NAG Library Routine Document

D02LAF

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

D02LAF is a routine for integrating a non-stiff system of second-order ordinary differential equations using Runge–Kutta–Nystrom techniques.

2 Specification

```
SUBROUTINE D02LAF(FCN, NEQ, T, TEND, Y, YP, YDP, RWORK, LRWORK, IFAIL)
INTEGER           NEQ, LRWORK, IFAIL
double precision  T, TEND, Y(NEQ), YP(NEQ), YDP(NEQ), RWORK(LRWORK)
EXTERNAL          FCN
```

3 Description

Given the initial values $x, y_1, y_2, \ldots, y_{\mathrm{NEQ}}, y'_1, y'_2, \ldots, y'_{\mathrm{NEQ}}$ D02LAF integrates a non-stiff system of second-order differential equations of the type

$$y''_i = f_i\big(x, y_1, y_2, \ldots, y_{\mathrm{NEQ}}\big), \qquad i = 1, 2, \ldots, \mathrm{NEQ},$$

from $x = \mathrm{T}$ to $x = \mathrm{TEND}$ using a Runge–Kutta–Nystrom formula pair. The system is defined by FCN, which evaluates f_i in terms of x and $y_1, y_2, \ldots, y_{\mathrm{NEQ}}$, where $y_1, y_2, \ldots, y_{\mathrm{NEQ}}$ are supplied at x.

There are two Runge–Kutta–Nystrom formula pairs implemented in this routine. The lower order method is intended if you have moderate accuracy requirements and may be used in conjunction with the interpolation routine D02LZF to produce solutions and derivatives at user-specified points. The higher order method is intended if you have high accuracy requirements.

In one-step mode the routine returns approximations to the solution, derivative and f_i at each integration point. In interval mode these values are returned at the end of the integration range. You select the order of the method, the mode of operation, the error control and various optional inputs by a prior call to D02LXF.

For a description of the Runge–Kutta–Nystrom formula pairs see Dormand *et al.* (1986a) and Dormand *et al.* (1986b) and for a description of their practical implementation see Brankin *et al.* (1989).

4 References

Brankin R W, Dormand J R, Gladwell I, Prince P J and Seward W L (1989) Algorithm 670: A Runge–Kutta–Nystrom Code *ACM Trans. Math. Software* **15** 31–40

Dormand J R, El–Mikkawy M E A and Prince P J (1986a) Families of Runge–Kutta–Nystrom formulae *Mathematical Report TPMR 86-1* Teesside Polytechnic

Dormand J R, El–Mikkawy M E A and Prince P J (1986b) High order embedded Runge–Kutta–Nystrom formulae *Mathematical Report TPMR 86-2* Teesside Polytechnic

5 Parameters

1: FCN – SUBROUTINE, supplied by the user. *External Procedure*

FCN must evaluate the functions f_i (that is the second derivatives y''_i) for given values of its arguments x, $y_1, y_2, \ldots, y_{\mathrm{NEQ}}$.

The specification of FCN is:

```
SUBROUTINE FCN(NEQ, T, Y, F)
INTEGER          NEQ
double precision T, Y(NEQ), F(NEQ)
```

1: NEQ – INTEGER *Input*

On entry: the number of differential equations.

2: T – ***double precision*** *Input*

On entry: x, the value of the argument.

3: Y(NEQ) – ***double precision*** array *Input*

On entry: y_i, the value of the argument, for $i = 1, 2, \ldots, \text{NEQ}$.

4: F(NEQ) – ***double precision*** array *Output*

On exit: the value of f_i, for $i = 1, 2, \ldots, \text{NEQ}$.

FCN must be declared as EXTERNAL in the (sub)program from which D02LAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: NEQ – INTEGER *Input*

On entry: the number of second-order ordinary differential equations to be solved by D02LAF. It must contain the same value as the parameter NEQ used in a prior call to D02LXF.

Constraint: NEQ ≥ 1.

3: T – ***double precision*** *Input/Output*

On entry: the initial value of the independent variable x.

On exit: the value of the independent variable, which is usually TEND, unless an error has occurred or the code is operating in one-step mode. If the integration is to be continued, possibly with a new value for TEND, T must not be changed.

Constraint: T $\neq$ TEND.

4: TEND – ***double precision*** *Input*

On entry: the end point of the range of integration. If TEND $<$ T on initial entry, integration will proceed in the negative direction. TEND may be reset, in the direction of integration, before any continuation call.

5: Y(NEQ) – ***double precision*** array *Input/Output*

On entry: the initial values of the solution $y_1, y_2, \ldots, y_{\text{NEQ}}$.

On exit: the computed values of the solution at the exit value of T. If the integration is to be continued, possibly with a new value for TEND, these values must not be changed.

6: YP(NEQ) – ***double precision*** array *Input/Output*

On entry: the initial values of the derivatives $y'_1, y'_2, \ldots, y'_{\text{NEQ}}$.

On exit: the computed values of the derivatives at the exit value of T. If the integration is to be continued, possibly with a new value for TEND, these values must not be changed.

7: YDP(NEQ) – ***double precision*** array *Input/Output*

On entry: must be unchanged from a previous call to D02LAF.

On exit: the computed values of the second derivative at the exit value of T, unless illegal input is detected, in which case the elements of YDP may not have been initialized. If the integration is to be continued, possibly with a new value for TEND, these values must not be changed.

8: RWORK(LRWORK) – ***double precision*** array *Communication Array*

This **must** be the same parameter RWORK as supplied to D02LXF. It is used to pass information from D02LXF to D02LAF, and from D02LAF to both D02LYF and D02LZF. Therefore the contents of this array **must not** be changed before the call to D02LAF or calling either of the routines D02LYF and D02LZF.

9: LRWORK – INTEGER *Input*

On entry: the dimension of the array RWORK as declared in the (sub)program from which D02LAF is called.

This must be the same parameter LRWORK as supplied to D02LXF.

10: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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, because for this routine the values of the output parameters may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **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 $= 1$

Illegal input detected, i.e., one of the following conditions:

on any call, T $=$ TEND, or the value of NEQ or LRWORK has been altered;

on a continuation call, the direction of integration has been changed;

D02LXF had not been called previously, or the previous call to D02LXF resulted in an error exit.

This error exit can be caused if elements of RWORK have been overwritten.

IFAIL $= 2$

The maximum number of steps has been attempted. (See parameter MAXSTP in D02LXF.) If integration is to be continued then you need only reset IFAIL and call the routine again and a further MAXSTP steps will be attempted.

IFAIL $= 3$

In order to satisfy the error requirements, the step size needed is too small for the ***machine precision*** being used.

IFAIL $= 4$

The code has detected two successive error exits at the current value of x and cannot proceed. Check all input variables.

IFAIL = 5

The code has detected inefficient use of the integration method. The step size has been reduced by a significant amount too often in order to hit the output points specified by TEND. (Of the last 100 or more successful steps more than 10% are steps with sizes that have had to be reduced by a factor of greater than a half.)

7 Accuracy

The accuracy of integration is determined by the parameters TOL, THRES and THRESP in a prior call to D02LXF. Note that only the local error at each step is controlled by these parameters. The error estimates obtained are not strict bounds but are usually reliable over one step. Over a number of steps the overall error may accumulate in various ways, depending on the system. The code is designed so that a reduction in TOL should lead to an approximately proportional reduction in the error. You are strongly recommended to call D02LAF with more than one value for TOL and to compare the results obtained to estimate their accuracy.

The accuracy obtained depends on the type of error test used. If the solution oscillates around zero a relative error test should be avoided, whereas if the solution is exponentially increasing an absolute error test should not be used. For a description of the error test see the specifications of the parameters TOL, THRES and THRESP in routine document D02LXF.

8 Further Comments

If D02LAF fails with IFAIL = 3 then the value of TOL may be so small that a solution cannot be obtained, in which case the routine should be called again with a larger value for TOL. If the accuracy requested is really needed then you should consider whether there is a more fundamental difficulty. For example:

(a) in the region of a singularity the solution components will usually be of a large magnitude. D02LAF could be used in one-step mode to monitor the size of the solution with the aim of trapping the solution before the singularity. In any case numerical integration cannot be continued through a singularity, and analytical treatment may be necessary;

(b) if the solution contains fast oscillatory components, the routine will require a very small step size to preserve stability. This will usually be exhibited by excessive computing time and sometimes an error exit with IFAIL = 3. The Runge–Kutta–Nystrom methods are not efficient in such cases and you should consider reposing your problem as a system of first-order ordinary differential equations and then using a routine from sub-chapter D02M–N with the Blend formulae (see D02NWF).

D02LAF can be used for producing results at short intervals (for example, for tabulation), in two ways. By far the less efficient is to call D02LAF successively over short intervals, $t + (i-1) \times h$ to $t + i \times h$, although this is the only way if the higher order method has been selected and precisely **not** what it is intended for. A more efficient way, **only** for use when the lower order method has been selected, is to use D02LAF in one-step mode. The output values of parameters Y, YP, YDP, T and RWORK are set correctly for a call to D02LZF to compute the solution and derivative at the required points.

9 Example

This example solves the following system (the two body problem)

$$\begin{aligned} y_1'' &= -y_1/\left(y_1^2 + y_2^2\right)^{3/2} \\ y_2'' &= -y_2/\left(y_1^2 + y_2^2\right)^{3/2} \end{aligned}$$

over the range $[0, 20]$ with initial conditions $y_1 = 1.0 - \epsilon$, $y_2 = 0.0$, $y_1' = 0.0$ and $y_2' = \sqrt{\left(\dfrac{1+\epsilon}{1-\epsilon}\right)}$ where ϵ, the eccentricity, is 0.5. The system is solved using the lower order method with relative local error tolerances 1.0D−4 and 1.0D−5 and default threshold tolerances. D02LAF is used in one-step mode (ONESTP = .TRUE.) and D02LZF provides solution values at intervals of 2.0.

9.1 Program Text

```
*     D02LAF Example Program Text
*     Mark 14 Revised. NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NEQ, LRWORK, NWANT
      PARAMETER        (NEQ=2,LRWORK=16+20*NEQ,NWANT=NEQ)
*     .. Local Scalars ..
      DOUBLE PRECISION ECC, H, HNEXT, HSTART, HUSED, T, TEND, TINC,
     +                 TNEXT, TOL, TSTART, Y1, Y2, YP1, YP2
      INTEGER          IFAIL, ITOL, K, MAXSTP, NATT, NFAIL, NSUCC
      LOGICAL          HIGH, ONESTP, START
*     .. Local Arrays ..
      DOUBLE PRECISION RWORK(LRWORK), THRES(NEQ), THRESP(NEQ), Y(NEQ),
     +                 YDP(NEQ), YP(NEQ), YPWANT(NWANT), YWANT(NWANT)
*     .. External Subroutines ..
      EXTERNAL         D02LAF, D02LXF, D02LYF, D02LZF, FCN
*     .. Intrinsic Functions ..
      INTRINSIC        SQRT
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02LAF Example Program Results'
      HIGH = .FALSE.
      ONESTP = .TRUE.
      TINC = 2.0D0
*
*     Initial conditions
*
      TSTART = 0.0D0
      ECC = 0.5D0
      Y1 = 1.0D0 - ECC
      Y2 = 0.0D0
      YP1 = 0.0D0
      YP2 = SQRT((1.0D0+ECC)/(1.0D0-ECC))
      TEND = 20.0D0
*
      DO 60 ITOL = 4, 5
         TOL = 10.0D0**(-ITOL)
         WRITE (NOUT,*)
*
*        Call D02LXF with default THRES,THRESP,MAXSTP and H
*
         THRES(1) = 0.0D0
         THRESP(1) = 0.0D0
         H = 0.0D0
         MAXSTP = 0
         START = .TRUE.
         IFAIL = 1
*
         CALL D02LXF(NEQ,H,TOL,THRES,THRESP,MAXSTP,START,ONESTP,HIGH,
     +               RWORK,LRWORK,IFAIL)
*
         IF (IFAIL.NE.0) THEN
            WRITE (NOUT,99995) IFAIL
            GO TO 80
         END IF
         WRITE (NOUT,99999) 'Calculation with TOL = ', TOL
         WRITE (NOUT,*)
         WRITE (NOUT,*) '   T      Y(1)       Y(2)'
*
*        Set initial values
*
         Y(1) = Y1
         Y(2) = Y2
         YP(1) = YP1
         YP(2) = YP2
         T = TSTART
         TNEXT = T + TINC
         WRITE (NOUT,99998) T, (Y(K),K=1,NEQ)
*
```

```
*        Loop point for onestep mode
*
   20    IFAIL = 1
*
         CALL D02LAF(FCN,NEQ,T,TEND,Y,YP,YDP,RWORK,LRWORK,IFAIL)
*
         IF (IFAIL.GT.0) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,99997) 'D02LAF returned with IFAIL = ', IFAIL,
     +        '  at T = ', T
            GO TO 80
         END IF
*
*        Loop point for interpolation
*
   40    IF (TNEXT.LE.T) THEN
            IFAIL = 0
*
            CALL D02LZF(NEQ,T,Y,YP,NEQ,TNEXT,YWANT,YPWANT,RWORK,LRWORK,
     +                  IFAIL)
*
            WRITE (NOUT,99998) TNEXT, (YWANT(K),K=1,NEQ)
            TNEXT = TNEXT + TINC
            GO TO 40
         END IF
*
         IF (T.LT.TEND) GO TO 20
*
         IFAIL = 0
*
         CALL D02LYF(NEQ,HNEXT,HUSED,HSTART,NSUCC,NFAIL,NATT,THRES,
     +               THRESP,RWORK,LRWORK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99996) ' Number of successful steps = ', NSUCC
         WRITE (NOUT,99996) ' Number of   failed   steps = ', NFAIL
   60 CONTINUE
   80 CONTINUE
*
99999 FORMAT (1X,A,1P,E9.1)
99998 FORMAT (1X,F5.1,2(2X,F9.5))
99997 FORMAT (1X,A,I2,A,1P,E10.3)
99996 FORMAT (1X,A,I5)
99995 FORMAT (1X,/1X,' ** D02LXF returned with IFAIL = ',I5)
      END
*
      SUBROUTINE FCN(NEQ,T,Y,YDP)
*
*     Derivatives for two body problem in  y'' = f(t,y)  form
*
*     .. Scalar Arguments ..
      DOUBLE PRECISION T
      INTEGER          NEQ
*     .. Array Arguments ..
      DOUBLE PRECISION Y(NEQ), YDP(NEQ)
*     .. Local Scalars ..
      DOUBLE PRECISION R
*     .. Intrinsic Functions ..
      INTRINSIC        SQRT
*     .. Executable Statements ..
      R = SQRT(Y(1)**2+Y(2)**2)**3
      YDP(1) = -Y(1)/R
      YDP(2) = -Y(2)/R
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D02LAF Example Program Results

 Calculation with TOL =   1.0E-04

    T      Y(1)       Y(2)
  0.0    0.50000    0.00000
  2.0   -1.20573    0.61357
  4.0   -1.33476   -0.47685
  6.0    0.35748   -0.44558
  8.0   -1.03762    0.73022
 10.0   -1.42617   -0.32658
 12.0    0.05515   -0.72032
 14.0   -0.82880    0.81788
 16.0   -1.48103   -0.16788
 18.0   -0.26719   -0.84223
 20.0   -0.57803    0.86339

 Number of successful steps =   108
 Number of   failed   steps =    16

 Calculation with TOL =   1.0E-05

    T      Y(1)       Y(2)
  0.0    0.50000    0.00000
  2.0   -1.20573    0.61357
  4.0   -1.33476   -0.47685
  6.0    0.35748   -0.44558
  8.0   -1.03762    0.73022
 10.0   -1.42617   -0.32658
 12.0    0.05516   -0.72031
 14.0   -0.82880    0.81787
 16.0   -1.48103   -0.16789
 18.0   -0.26718   -0.84223
 20.0   -0.57804    0.86338

 Number of successful steps =   169
 Number of   failed   steps =    15
```