NAG Fortran 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,
1                 IFAIL)
INTEGER           K, IFAIL
real              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 a subroutine COEFFN supplied by the user.

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 non-zero and of constant sign throughout the closed interval $[a, b]$;

(b) $\frac{\partial q}{\partial \lambda}$ must be of constant sign and non-zero 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, 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 the user-supplied value 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 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

1: XL – ***real*** *Input*

2: XR – ***real*** *Input*

On entry: the left-hand 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 give the conditions which p and q must satisfy.

Its specification is:

```
SUBROUTINE COEFFN(P, Q, DQDL, X, ELAM, JINT)
INTEGER           JINT
real              P, Q, DQDL, X, ELAM
```

1: P – ***real*** *Output*

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

2: Q – ***real*** *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*** *Output*

On exit: the value of $\frac{\partial q}{\partial \lambda}(x;\lambda)$ 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 percent) will suffice.

4: X – ***real*** *Input*

On entry: the current value of x.

5: ELAM – ***real*** *Input*

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

6: JINT – INTEGER *Input*

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

COEFFN must be declared as EXTERNAL in the (sub)program from which D02KAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

4: BCOND(3,2) – ***real*** 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 $|a_2| + |a_1 p(a)| \neq 0$

while 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 parameter σ_r may be of interest. Refer to the specification of Section 8.5 of the document for D02KDF, for the actual meaning of σ_r and σ_l.

5: K – INTEGER *Input*

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

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

Constraint: K ≥ 0.

6: TOL – ***real*** *Input*

On entry: 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'

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

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 – ***real*** *Input/Output*

On entry: an initial estimate of the eigenvalue.

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

8: DELAM – ***real*** *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. 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 $0.25 \times \max(1.0, |\text{ELAM}|)$.

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

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

MONIT is called by D02KAF at the end of each iteration for λ and allows the user 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 routine.**

If no monitoring is required, the dummy subroutine D02KAY may be used. (D02KAY is included in the NAG Fortran Library. In some implementations of the Library the name is changed to KAYD02: refer to the Users' Note for your implementation.)

Its specification is:

```
SUBROUTINE MONIT(NIT, IFLAG, ELAM, FINFO)
INTEGER          NIT, IFLAG
real             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: IFLAG describes what phase the computation is in.

IFLAG < 0

An error occurred during 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*** *Input*

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

4: FINFO(15) – ***real*** 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 the user. 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 $\delta\lambda$ 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 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 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 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 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 be declared as EXTERNAL in the (sub)program from which D02KAF is called. Parameters 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. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL $= 0$ unless the routine detects an error (see Section 6).

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, for users not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

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.

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 parameter IFLAG of 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 routine.

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 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 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 and 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 subroutine 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 the routine 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 8 of the document for D02KDF for a discussion of the technique used.

9 Example

To find 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 value of π is calculated using X01AAF.

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     D02KAF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Scalars in Common ..
      INTEGER          QQ
*     .. Local Scalars ..
      real             DELAM, ELAM, PI, TOL, XL, XR
      INTEGER          I, IFAIL, K
*     .. Local Arrays ..
      real             BCOND(3,2)
*     .. External Functions ..
      real             X01AAF
      EXTERNAL         X01AAF
*     .. External Subroutines ..
      EXTERNAL         COEFFN, D02KAF, D02KAY
*     .. Common blocks ..
      COMMON           QQ
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02KAF Example Program Results'
      PI = X01AAF(DELAM)
      XL = 0
      XR = PI
      BCOND(1,1) = 1.0e0
      BCOND(2,1) = 0.0e0
      BCOND(1,2) = 1.0e0
      BCOND(2,2) = 0.0e0
      K = 4
      QQ = 5
      DO 20 I = 5, 6
         TOL = 10.0e0**(-I)
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Calculation with TOL =', TOL
         ELAM = 15.0e0
         DELAM = 4.0e0
         IFAIL = 1
*
*        * To obtain monitoring information from the supplied
*        subroutine MONIT replace the name D02KAY by MONIT in
*        the next statement, and declare MONIT as external *
*
         CALL D02KAF(XL,XR,COEFFN,BCOND,K,TOL,ELAM,DELAM,D02KAY,IFAIL)
*
```

```
         WRITE (NOUT,*)
         IF (IFAIL.NE.0) THEN
            WRITE (NOUT,99996) ' D02KAF fails. IFAIL =', IFAIL
         ELSE
            WRITE (NOUT,*) ' Final results'
            WRITE (NOUT,*)
            WRITE (NOUT,99998) ' K =', K, '  QQ =', QQ, '  ELAM =',
     +        ELAM, '    DELAM =', DELAM
            WRITE (NOUT,99997) ' BCOND(3,1) =', BCOND(3,1),
     +        '    BCOND(3,2) =', BCOND(3,2)
            WRITE (NOUT,*)
         END IF
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,A,e16.4)
99998 FORMAT (1X,A,I3,A,I3,A,F12.3,A,e12.2)
99997 FORMAT (1X,A,e12.4,A,e12.4)
99996 FORMAT (1X,A,I3)
      END
*
      SUBROUTINE COEFFN(P,Q,DQDL,X,ELAM,JINT)
*     .. Scalar Arguments ..
      real               DQDL, ELAM, P, Q, X
      INTEGER            JINT
*     .. Scalars in Common ..
      INTEGER            QQ
*     .. Intrinsic Functions ..
      INTRINSIC          COS, real
*     .. Common blocks ..
      COMMON             QQ
*     .. Executable Statements ..
      P = 1.0e0
      DQDL = 1.0e0
      Q = ELAM - 2.0e0*real(QQ)*COS(2.0e0*X)
      RETURN
      END
*
      SUBROUTINE MONIT(NIT,IFLAG,ELAM,FINFO)
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Scalar Arguments ..
      real               ELAM
      INTEGER          IFLAG, NIT
*     .. Array Arguments ..
      real               FINFO(15)
*     .. Local Scalars ..
      INTEGER          I
*     .. Executable Statements ..
      IF (NIT.EQ.14) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Output from MONIT'
      END IF
      WRITE (NOUT,99999) NIT, IFLAG, ELAM, (FINFO(I),I=1,4)
      RETURN
*
99999 FORMAT (1X,2I4,F10.3,2e12.2,2F8.1)
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D02KAF Example Program Results

 Calculation with TOL =      0.1000E-04

  Final results

  K =  4  QQ =  5  ELAM =      17.097    DELAM =    0.11E-03
  BCOND(3,1) = -0.9064E+00    BCOND(3,2) =  0.9064E+00


 Calculation with TOL =      0.1000E-05

  Final results

  K =  4  QQ =  5  ELAM =      17.097    DELAM =    0.11E-04
  BCOND(3,1) = -0.9075E+00    BCOND(3,2) =  0.9075E+00
```