.3f  delam = %8.2e\n', k, elam, delam);
  fprintf('Boundary sensitivities, left: %12.4e, right: %12.4e\n',...
          bcond(3,1:2))
end

function [p, q, dqdl] = coeffn(x, elam, jint)
  p = 1;
  dqdl = 1;
  q = elam - 10*cos(2*x);

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(1:4));
```

9.2 Program Results

```
     d02ka example results

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   17.097 +0.000 -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

Calculation with tol =   1.0000e-05

Final results

K =   4  elam =  17.097  delam = 1.50e-04
Boundary sensitivities, left:  -9.0635e-01, right:   9.0635e-01

Output from Monit
14   1   15.000 -0.322 -0.000108 +1.000 +206.000
13   1   15.000 -0.322 -5.64e-06 +2.000 +410.000
12   1   19.000 +0.257 -6.75e-06 +1.000 +406.000
11   2   17.225 +0.018 -6.75e-06 +1.000 +394.000
10   2   17.097 +0.000 -6.43e-06 +1.000 +394.000
 9   2   17.097 +0.000 -6.41e-06 +1.000 +394.000
 8   2   17.097 -0.000 -6.41e-06 +1.000 +394.000
 7   2   17.097 +0.000 -6.41e-06 +1.000 +394.000

Calculation with tol =   1.0000e-06

Final results

K =   4  elam =  17.097  delam = 1.50e-05
Boundary sensitivities, left:  -9.0753e-01, right:   9.0753e-01
```