1 SUMMARY

This subroutine **solves a system of linear or non-linear first order differential equations**

$$y_i' = f_i(y_1, y_2, ..., y_m, x) \qquad i=1,2,...,m \qquad a \le x \le b \tag{1.1}$$

with linear or non-linear two-point boundary conditions

$$g_i(y_1(a), ..., y_m(a), y_1(b), ..., y_m(b)) = 0 \qquad i=1,2,...,m. \tag{1.2}$$

The program should adequately solve problems with mild boundary layers or spikes. It also solves smooth and linear problems efficiently and accurately.

DD14A requires the user to calculate analytic derivatives and returns a solution according to a user-specified absolute accuracy. For highly non-linear problems the user may specify an initial mesh and/or an approximation to the solution. A continuation option is also provided. DD14A will refine the initial uniform or user-supplied mesh adaptively inserting mesh points non-uniformly according to an estimate of the error in the solution.

The method used is a modification of that of Lentini and Pereyra (*SIAM J. Numer. Anal.* **14,** 91-111 (1977)).

ATTRIBUTES — **Version:** 1.1.0. **Types:** Real (single, double). **Calls:** FD15. **Original date:** April 2001. **Remark:** The DD14 entries are threadsafe versions of DD04. **Origin:** Lentini and Pereyra, University of Caracas, modified for Harwell by I.S. Duff.

2 HOW TO USE THE PACKAGE

2.1 Initialization

The DD14I/ID entry must be called prior to the first call to the DD14A/AD entry to initialize the control arrays.

The single precision version

```
CALL DD14I(ICNTL,CNTL)
```

The double precision version

```
CALL DD14ID(ICNTL,CNTL)
```

ICNTL is an INTEGER array of length 10, see Section 2.3.

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 10, see Section 2.3.

2.2 The argument list and calling sequence

The single precision version

```
CALL DD14A(M,NMAX,N,NUMBEG,NUMMIX,A,B,TOL,X,Y,
  ABT,FF,JACOB,WORK,LWORK,IWORK,LIWORK,IFLAG,ICNTL,CNTL)
```

The double precision version

```
CALL DD14AD(M,NMAX,N,NUMBEG,NUMMIX,A,B,TOL,X,Y,
  ABT,FF,JACOB,WORK,LWORK,IWORK,LIWORK,IFLAG,ICNTL,CNTL)
```

M is an INTEGER variable which must be set by the user to the number of equations to be solved. It is not altered by the subroutine.

NMAX is an INTEGER variable which must be set by the user to the maximum number of values for x for which the solution is constructed (we call this the grid of mesh points). If the workspace allocated by the user is too small

for this value of NMAX, then the values used internally by the subroutine will be reduced to the largest value commensurate with the workspace allocation as long as this value is not less than N. If it is less, then an error return is invoked (see Section 2.4).

N is an INTEGER variable which must be set by the user to the initial number of mesh points, including the end points. N can be altered by the subroutine and will, on output, give the number of grid points in the final mesh. The initial points will also be a subset of the final set.

Restriction: $3 < \texttt{N} \leq \texttt{NMAX}$.

NUMBEG is an INTEGER variable which must be set by the user to the number of initial boundary conditions (i.e. involving only $y_i(a)$, i=1,2,...,m). It is not altered by the subroutine.

Restriction: $0 \leq \texttt{NUMBEG} < \texttt{M}$.

NUMMIX is an INTEGER variable which must be set by the user to the number of coupled boundary conditions (i.e. those involving $y_i i(a)$ and $y_i(b)$, i=1,2,...,m). It is not altered by the subroutine.

Restrictions: $0 \leq \texttt{NUMMIX, NUMMIX+NUMBEG} \leq \texttt{M}$.

A is a REAL (DOUBLE PRECISION in the D version) variable which must be set by the user to the value of *a*. It is not altered by the subroutine.

B is a REAL (DOUBLE PRECISION in the D version) variable which must be set by the user to the value of *b*. It is not altered by the subroutine.

TOL is a REAL (DOUBLE PRECISION in the D version) variable which must be set by the user to the accuracy required. It is the desired final maximum absolute error for all components and grid points. The user should note that there is no provision in DD14 for relative precision and should consult Section 2.5 for further information on this. It is not altered by the subroutine.

X is a REAL (DOUBLE PRECISION in the D version) array of length at least NMAX. It need not be set by the user on input unless an input grid is to be specified (see ICNTL(3), Section 2.3). On output, X(I), I=1,...,N contains the grid points of the final mesh used where A=X(1) < X(2)... < X(I) < X(I+1)... < X(N)=B. These will always include those of the initial mesh, which will be a uniform grid unless specified otherwise.

Y is a REAL (DOUBLE PRECISION in the D version) array of dimensions (M,NMAX). It need not be set by the user on input unless he wishes to specify an initial approximation to the solution (see ICNTL(3), Section 2.3). Y(J,I) holds the numerical approximation to $y_J(x_I)$. Unless otherwise requested it is initially set to zero by the subroutine. On exit, it holds the computed solution.

ABT is a REAL (DOUBLE PRECISION in the D version) array of length at least M. It need not be set by the user and, on output, ABT(I), I=1,...,M holds the largest estimated error on the Ith component of the solution over all grid points.

FF is a user supplied SUBROUTINE and must be declared EXTERNAL in the user's program. Its calling sequence should be

```
SUBROUTINE FF(X,Y,M,N,F,G,EPSU,CONT)
```

where X, Y, M and N are as previously defined, F is a REAL (DOUBLE PRECISION in the D version) array of dimensions (M,NMAX), G is a REAL (DOUBLE PRECISION in the D version) array of length M and EPSU is a REAL (DOUBLE PRECISION in the D version) variable passed to the subroutine set to the value of ε.

CONT is a LOGICAL variable which specifies how DD14A/AD expects the subroutine to set the arrays F and G. If CONT=.FALSE. the user should return in F(I,J) the value of $f_i(y,x_J)$. Additionally, G(I), I=1,...,M must be set to the value of $g_i(y(a),y(b))$. However, if CONT=.TRUE. the arrays F and G should be filled with $\partial f_i/\partial\varepsilon|_{x=x_J}$ and $\partial g_{i/}\partial\varepsilon$ respectively, i=1,...,m, j=1,...,N. Initial conditions must precede mixed conditions which must precede end conditions.

JACOB is a user supplied SUBROUTINE and must be declared EXTERNAL in the user's program. Its calling sequence

should be

```
SUBROUTINE JACOB(X,Y,II,C,M,N,A1,B1,EPSU)
```

where X, Y, M and N are as previously defined. II is an INTEGER variable whose value ($1 \leq$ II $\leq$ N) is input to JACOB by the DD14 suite, C, A1 and B1 are REAL (DOUBLE PRECISION in the D version) arrays of dimension (M,M) and EPSU is a REAL (DOUBLE PRECISION in the D version) variable passed to the subroutine set to the value of ε.

The user should write his subroutine so that it returns in C(I,J) the value of $\partial f_I/\partial y_J$ at the point X_{II}. A1(I,J) should contain the value of $\partial g_I/\partial y_J(a)$ and B1(I,J) should contain the value of $\partial g_I/\partial y_J(b)$, that is, the derivatives at the end points of the calculation. A1 and B1 need only be evaluated when II equals 1.

WORK is a REAL (DOUBLE PRECISION in the D version) array of length at least 3 * M ** 2 * NMAX+6 * M * NMAX+2 * NMAX+4 * M^2+M. It is used as workspace by DD14.

LWORK is an INTEGER variable which must be set to the length of array WORK. Notice that if it is too small, DD14 will act as if NMAX was smaller. It is not altered by the subroutine.

IWORK is an INTEGER array of length at least 2 * M * NMAX+NMAX+M. It is used as workspace as DD14.

LIWORK is an INTEGER variable which must be set by the user to the length of array IWORK. Notice that if it is too small, DD14 will act as if NMAX was smaller. It is not altered by the subroutine.

IFLAG is an INTEGER variable. It need not be set by the user and on output it indicates whether the run has been successful (IFLAG=0) or an error condition has occurred (IFLAG=1,2,3,4,5)... see Section 2.4.

ICNTL is an INTEGER array of length 10, see Section 2.3.

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 10, see Section 2.3.

2.3 The control arrays

The ICNTL array argument which must be of length 10 can be used to pass optional integer control values to the routine.

ICNTL(1) specifies the unit number to be used to output error messages. It has a default value of 6 which can be reset by the user to another unit or set negative if messages are to be suppressed.

ICNTL(2) specifies the unit number to be used when writing information concerning the progress of the computation. By default, the output of such information is suppressed by a negative value but the user can turm the information messages on by resetting ICNTL(2) to any non-negative valid unit number.

ICNTL(3) is set to a non-zero value when an initial grid and initial guess at the solution is being supplied. In that case the grid points should be put in X and the trial solution in Y in the same way as was mentioned earlier (see X and Y in Section 2.2). The default value for ICNTL(3) is zero which causes DD14A/AD to choose a uniform grid on N points with trial solution zero everywhere.

ICNTL(4) has a default value of zero. If the problem (1.1), (1.2) is linear in *y* a more efficient performance can be achieved by setting ICNTL(4) to non-zero (to 1 say).

ICNTL(5) to ICNTL(10) should not be altered and at present are not used by DD14.

The CNTL array argument which must be of length 10 can be used to pass optional floating point control values to the routine.

CNTL(1) has a default value of zero. If it is set by the user to a value strictly between 0 and 1, then a continuation method is invoked to generate good initial values in severely non-linear cases. Consequent changes should be made to the user supplied subroutines FF and JACOB.

The user must set up a family of problems

$$y' = f(x,y,\varepsilon) \tag{2.2}$$

$$g = (y(a), y(b), \varepsilon) = 0$$

such that for ε=0 the problem is simple and for ε=1 the original problem is recovered. The subroutine will automatically attempt to go from ε=0 to ε=1 using `CNTL(1)` as the starting step in the continuation. Since an arbitrary lower bound of 0.01 is imposed on the step, the user has to rescale ε to use smaller steps. Once ε≥1, the regular procedure continues.

`CNTL(2)` to `CNTL(10)` should not be altered and at present are not used by `DD14`.

2.4 Diagnostic messages

`IFLAG` is an `INTEGER` parameter which is always set by the subroutine to indicate success or failure. In the case of failure, an appropriate message is output on unit `ICNTL(1)`. Possible values for `IFLAG` are:

`0` Execution successful. No message output.

`1` One of the restrictions on `M, N, NUMBEG` or `NUMMIX` has been violated. The values of these variables are printed out in addition to the error message.

`2` A finer mesh is required for the accuracy requested; `NMAX` is not large enough.

`3` Newton iteration fails to converge. A possible source of error is due to a faulty Jacobian evaluation `(A1, B1, C)` in `JACOB.` The user should check this before investigating further.

`4` Newton iteration reached roundoff level. It could be, however, that the answer returned is satisfactory.

`5` Workspace is too small. Values required, for the given value of `NMAX,` for `LWORK` and/or `LIWORK` are printed. The failure occurs because the reduced effective value of `NMAX` which the workspace allows is less than `N.`

2.5 A note on absolute error tolerance

The user should note that `DD14` attempts to obtain a solution whose maximum absolute error over all components and grid points is less than the user specified `TOL.` This may not be a suitable criterion when the components vary widely in magnitude from each other or over the range of integration. Wherever possible the user should scale the problem so that all components will have similar orders of magnitude. For example for the problem

$$y'' - \lambda^2 y = 0$$

if $\lambda >> 1$, it would be better to work with

$$y_1' = \lambda y_2$$

$$y_2' = \lambda y_1$$

rather than either of the alternative forms

$$y_1' = y_2$$

$$y_2' = \lambda^2 y_1$$

or

$$y_1' = \lambda^2 y_2$$

$$y_2' = y_1$$

2.6 Other routines used

`DD14` is really a driver subroutine to provide a convenient user interface to a package of subroutines. The package is shown on the diagram which follows, with a line between subroutines indicating that the upper subroutine calls the lower.

3 GENERAL INFORMATION

Use of common: None.

Workspace: Real work array of length NMAX(3M^2 +6M+2)+4M^2+M. Integer work array of length NMAX(2M+1)+M (see Section 2.2).

Input/output: Diagnostic messages are output to stream ICNTL(1) and messages on progress of the calculation to unit ICNTL(2). If either is set to zero the corresponding output is suppressed (see Sections 2.3 and 2.4).

Other routines called directly: User supplied subroutines to calculate function values and derivatives (see Section 2.2).

Restrictions: 3 < N ≤ MAX, M > 1, 0 ≤ NUMBEG < M, 0 < NUMBEG+NUMMIX ≤ M, NUMMIX ≥ 0.

4 METHOD

DD14D is an adaptive (variable order, variable step) finite difference solver for nonlinear first order systems of ordinary differential equations subject to nonlinear two-point boundary conditions. It is a minor modification of PASVA3 written by Lentini and Pereyra in December 1977. An earlier version PASVAR was described in Lentini and Pereyra (*SIAM J. Numer. Analy.* **14,** 111 (1977)).

The trapezoidal rule is used as the basis for the discretization of the system of equations (1.1). Thus, if the grid is $a=x_1<x_2...x_i<x_{i+1}...x_n=b$ and $y(x_i)\equiv y_i$ and $h_i\equiv x_{i+1}-x_i$, then the discretization produces the set of non-linear equations

$$\frac{(y_{i+1})_j-(y_i)_j}{h_i}=\tfrac{1}{2}\{f_j(x_{i+1},y_{i+1})+f_j(x_i,y_i)\}$$

j=1,...,m, i=1,...,n–1, and the associated boundary conditions

$$g_j(y_1,y_n)=0 \qquad j=1,...,m.$$

This set of mn equations is solved using Newton's method where the user must supply Jacobians for **f** and **g**. A descent Newton iteration with step and angle control is used with an optional continuation procedure available in cases where there are convergence difficulties.

The subroutine then checks to see if a refinement of the mesh is required (this refinement is based on the principle of equidistribution of the error, basically requiring that the first term in the local truncation error be nearly constant in norm over the mesh). When it is happy with the initial mesh, it then keeps increasing the order of the method by means of using deferred corrections until one of three things happen. Either a satisfactory solution is obtained or the deferred corrections do not give a suitable reduction in error or the maximum number of deferred corrections is reached. In these latter cases the mesh is further refined and the correction process continued until either the problem is solved, too many mesh points are required (IFLAG=2), Newton iterations diverge (IFLAG=3), or roundoff errors are reached with subsequent corrections not giving any improvement (IFLAG=4).

5. AN EXAMPLE OF USE

Consider the problem

$$y'''+yy''+2(1-y'^2)=0$$

with

$$y(0)=y'(0)=0 \quad y'(10)=1.$$

This can be reformulated as a system of first order equations

$$y_1'=y_2$$

$$y_2'=y_3$$

$$y_3'=-y_1.y_3-2(1-y_2^2)$$

with boundary conditions $y_1(0)=y_2(0)=0$, $y_2(10)=1$.

We employ continuation to illustrate its use, although in this case it is not necessary.

The problem used for continuation is

$$y_1' = y_2$$
$$y_2' = y_3$$
$$y_3' = -y_1 . y_3 - 2\varepsilon(1 - y_2^2)$$

with initial value of ε = 0.1.

The following program could be used to solve this problem:

```
C TEST DECK FOR DD14AD.
C CONTINUATION IS USED TO ILLUSTRATE ITS USE EVEN ALTHOUGH
C     IT IS NOT STRICTLY NECESSARY FOR THIS PROBLEM.
C     .. Local Scalars ..
      DOUBLE PRECISION A,B,TOL
      INTEGER I,IFLAG,J,LIWORK,LWORK,M,N,NMAX,NUMBEG,NUMMIX
C     ..
C     .. Local Arrays ..
      DOUBLE PRECISION ABT(3),WORK(10191),X(216),Y(3,216)
      INTEGER IWORK(1515)
      DOUBLE PRECISION CNTL(10)
      INTEGER ICNTL(10)
C     ..
C     .. External Subroutines ..
      EXTERNAL DD14AD,FF,JACOB
C     ..
C     .. Executable Statements ..
      CALL DD14ID(ICNTL,CNTL)
C MAXIMUM NUMBER OF MESH POINTS ALLOWED.
      NMAX = 216
C REQUESTED ABSOLUTE ERROR. MAXIMUM OVER ALL COMPONENTS AND
C     GRID POINTS.
      TOL = 1.D-6
C NUMBER OF COMPONENTS (DEPENDENT VARIABLES).
      M = 3
C INITIAL NUMBER OF MESH POINTS.
      N = 17
C TWO INITIAL CONDITIONS AND NO MIXED ONES.
      NUMBEG = 2
      NUMMIX = 0
C RANGE OF INTEGRATION IS 0. TO 10.
      A = 0.
      B = 10.
C CONTINUATION REQUESTED.  INITIAL CONTINUATION PARAMETER VALUE
C     IS 0.1
      CNTL(1) = 0.1
C SIZE OF WORK ARRAYS.
      LWORK = 10191
      LIWORK = 1515
C CALL TO DD14AD.
      CALL DD14AD(M,NMAX,N,NUMBEG,NUMMIX,A,B,TOL,X,Y,ABT,FF,JACOB,WORK,
     +            LWORK,IWORK,LIWORK,IFLAG,ICNTL,CNTL)
C
C TEST FOR ERROR CONDITION.
      IF (IFLAG.NE.0) GO TO 10
C OUTPUT OF SOLUTION ON FINAL GRID.
      WRITE (6,FMT=9000) N,ABT

 9000 FORMAT (' SOLUTION ON FINAL GRID OF ',I2,' POINTS',/,' ERROR IS ',
```

```
     +      1P,3D22.14)

       WRITE (6,9010)
       WRITE (6,9020)
       WRITE (6,9030) (X(J), (Y(I,J),I=1,M),J=1,N)
 9010 FORMAT (/' SOLUTION IS')
 9020 FORMAT (8X,'X(I)',14X,'Y1(I)',13X,'Y2(I)',13X,'Y3(I)')
 9030 FORMAT ((6X,F7.3,5X,3 (2X,1PD16.9)))
       GO TO 20
   10 WRITE (6,9040) IFLAG
 9040 FORMAT (' ERROR DETECTED IFLAG = ',I2)
   20 STOP
       END

       SUBROUTINE FF(X,Y,M,N,F,G,EPSNU,CONT)
C      .. Scalar Arguments ..
       INTEGER M,N
       DOUBLE PRECISION EPSNU
       LOGICAL CONT
C      ..
C      .. Array Arguments ..
       DOUBLE PRECISION F(M,N),G(M),X(M,N),Y(M,N)
C      ..
C      .. Local Scalars ..
       INTEGER I
C      ..
C      .. Executable Statements ..
C JUMP IF WE ARE STILL IN CONTINUATION PHASE.
       IF (CONT) GO TO 20
C BOUNDARY CONDITIONS.
       G(1) = Y(1,1)
       G(2) = Y(2,1)
       G(3) = Y(2,N) - 1.0
C EQUATIONS.
       DO 10 I = 1,N
         F(1,I) = Y(2,I)
         F(2,I) = Y(3,I)
         F(3,I) = -Y(1,I)*Y(3,I) - 2.* (1.-Y(2,I)**2)*EPSNU
   10 CONTINUE
       GO TO 50
C CONTINUATION PHASE.
C DERIVATIVES OF BOUNDARY CONDITIONS AND EQUATIONS WITH RESPECT TO
C      CONTINUATION PARAMETER EPSNU ARE CALCULATED.
   20 DO 30 I = 1,M
         G(I) = 0.
   30 CONTINUE
       DO 40 I = 1,N
         F(1,I) = 0.
         F(2,I) = 0.
         F(3,I) = -2.* (1.-Y(2,I)**2)
   40 CONTINUE
   50 RETURN

       END
       SUBROUTINE JACOB(X,Y,II,C1,M,N,A1,B1,EPSNU)
C      .. Scalar Arguments ..
       INTEGER II,M,N
       DOUBLE PRECISION EPSNU
C      ..
C      .. Array Arguments ..
       DOUBLE PRECISION A1(M,M),B1(M,M),C1(M,M),X(M,N),Y(M,N)
```

```
C     ..
C     .. Local Scalars ..
      INTEGER I,J
C     ..
C     .. Executable Statements ..
C JACOBIAN AT POINT X(II) CALCULATED.
C BOUNDARY CONDITIONS IN A1,B1.
C EQUATIONS IN C1.
      DO 20 I = 1,M
        DO 10 J = 1,M
          A1(I,J) = 0.
          B1(I,J) = 0.
          C1(I,J) = 0.
   10   CONTINUE
   20 CONTINUE
      A1(1,1) = 1.
      A1(2,2) = 1.
      B1(3,2) = 1.
      C1(1,2) = 1.
      C1(2,3) = 1.
      C1(3,1) = -Y(3,II)
      C1(3,2) = 4.*Y(2,II)*EPSNU
      C1(3,3) = -Y(1,II)
      RETURN

      END
```

Giving the output:

```
 SOLUTION ON FINAL GRID OF 52 POINTS
 ERROR IS   2.60412821339564D-08  1.92579319203505D-08  5.17216103169783D-08

 SOLUTION IS
        X(I)             Y1(I)             Y2(I)             Y3(I)
        0.000        0.000000000D+00   0.000000000D+00   1.687218155D+00
        0.625        3.214033325D-03   1.015501031D-01   1.562548753D+00
        1.250        1.253238218D-02   1.953574594D-01   1.439732267D+00
        1.875        2.747578534D-02   2.815869305D-01   1.320278576D+00
        2.500        4.757800728D-02   3.604869331D-01   1.205358186D+00
        3.750        1.014854348D-01   4.975927064D-01   9.923426636D-01
        5.000        1.709315414D-01   6.096369547D-01   8.047733911D-01
        6.250        2.529869414D-01   6.998954068D-01   6.437637932D-01
        7.031        3.095393935D-01   7.467085600D-01   5.562953777D-01
        7.812        3.694920855D-01   7.870664853D-01   4.784201519D-01
        9.375        4.977506513D-01   8.512757030D-01   3.490178275D-01
       10.938        6.345923324D-01   8.977260633D-01   2.501673732D-01
       12.500        7.775923492D-01   9.307521856D-01   1.762751200D-01
       14.583        9.747741598D-01   9.598173586D-01   1.076745526D-01
       16.667        1.176718482D+00   9.773307799D-01   6.383951305D-02
       18.750        1.381493711D+00   9.875734549D-01   3.672864262D-02
       20.312        1.536192487D+00   9.922344864D-01   2.378169562D-02
       21.875        1.691480566D+00   9.952282616D-01   1.513218790D-02
       25.000        2.003042667D+00   9.982890925D-01   5.807564465D-03
       26.562        2.159085732D+00   9.990022698D-01   3.500894051D-03
       28.125        2.315216038D+00   9.994285563D-01   2.071534132D-03
       31.250        2.627610134D+00   9.998226049D-01   6.852207349D-04
       34.375        2.940078234D+00   9.999489218D-01   2.097583423D-04
       35.937        3.096322360D+00   9.999733705D-01   1.126654762D-04
       37.500        3.252569324D+00   9.999863852D-01   5.931383221D-05
       40.625        3.565067021D+00   9.999966450D-01   1.547213947D-05
       43.750        3.877566471D+00   9.999992382D-01   3.710834811D-06
       45.312        4.033816388D+00   9.999996477D-01   1.762076969D-06
```

```
 46.875       4.190066350D+00   9.999998403D-01   8.196241765D-07
 48.437       4.346316333D+00   9.999999291D-01   3.731346768D-07
 50.000       4.502566325D+00   9.999999689D-01   1.679226604D-07
 51.562       4.658816322D+00   9.999999862D-01   7.625009693D-08
 53.125       4.815066320D+00   9.999999945D-01   3.108223815D-08
 56.250       5.127566318D+00   1.000000005D+00  -2.831023804D-08
 57.812       5.283816319D+00   1.000000001D+00  -8.122663627D-09
 59.375       5.440066319D+00   9.999999993D-01   3.514575563D-09
 62.500       5.752566320D+00   9.999999973D-01   1.908671580D-08
 64.062       5.908816319D+00   1.000000000D+00   1.527460495D-09
 65.625       6.065066319D+00   1.000000001D+00  -5.238010477D-09
 68.750       6.377566319D+00   1.000000000D+00  -1.795257767D-09
 71.875       6.690066320D+00   9.999999969D-01   2.165989930D-08
 75.000       7.002566319D+00   1.000000001D+00  -1.036804953D-08
 78.125       7.315066320D+00   1.000000000D+00  -1.411208386D-09
 81.250       7.627566320D+00   9.999999996D-01   3.254622076D-09
 84.375       7.940066320D+00   1.000000000D+00  -1.557810450D-09
 87.500       8.252566320D+00   1.000000000D+00   6.629564539D-10
 90.625       8.565066320D+00   9.999999999D-01   1.638836705D-09
 93.750       8.877566320D+00   1.000000000D+00  -3.562857031D-09
 95.312       9.033816320D+00   1.000000000D+00  -2.178407919D-10
 96.875       9.190066320D+00   1.000000000D+00   5.417638414D-10
 98.437       9.346316320D+00   1.000000000D+00  -1.668680988D-10
100.000       9.502566320D+00   1.000000000D+00   1.190170257D-09
```