NAG Library Routine Document

E02BFF

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

E02BFF evaluates a cubic spline and up to its first three derivatives from its B-spline representation at a vector of points. E02BFF can be used to compute the values and derivatives of cubic spline fits and interpolants produced by reference to E01BAF, E02BAF and E02BEF.

2 Specification

```
SUBROUTINE E02BFF (START, NCAP7, LAMDA, C, DERIV, XORD, X, IXLOC, NX, S,    &
                   LDS, IWRK, LIWRK, IFAIL)
INTEGER            START, NCAP7, DERIV, XORD, IXLOC(NX), NX, LDS,           &
                   IWRK(LIWRK), LIWRK, IFAIL
REAL (KIND=nag_wp) LAMDA(NCAP7), C(NCAP7), X(NX), S(LDS,*)
```

3 Description

E02BFF evaluates the cubic spline $s(x)$ and optionally derivatives up to order 3 for a vector of points x_j, for $j = 1, 2, \ldots, n_x$. It is assumed that $s(x)$ is represented in terms of its B-spline coefficients c_i, for $i = 1, 2, \ldots, \bar{n} + 3$, and (augmented) ordered knot set λ_i, for $i = 1, 2, \ldots, \bar{n} + 7$, (see E02BAF and E02BEF), i.e.,

$$s(x) = \sum_{i=1}^{q} c_i N_i(x).$$

Here $q = \bar{n} + 3$, $\bar{n}$ is the number of intervals of the spline and $N_i(x)$ denotes the normalized B-spline of degree 3 (order 4) defined upon the knots $\lambda_i, \lambda_{i+1}, \ldots, \lambda_{i+4}$. The knots $\lambda_5, \lambda_6, \ldots, \lambda_{\bar{n}+3}$ are the interior knots. The remaining knots, λ_1, λ_2, λ_3, λ_4 and $\lambda_{\bar{n}+4}$, $\lambda_{\bar{n}+5}$, $\lambda_{\bar{n}+6}$, λ_{n+7} are the exterior knots. The knots λ_4 and $\lambda_{\bar{n}+4}$ are the boundaries of the spline.

Only abscissae satisfying,

$$\lambda_4 \leq x_j \leq \lambda_{\bar{n}+4},$$

will be evaluated. At a simple knot λ_i (i.e., one satisfying $\lambda_{i-1} < \lambda_i < \lambda_{i+1}$), the third derivative of the spline is, in general, discontinuous. At a multiple knot (i.e., two or more knots with the same value), lower derivatives, and even the spline itself, may be discontinuous. Specifically, at a point $x = u$ where (exactly) r knots coincide (such a point is termed a knot of multiplicity r), the values of the derivatives of order $4 - j$, for $j = 1, 2, \ldots, r$, are, in general, discontinuous. (Here $1 \leq r \leq 4$; $r > 4$ is not meaningful.) The maximum order of the derivatives to be evaluated D_{ord}, and the left- or right-handedness of the computation when an abscissa corresponds exactly to an interior knot, are determined by the value of DERIV.

Each abscissa (point at which the spline is to be evaluated) x_j contained in X has an associated enclosing interval number, $ixloc_j$ either supplied or returned in IXLOC (see argument START). A simple call to E02BFF would set START $= 0$ and the contents of IXLOC need never be set nor referenced, and the following description on modes of operation can be ignored. However, where efficiency is an important consideration, the following description will help to choose the appropriate mode of operation.

The interval numbers are used to determine which B-splines must be evaluated for a given abscissa, and are defined as

$$ixloc_j = \begin{pmatrix} \leq 0 & x_j < \lambda_1 & \\ 4 & \lambda_4 = x_j & \\ k & \lambda_k < x_j < \lambda_{k+1} & \\ k & \lambda_4 < \lambda_k = x_j & \text{left derivatives} \\ k & x_j = \lambda_{k+1} < \lambda_{\bar{n}+4} & \text{right derivatives or no derivatives} \\ \bar{n}+4 & \lambda_{\bar{n}+4} = x_j & \\ > \bar{n}+7 & x_j > \lambda_{\bar{n}+7} & \end{pmatrix} \tag{1}$$

The algorithm has two modes of vectorization, termed here sorted and unsorted, which are selectable by the argument START.

Furthermore, if the supplied abscissae are sufficiently ordered, as indicated by the argument XORD, the algorithm will take advantage of significantly faster methods for the determination of both the interval numbers and the subsequent spline evaluations.

The sorted mode has two phases, a sorting phase and an evaluation phase. This mode is recommended if there are many abscissae to evaluate relative to the number of intervals of the spline, or the abscissae are distributed relatively densely over a subsection of the spline. In the first phase, $ixloc_j$ is determined for each x_j and a permutation is calculated to sort the x_j by interval number. The first phase may be either partially or completely by-passed using the argument START if the enclosing segments and/or the subsequent ordering are already known *a priori*, for example if multiple spline coefficients C are to be evaluated over the same set of knots LAMDA.

In the second phase of the sorted mode, spline approximations are evaluated by segment, so that non-abscissa dependent calculations over a segment may be reused in the evaluation for all abscissae belonging to a specific segment. For example, all third derivatives of all abscissae in the same segment will be identical.

In the unsorted mode of vectorization, no *a priori* segment sorting is performed, and if the abscissae are not sufficiently ordered, the evaluation at an abscissa will be independent of evaluations at other abscissae; also non-abscissa dependent calculations over a segment will be repeated for each abscissa in a segment. This may be quicker if the number of abscissa is small in comparison to the number of knots in the spline, and they are distributed sparsely throughout the domain of the spline. This is effectively a direct vectorization of E02BBF and E02BCF, although if the enclosing interval numbers $ixloc_j$ are known, these may again be provided.

If the abscissae are sufficiently ordered, then once the first abscissa in a segment is known, an efficient algorithm will be used to determine the location of the final abscissa in this segment. The spline will subsequently be evaluated in a vectorized manner for all the abscissae indexed between the first and last of the current segment.

If no derivatives are required, the spline evaluation is calculated by taking convex combinations due to de Boor (1972). Otherwise, the calculation of $s(x)$ and its derivatives is based upon,

(i) evaluating the nonzero B-splines of orders 1, 2, 3 and 4 by recurrence (see Cox (1972) and Cox (1978)),

(ii) computing all derivatives of the B-splines of order 4 by applying a second recurrence to these computed B-spline values (see de Boor (1972)),

(iii) multiplying the fourth-order B-spline values and their derivative by the appropriate B-spline coefficients, and summing, to yield the values of $s(x)$ and its derivatives.

The method of convex combinations is significantly faster than the recurrence based method. If higher derivatives of order 2 or 3 are not required, as much computation as possible is avoided.

4 References

Cox M G (1972) The numerical evaluation of B-splines *J. Inst. Math. Appl.* **10** 134–149

Cox M G (1978) The numerical evaluation of a spline from its B-spline representation *J. Inst. Math. Appl.* **21** 135–143

de Boor C (1972) On calculating with B-splines *J. Approx. Theory* **6** 50–62

5 Arguments

1: START – INTEGER *Input*

On entry: indicates the completion state of the first phase of the algorithm.

START = 0
: The enclosing interval numbers $ixloc_j$ for the abscissae x_j contained in X have not been determined, and you wish to use the sorted mode of vectorization.

START = 1
: The enclosing interval numbers $ixloc_j$ have been determined and are provided in IXLOC, however the required permutation and interval related information has not been determined and you wish to use the sorted mode of vectorization.

START = 2
: You wish to use the sorted mode of vectorization, and the entire first phase has been completed, with the enclosing interval numbers supplied in IXLOC, and the required permutation and interval related information provided in IWRK (from a previous call to E02BFF).

START = 10
: The enclosing interval numbers $ixloc_j$ for the abscissae x_j contained in X have not been determined, and you wish to use the unsorted mode of vectorization.

START = 11
: The enclosing interval numbers $ixloc_j$ for the abscissae x_j contained in X have been supplied in IXLOC, and you wish to use the unsorted mode of vectorization.

Constraint: START = 0, 1, 2, 10 or 11.

Additional: START = 0 or 10 should be used unless you are sure that the knot set is unchanged between calls.

2: NCAP7 – INTEGER *Input*

On entry: $\bar{n} + 7$, where $\bar{n}$ is the number of intervals of the spline (which is one greater than the number of interior knots, i.e., the knots strictly within the range λ_4 to $\lambda_{\bar{n}+4}$ over which the spline is defined). Note that if E02BEF was used to generate the knots and spline coefficients then NCAP7 should contain the same value as returned in N by E02BEF.

Constraint: NCAP7 ≥ 8.

3: LAMDA(NCAP7) – REAL (KIND=nag_wp) array *Input*

On entry: LAMDA(j) must be set to the value of the jth member of the complete set of knots, λ_j, for $j = 1, 2, \ldots, \bar{n} + 7$.

Constraint: the LAMDA(j) must be in nondecreasing order with LAMDA(NCAP7 $-$ 3) $>$ LAMDA(4).

4: C(NCAP7) – REAL (KIND=nag_wp) array *Input*

On entry: the coefficient c_i of the B-spline $N_i(x)$, for $i = 1, 2, \ldots, \bar{n} + 3$. The remaining elements of the array are not referenced.

5: DERIV – INTEGER *Input*

On entry: the order of derivatives required.

If DERIV < 0 left derivatives are calculated, otherwise right derivatives are calculated. For abscissae satisfying $x_j = \lambda_4$ or $x_j = \lambda_{\bar{n}+4}$ only right-handed or left-handed computation will be used respectively. For abscissae which do not coincide exactly with a knot, the handedness of the computation is immaterial.

DERIV $= 0$
No derivatives required.

DERIV $= \pm 1$
Only $s(x)$ and its first derivative are required.

DERIV $= \pm 2$
Only $s(x)$ and its first and second derivatives are required.

DERIV $= \pm 3$
$s(x)$ and its first, second and third derivatives are required.

Note: if |DERIV| is greater than 3 only the derivatives up to and including 3 will be returned.

6: XORD – INTEGER *Input*

On entry: indicates whether X is supplied in a sufficiently ordered manner. If X is sufficiently ordered E02BFF will complete faster.

XORD $\neq 0$
The abscissae in X are ordered at least by ascending interval, in that any two abscissae contained in the same interval are only separated by abscissae in the same interval, and the intervals are arranged in ascending order. For example, $x_j < x_{j+1}$, for $j = 1, 2, \ldots, \mathrm{NX} - 1$.

XORD $= 0$
The abscissae in X are not sufficiently ordered.

7: X(NX) – REAL (KIND=nag_wp) array *Input*

On entry: the abscissae x_j, for $j = 1, 2, \ldots, n_x$. If START $= 0$ or 10 then evaluations will only be performed for these x_j satisfying $\lambda_4 \leq x_j \leq \lambda_{\bar{n}+4}$. Otherwise evaluation will be performed unless the corresponding element of IXLOC contains an invalid interval number. Please note that if the IXLOC(j) is a valid interval number then no check is made that X(j) actually lies in that interval.

Constraint: at least one abscissa must fall between LAMDA(4) and LAMDA(NCAP7 $-$ 3).

8: IXLOC(NX) – INTEGER array *Input/Output*

On entry: if START $= 1$, 2 or 11, if you wish x_j to be evaluated, IXLOC(j) must be the enclosing interval number $ixloc_j$ of the abscissae x_j (see (1)). If you do not wish x_j to be evaluated, you may set the interval number to be either less than 4 or greater than $\bar{n} + 4$.

Otherwise, IXLOC need not be set.

On exit: if START $= 1$, 2 or 11, IXLOC is unchanged on exit.

Otherwise, IXLOC(j), contains the enclosing interval number $ixloc_j$, for the abscissa supplied in X(j), for $j = 1, 2, \ldots, n_x$. Evaluations will only be performed for abscissae x_j satisfying $\lambda_4 \leq x_j \leq \lambda_{\bar{n}+4}$. If evaluation is not performed IXLOC(j) is set to 0 if $x_j < \lambda_4$ or $\bar{n} + 7$ if $x_j > \lambda_{\bar{n}+4}$.

Constraint: if START $= 1$, 2 or 11, at least one element of IXLOC must be between 4 and NCAP7 $-$ 3.

9: NX – INTEGER *Input*

On entry: n_x, the total number of abscissae contained in X, including any that will not be evaluated.

Constraint: NX ≥ 1.

10: S(LDS, $*$) – REAL (KIND=nag_wp) array *Output*

Note: the second dimension of the array S must be at least $D_{\text{ord}} + 1$, see DERIV for the definition of D_{ord}.

On exit: if x_j is valid, S(j, d) will contain the $(d-1)$th derivative of $s(x)$, for $d = 1, 2, \ldots, D_{\text{ord}} + 1$ and $j = 1, 2, \ldots, n_x$. In particular, S$(j, 1)$ will contain the approximation of $s\big(x_j\big)$ for all legal values in X.

11: LDS – INTEGER *Input*

On entry: the first dimension of the array S as declared in the (sub)program from which E02BFF is called.

Constraint: LDS $\geq$ NX, regardless of the acceptability of the elements of X.

12: IWRK(LIWRK) – INTEGER array *Input/Output*

On entry: if START $= 2$, IWRK must be unchanged from a previous call to E02BFF with START $= 0$ or 1.

Otherwise, IWRK need not be set.

On exit: if START $= 10$ or 11, IWRK is unchanged on exit.

Otherwise, IWRK contains the required permutation of elements of X, if any, and information related to the division of the abscissae x_j between the intervals derived from LAMDA.

13: LIWRK – INTEGER *Input*

On entry: the dimension of the array IWRK as declared in the (sub)program from which E02BFF is called.

Constraint: if START $= 0$, 1 or 2, LIWRK $\geq 3 + 3 \times$ NX.

14: 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, because for this routine the values of the output arguments 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.**

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).

Note: E02BFF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, at least one element of X has an enclosing interval number in IXLOC outside the set allowed by the provided spline. The spline has been evaluated for all X with enclosing interval numbers inside the allowable set.
⟨*value*⟩ entries of X were indexed below the lower bound ⟨*value*⟩.
⟨*value*⟩ entries of X were indexed above the upper bound ⟨*value*⟩.

IFAIL $= 2$

On entry, all elements of X had enclosing interval numbers in IXLOC outside the domain allowed by the provided spline.
$\langle value \rangle$ entries of X were indexed below the lower bound $\langle value \rangle$.
$\langle value \rangle$ entries of X were indexed above the upper bound $\langle value \rangle$.

IFAIL $= 11$

On entry, START $= \langle value \rangle$.
Constraint: START $= 0$, 1, 2, 10 or 11.

IFAIL $= 12$

On entry, START $= 2$ and NX is not consistent with the previous call to E02BFF.
On entry, NX $= \langle value \rangle$.
Constraint: NX $= \langle value \rangle$.

IFAIL $= 21$

On entry, NCAP7 $= \langle value \rangle$.
Constraint: NCAP7 ≥ 8.

IFAIL $= 31$

On entry, LAMDA$(4) = \langle value \rangle$, NCAP7 $= \langle value \rangle$ and LAMDA$(\text{NCAP7} - 3) = \langle value \rangle$.
Constraint: LAMDA$(4) <$ LAMDA$(\text{NCAP7} - 3)$.

IFAIL $= 91$

On entry, NX $= \langle value \rangle$.
Constraint: NX ≥ 1.

IFAIL $= 111$

On entry, LDS $= \langle value \rangle$.
Constraint: LDS $\geq$ NX $= \langle value \rangle$.

IFAIL $= 131$

On entry, LIWRK $= \langle value \rangle$.
Constraint: LIWRK $\geq 3 \times$ NX $+ 3 = \langle value \rangle$.

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 computed value of $s(x)$ has negligible error in most practical situations. Specifically, this value has an absolute error bounded in modulus by $18 \times cmax \times$ ***machine precision***, where $cmax$ is the largest in modulus of c_j, $c_j + 1$, $c_j + 2$ and $c_j + 3$, and j is an integer such that $\lambda_j + 3 < x \leq \lambda_j + 4$. If c_j, $c_j + 1$, $c_j + 2$ and $c_j + 3$ are all of the same sign, then the computed value of $s(x)$ has relative error bounded by $20 \times$ ***machine precision***. For full details see Cox (1978).

No complete error analysis is available for the computation of the derivatives of $s(x)$. However, for most practical purposes the absolute errors in the computed derivatives should be small. Note that this is in comparison to the derivatives of the spline, which may or may not be comparable to the derivatives of the function that has been approximated by the spline.

8 Parallelism and Performance

E02BFF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

If using the sorted mode of vectorization, the time required for the first phase to determine the enclosing intervals is approximately proportional to $O(n_x \log(\bar{n}))$. The time required to then generate the required permutations and interval information is $O(n_x)$ if X is ordered sufficiently, or at worst $O(n_x \min(n_x, \bar{n}) \log(\min(n_x, \bar{n})))$ if X is not ordered. The time required by the second phase is then proportional to $O(n_x)$.

If using the unsorted mode of vectorization, the time required is proportional to $O(n_x \log(\bar{n}))$ if the enclosing interval numbers are not provided, or $O(n_x)$ if they are provided. However, the repeated calculation of various quantities will typically make this slower than the sorted mode when the ratio of abscissae to knots is high, or the abscissae are densely distributed over a relatively small subset of the intervals of the spline.

Note: the routine does not test all the conditions on the knots given in the description of LAMDA in Section 5, since to do this would result in a computation time with a linear dependency upon $\bar{n}$ instead of $\log(\bar{n})$. All the conditions are tested in E02BAF and E02BEF, however.

10 Example

This example fits a spline through a set of data points using E02BEF and then evaluates the spline at a set of supplied abscissae.

10.1 Program Text

```
    Program e02bffe

!     E02BFF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: e02bef, e02bff, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: fp, sfac
      Integer                          :: deriv, ifail, ifail_e02bef, lds,     &
                                          liwrk, lwrk, m, ncap7, nest, nx, r,  &
```

```
                                       sd2, start, xord
      Character (1)                 :: cstart
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: c(:), lamda(:), s(:,:), wdata(:),    &
                                       wrk(:), x(:), xdata(:), ydata(:)
      Integer, Allocatable          :: iwrk(:), ixloc(:)
!     .. Intrinsic Procedures ..
      Intrinsic                     :: abs, min
!     .. Executable Statements ..
      Write (nout,*) 'E02BFF Example Program Results'

!     Skip heading in data file
      Read (nin,*)

!     Input the number of data points for the spline,
!     followed by the data points (XDATA), the function values (YDATA)
!     and the weights (WDATA).

      Read (nin,*) m
      nest = m + 4
      lwrk = 4*m + 16*nest + 41
!     allocate memory for generating the spline
      Allocate (xdata(m),ydata(m),wdata(m),iwrk(nest),lamda(nest),wrk(lwrk),   &
        c(nest))

      Read (nin,*)(xdata(r),ydata(r),wdata(r),r=1,m)

      cstart = 'C'

!     Read in the requested smoothing factor.
      Read (nin,*) sfac

!     Determine the spline approximation.

      ifail_e02bef = 0
      Call e02bef(cstart,m,xdata,ydata,wdata,sfac,nest,ncap7,lamda,c,fp,wrk,   &
        lwrk,iwrk,ifail_e02bef)
      If (ifail_e02bef/=0) Then
        Write (nout,99997)                                                     &
          'Failed to generate spline using data set provided.'
        Write (nout,99996) 'E02BEF returned IFAIL = ', ifail_e02bef
        Go To 100
      End If
      Deallocate (iwrk)

!     Read in the number of sample points requested.
      Read (nin,*) nx

!     Allocate memory for sample point locations and
!      function and derivative approximations.
      lds = nx
      liwrk = 3 + 3*nx
      Allocate (x(nx),s(lds,4),ixloc(nx),iwrk(liwrk))

!     Read in sample points.
      Read (nin,*) x(1:nx)

      xord = 0
      start = 0
      deriv = 3
      ifail = 1
      Call e02bff(start,ncap7,lamda,c,deriv,xord,x,ixloc,nx,s,lds,iwrk,liwrk,  &
        ifail)
      If (ifail>1) Then
        Write (nout,99996) ' E02BFF detected a fatal error. IFAIL =  ', ifail
        Go To 100
      End If

!     Output the results.
      Write (nout,*)
      Write (nout,99999)
```

```
      sd2 = min(abs(deriv),3) + 1
      Do r = 1, nx
        If (ixloc(r)>=4 .And. ixloc(r)<=ncap7-3) Then
          Write (nout,99998) x(r), ixloc(r), s(r,1:sd2)
        Else
          Write (nout,99998) x(r), ixloc(r)
        End If
      End Do

100   Continue
99999 Format (                                                          &
        '        x   ixloc         s(x)        ds/dx      d2s/dx2      d3s/dx3'&
        )
99998 Format (1X,F8.4,3X,I5,4(1X,Es12.4))
99997 Format (1X,A)
99996 Format (1X,A,1X,I5)
    End Program e02bffe
```

10.2 Program Data

```
E02BFF Example Program Data
  15                              : M, the number of data points.
  0.0000E+00  -1.1000E+00   1.00
  5.0000E-01  -3.7200E-01   1.00
  1.0000E+00   4.3100E-01   1.50
  1.5000E+00   1.6900E+00   1.00
  2.0000E+00   2.1100E+00   1.00
  2.5000E+00   3.1000E+00   1.00
  3.0000E+00   4.2300E+00   1.00
  4.0000E+00   4.3500E+00   1.00
  4.5000E+00   4.8100E+00   1.00
  5.0000E+00   4.6100E+00   1.00
  5.5000E+00   4.7900E+00   1.00
  6.0000E+00   5.2300E+00   1.00
  7.0000E+00   6.3500E+00   1.00
  7.5000E+00   7.1900E+00   1.00
  8.0000E+00   7.9700E+00   1.00  : xdata(1:m), ydata(1:m), wdata(1:m)
  0.001                           : S, smoothing factor.

  20                              : NX, the number of evaluation points.
  6.5178  7.2463  1.0159  7.3070
  5.0589  0.7803  2.2280  4.3751
  7.6601  7.7191  1.2609  7.7647
  7.6573  3.8830  6.4022  1.1351
  3.3741  7.3259  6.3377  7.6759  : Unordered evaluation points x(1:nx).
```

10.3 Program Results

```
 E02BFF Example Program Results

       x   ixloc         s(x)        ds/dx      d2s/dx2      d3s/dx3
  6.5178      14   5.7418E+00   1.0741E+00   5.6736E-01   1.3065E+00
  7.2463      15   6.7486E+00   1.7074E+00   4.9054E-01  -2.8697E+00
  1.0159       5   4.7469E-01   2.4179E+00   3.8175E+00  -2.2171E+01
  7.3070      15   6.8531E+00   1.7319E+00   3.1634E-01  -2.8697E+00
  5.0589      12   4.6105E+00  -1.0363E-01   2.9075E+00  -4.4467E+00
  0.7803       4   6.6885E-03   1.6216E+00   2.5007E+00   7.5980E+00
  2.2280       7   2.4751E+00   1.9559E+00   3.0615E+00  -6.6690E+00
  4.3751      10   4.7199E+00   8.5194E-01  -3.0718E+00  -1.9866E+01
  7.6601      15   7.4633E+00   1.6647E+00  -6.9696E-01  -2.8697E+00
  7.7191      15   7.5602E+00   1.6186E+00  -8.6627E-01  -2.8697E+00
  1.2609       5   1.1273E+00   2.6878E+00  -1.6146E+00  -2.2171E+01
  7.7647      15   7.6330E+00   1.5761E+00  -9.9713E-01  -2.8697E+00
  7.6573      15   7.4586E+00   1.6667E+00  -6.8892E-01  -2.8697E+00
  3.8830       9   4.3152E+00   1.6458E-01   3.1754E+00   1.0296E+01
  6.4022      14   5.6211E+00   1.0172E+00   4.1633E-01   1.3065E+00
```

```
   1.1351       5   7.8376E-01   2.7154E+00   1.1746E+00  -2.2171E+01
   3.3741       9   4.4165E+00  -1.1809E-01  -2.0644E+00   1.0296E+01
   7.3259      15   6.8859E+00   1.7374E+00   2.6211E-01  -2.8697E+00
   6.3377      14   5.5563E+00   9.9310E-01   3.3206E-01   1.3065E+00
   7.6759      15   7.4895E+00   1.6534E+00  -7.4230E-01  -2.8697E+00
```