NAG Library Routine Document

D02KAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

D02KAF 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 Specification

```
SUBROUTINE D02KAF (XL, XR, COEFFN, BCOND, K, TOL, ELAM, DELAM, MONIT,      &
                   IFAIL)

INTEGER             K, IFAIL
REAL (KIND=nag_wp) XL, XR, BCOND(3,2), TOL, ELAM, DELAM
EXTERNAL            COEFFN, MONIT
```

3 Description

D02KAF 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, D02KDF 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. D02KAF is a driver routine for the more complicated routine D02KDF 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 Arguments

1: XL – REAL (KIND=nag_wp) *Input*

2: XR – REAL (KIND=nag_wp) *Input*

On entry: 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. *External Procedure*

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.

The specification of COEFFN is:

```
SUBROUTINE COEFFN (P, Q, DQDL, X, ELAM, JINT)
INTEGER             JINT
REAL (KIND=nag_wp) P, Q, DQDL, X, ELAM
```

1: P – REAL (KIND=nag_wp) *Output*

On exit: the value of $p(x)$ for the current value of x.

2: Q – REAL (KIND=nag_wp) *Output*

On exit: 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) *Output*

On exit: 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: X – REAL (KIND=nag_wp) *Input*

On entry: the current value of x.

5: ELAM – REAL (KIND=nag_wp) *Input*

On entry: the current trial value of the eigenvalue argument λ.

6: JINT – INTEGER *Input*

This argument is included for compatibility with the more complex routine D02KDF (which is called by D02KAF).

On entry: need not be set.

COEFFN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D02KAF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

4: BCOND(3, 2) – REAL (KIND=nag_wp) array *Input/Output*

On entry: 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.

On exit: 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 D02KDF, for the actual meaning of σ_r and σ_l.

5: K – INTEGER *Input*

On entry: 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) *Input*

On entry: 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

$$\text{DELAM} \leq \text{TOL} \times \max(1.0, |\text{ELAM}|), \tag{1}$$

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) *Input/Output*

On entry: an initial estimate of the eigenvalue $\tilde{\lambda}$.

On exit: the final computed estimate, whether or not an error occurred.

8: DELAM – REAL (KIND=nag_wp) *Input/Output*

On entry: 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 routine 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 $0.25 \times \max(1.0, |\text{ELAM}|)$.

On exit: if IFAIL $= 0$, DELAM holds an estimate of the absolute error in the computed eigenvalue, that is $\left|\tilde{\lambda} - \text{ELAM}\right| \simeq \text{DELAM}$, 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.

9: MONIT – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

MONIT is called by D02KAF 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 D02KAY may be used. (D02KAY is included in the NAG Library.)

The specification of MONIT is:

```
SUBROUTINE MONIT (NIT, IFLAG, ELAM, FINFO)
INTEGER             NIT, IFLAG
REAL (KIND=nag_wp) ELAM, FINFO(15)
```

1: NIT – INTEGER *Input*

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

2: IFLAG – INTEGER *Input*

On entry: describes what phase the computation is in.

IFLAG < 0
An error occurred in the computation at this iteration; an error exit from D02KAF will follow.

IFLAG $= 1$
The routine is trying to bracket the eigenvalue $\tilde{\lambda}$.

IFLAG $= 2$
The routine 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) *Input*

On entry: the current trial value of $\tilde{\lambda}$.

4: FINFO(15) – REAL (KIND=nag_wp) array *Input*

On entry: 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 D02KEF 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.

MONIT must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which D02KAF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

10: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

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 routine.

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 $= 8$

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 $= 9$

C05AZF, called by D02KAF, 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 routine. 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.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

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

8 Parallelism and Performance

D02KAF is not threaded in any implementation.

9 Further Comments

The time taken by D02KAF 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 routine, one should try to obtain good initial values for ELAM and DELAM.

See Section 9 in D02KDF for a discussion of the technique used.

10 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.

10.1 Program Text

```
!   D02KAF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d02kafe_mod

!     Data for D02KAF example program

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: coeffn, monit
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter, Public :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: zero = 0.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: two = 2.0_nag_wp
      Integer, Parameter, Public       :: nin = 5, nout = 6, qq = 5
    Contains
      Subroutine coeffn(p,q,dqdl,x,elam,jint)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: dqdl, p, q
        Real (Kind=nag_wp), Intent (In) :: elam, x
        Integer, Intent (In)           :: jint
!       .. Intrinsic Procedures ..
        Intrinsic                      :: cos, real
!       .. Executable Statements ..
        p = one
        dqdl = one
        q = elam - two*real(qq,kind=nag_wp)*cos(two*x)
        Return
      End Subroutine coeffn
      Subroutine monit(nit,iflag,elam,finfo)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: elam
        Integer, Intent (In)           :: iflag, nit
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: finfo(15)
!       .. Executable Statements ..
        If (nit==14) Then
          Write (nout,*)
          Write (nout,*) 'Output from MONIT'
        End If
        Write (nout,99999) nit, iflag, elam, finfo(1:4)
        Return

99999   Format (1X,2I4,F10.3,2E12.2,2F8.1)
      End Subroutine monit
    End Module d02kafe_mod
    Program d02kafe

!     D02KAF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02kaf, nag_wp
      Use d02kafe_mod, Only: coeffn, monit, nin, nout, one, qq, zero
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: delam, delam1, elam, elam1, tol, xl, &
                                          xr
      Integer                          :: i, ifail, k
!     .. Local Arrays ..
      Real (Kind=nag_wp)               :: bcond(3,2)
!     .. Executable Statements ..
      Write (nout,*) 'D02KAF Example Program Results'
```

```
!     Skip heading in data file
      Read (nin,*)
!     xl: left-hand end point,  xr: right-hand end point,
!     k: index of the required eigenvalue
!     elam1: initial estimate of the eigenvalue
!     delam1: initial search step
      Read (nin,*) xl, xr
      Read (nin,*) k
      Read (nin,*) elam1, delam1
      bcond(1,1:2) = one
      bcond(2,1:2) = zero
      Do i = 5, 6
        tol = 10.0_nag_wp**(-i)
        elam = elam1
        delam = delam1

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call d02kaf(xl,xr,coeffn,bcond,k,tol,elam,delam,monit,ifail)

        Write (nout,*)
        Write (nout,99999) 'Calculation with TOL =', tol
        Write (nout,*)
        Write (nout,*) ' Final results'
        Write (nout,*)
        Write (nout,99998) k, qq, elam, delam
        Write (nout,99997) bcond(3,1), bcond(3,2)
        Write (nout,*)
      End Do

99999 Format (1X,A,E16.4)
99998 Format (1X,' K =',I3,'  QQ =',I3,'  ELAM =',F12.3,'    DELAM =',E12.2)
99997 Format (1X,' BCOND(3,1) =',E12.4,'    BCOND(3,2) =',E12.4)
    End Program d02kafe
```

10.2 Program Data

```
D02KAF Example Program Data
   0.0  3.14159265358979323846       : xl, xr
   4                                 : k
   15.0  4.0                         : elam1, delam1
```

10.3 Program Results

```
 D02KAF Example Program Results

 Output from MONIT
   14   1    15.000   -0.32E+00   -0.11E-03     1.0   206.0
   13   1    15.000   -0.32E+00   -0.57E-04     2.0   234.0
   12   1    19.000    0.26E+00   -0.67E-04     1.0   226.0
   11   2    17.225    0.18E-01   -0.68E-04     1.0   226.0
   10   2    17.097    0.67E-05   -0.64E-04     1.0   226.0
    9   2    17.097   -0.55E-05   -0.64E-04     1.0   226.0
    8   2    17.097   -0.55E-05   -0.64E-04     1.0   226.0

 Calculation with TOL =      0.1000E-04

  Final results

  K =  4  QQ =  5  ELAM =      17.097    DELAM =    0.15E-03
  BCOND(3,1) = -0.9064E+00    BCOND(3,2) =  0.9064E+00


 Output from MONIT
   14   1    15.000   -0.32E+00   -0.11E-03     1.0   206.0
   13   1    15.000   -0.32E+00   -0.56E-05     2.0   410.0
   12   1    19.000    0.26E+00   -0.68E-05     1.0   406.0
   11   2    17.225    0.18E-01   -0.67E-05     1.0   394.0
   10   2    17.097    0.69E-05   -0.64E-05     1.0   394.0
```

```
    9   2    17.097    0.89E-08   -0.64E-05     1.0   394.0
    8   2    17.097   -0.12E-05   -0.64E-05     1.0   394.0
    7   2    17.097    0.90E-08   -0.64E-05     1.0   394.0

 Calculation with TOL =      0.1000E-05

  Final results

  K =  4  QQ =  5  ELAM =      17.097    DELAM =    0.15E-04
  BCOND(3,1) = -0.9075E+00    BCOND(3,2) =  0.9075E+00
```
