NAG Library Routine Document

E02BEF

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

E02BEF computes a cubic spline approximation to an arbitrary set of data points. The knots of the spline are located automatically, but a single argument must be specified to control the trade-off between closeness of fit and smoothness of fit.

2 Specification

```
SUBROUTINE E02BEF (START, M, X, Y, W, S, NEST, N, LAMDA, C, FP, WRK,       &
                   LWRK, IWRK, IFAIL)

INTEGER             M, NEST, N, LWRK, IWRK(NEST), IFAIL
REAL (KIND=nag_wp) X(M), Y(M), W(M), S, LAMDA(NEST), C(NEST), FP,          &
                   WRK(LWRK)
CHARACTER(1)        START
```

3 Description

E02BEF determines a smooth cubic spline approximation $s(x)$ to the set of data points (x_r, y_r), with weights w_r, for $r = 1, 2, \ldots, m$.

The spline is given in the B-spline representation

$$s(x) = \sum_{i=1}^{n-4} c_i N_i(x), \tag{1}$$

where $N_i(x)$ denotes the normalized cubic B-spline defined upon the knots $\lambda_i, \lambda_{i+1}, \ldots, \lambda_{i+4}$.

The total number n of these knots and their values $\lambda_1, \ldots, \lambda_n$ are chosen automatically by the routine. The knots $\lambda_5, \ldots, \lambda_{n-4}$ are the interior knots; they divide the approximation interval $[x_1, x_m]$ into $n - 7$ sub-intervals. The coefficients $c_1, c_2, \ldots, c_{n-4}$ are then determined as the solution of the following constrained minimization problem:

minimize

$$\eta = \sum_{i=5}^{n-4} \delta_i^2 \tag{2}$$

subject to the constraint

$$\theta = \sum_{r=1}^{m} \epsilon_r^2 \leq S, \tag{3}$$

where δ_i stands for the discontinuity jump in the third order derivative of $s(x)$ at the interior knot λ_i,
ϵ_r denotes the weighted residual $w_r(y_r - s(x_r))$,
and S is a non-negative number to be specified by you.

The quantity η can be seen as a measure of the (lack of) smoothness of $s(x)$, while closeness of fit is measured through θ. By means of the argument S, 'the smoothing factor', you can then control the balance between these two (usually conflicting) properties. If S is too large, the spline will be too smooth and signal will be lost (underfit); if S is too small, the spline will pick up too much noise (overfit). In the extreme cases the routine will return an interpolating spline ($\theta = 0$) if S is set to zero,

and the weighted least squares cubic polynomial ($\eta = 0$) if S is set very large. Experimenting with S values between these two extremes should result in a good compromise. (See Section 9.2 for advice on choice of S.)

The method employed is outlined in Section 9.3 and fully described in Dierckx (1975), Dierckx (1981) and Dierckx (1982). It involves an adaptive strategy for locating the knots of the cubic spline (depending on the function underlying the data and on the value of S), and an iterative method for solving the constrained minimization problem once the knots have been determined.

Values of the computed spline, or of its derivatives or definite integral, can subsequently be computed by calling E02BBF, E02BCF or E02BDF, as described in Section 9.4.

4 References

Dierckx P (1975) An algorithm for smoothing, differentiating and integration of experimental data using spline functions *J. Comput. Appl. Math.* **1** 165–184

Dierckx P (1981) An improved algorithm for curve fitting with spline functions *Report TW54* Department of Computer Science, Katholieke Univerciteit Leuven

Dierckx P (1982) A fast algorithm for smoothing data on a rectangular grid while using spline functions *SIAM J. Numer. Anal.* **19** 1286–1304

Reinsch C H (1967) Smoothing by spline functions *Numer. Math.* **10** 177–183

5 Arguments

1: START – CHARACTER(1) *Input*

On entry: must be set to `C' or `W'.

START = 'C'
: The routine will build up the knot set starting with no interior knots. No values need be assigned to the arguments N, LAMDA, WRK or IWRK.

START = 'W'
: The routine will restart the knot-placing strategy using the knots found in a previous call of the routine. In this case, the arguments N, LAMDA, WRK, and IWRK must be unchanged from that previous call. This warm start can save much time in searching for a satisfactory value of S.

Constraint: START = 'C' or 'W'.

2: M – INTEGER *Input*

On entry: m, the number of data points.

Constraint: M ≥ 4.

3: X(M) – REAL (KIND=nag_wp) array *Input*

On entry: the values x_r of the independent variable (abscissa) x, for $r = 1, 2, \ldots, m$.

Constraint: $x_1 < x_2 < \cdots < x_m$.

4: Y(M) – REAL (KIND=nag_wp) array *Input*

On entry: the values y_r of the dependent variable (ordinate) y, for $r = 1, 2, \ldots, m$.

5: W(M) – REAL (KIND=nag_wp) array *Input*

On entry: the values w_r of the weights, for $r = 1, 2, \ldots, m$. For advice on the choice of weights, see Section 2.1.2 in the E02 Chapter Introduction.

Constraint: W$(r) > 0.0$, for $r = 1, 2, \ldots, m$.

6: S – REAL (KIND=nag_wp) *Input*

On entry: the smoothing factor, S.

If $S = 0.0$, the routine returns an interpolating spline.

If S is smaller than ***machine precision***, it is assumed equal to zero.

For advice on the choice of S, see Sections 3 and 9.2.

Constraint: S ≥ 0.0.

7: NEST – INTEGER *Input*

On entry: an overestimate for the number, n, of knots required.

Constraint: NEST ≥ 8. In most practical situations, NEST $= \mathrm{M}/2$ is sufficient. NEST never needs to be larger than M $+ 4$, the number of knots needed for interpolation (S $= 0.0$).

8: N – INTEGER *Input/Output*

On entry: if the warm start option is used, the value of N must be left unchanged from the previous call.

On exit: the total number, n, of knots of the computed spline.

9: LAMDA(NEST) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if the warm start option is used, the values LAMDA(1), LAMDA(2), …, LAMDA(N) must be left unchanged from the previous call.

On exit: the knots of the spline, i.e., the positions of the interior knots LAMDA(5), LAMDA(6), …, LAMDA(N $-$ 4) as well as the positions of the additional knots

$$\mathrm{LAMDA}(1) = \mathrm{LAMDA}(2) = \mathrm{LAMDA}(3) = \mathrm{LAMDA}(4) = x_1$$

and

$$\mathrm{LAMDA}(\mathrm{N}-3) = \mathrm{LAMDA}(\mathrm{N}-2) = \mathrm{LAMDA}(\mathrm{N}-1) = \mathrm{LAMDA}(\mathrm{N}) = x_m$$

needed for the B-spline representation.

10: C(NEST) – REAL (KIND=nag_wp) array *Output*

On exit: the coefficient c_i of the B-spline $N_i(x)$ in the spline approximation $s(x)$, for $i = 1, 2, \ldots, n-4$.

11: FP – REAL (KIND=nag_wp) *Output*

On exit: the sum of the squared weighted residuals, θ, of the computed spline approximation. If FP $= 0.0$, this is an interpolating spline. FP should equal S within a relative tolerance of 0.001 unless $n = 8$ when the spline has no interior knots and so is simply a cubic polynomial. For knots to be inserted, S must be set to a value below the value of FP produced in this case.

12: WRK(LWRK) – REAL (KIND=nag_wp) array *Communication Array*

If the warm start option is used on entry, the values WRK(1), …, WRK(n) must be left unchanged from the previous call.

13: LWRK – INTEGER *Input*

On entry: the dimension of the array WRK as declared in the (sub)program from which E02BEF is called.

Constraint: LWRK $\geq 4 \times$ M $+ 16 \times$ NEST $+ 41$.

14: IWRK(NEST) – INTEGER array *Communication Array*

If the warm start option is used, on entry, the values IWRK$(1), \ldots,$IWRK(n) must be left unchanged from the previous call.

This array is used as workspace.

15: 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, START $\neq$ 'C' or 'W',
or M < 4,
or S < 0.0,
or S $= 0.0$ and NEST $<$ M $+ 4$,
or NEST < 8,
or LWRK $< 4 \times$ M $+ 16 \times$ NEST $+ 41$.

IFAIL $= 2$

The weights are not all strictly positive.

IFAIL $= 3$

The values of X(r), for $r = 1, 2, \ldots,$ M, are not in strictly increasing order.

IFAIL $= 4$

The number of knots required is greater than NEST. Try increasing NEST and, if necessary, supplying larger arrays for the arguments LAMDA, C, WRK and IWRK. However, if NEST is already large, say NEST $>$ M$/2$, then this error exit may indicate that S is too small.

IFAIL $= 5$

The iterative process used to compute the coefficients of the approximating spline has failed to converge. This error exit may occur if S has been set very small. If the error persists with increased S, contact NAG.

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.

If IFAIL $= 4$ or 5, a spline approximation is returned, but it fails to satisfy the fitting criterion (see (2) and (3)) – perhaps by only a small amount, however.

7 Accuracy

On successful exit, the approximation returned is such that its weighted sum of squared residuals θ (as in (3)) is equal to the smoothing factor S, up to a specified relative tolerance of 0.001 – except that if $n = 8$, θ may be significantly less than S: in this case the computed spline is simply a weighted least squares polynomial approximation of degree 3, i.e., a spline with no interior knots.

8 Parallelism and Performance

E02BEF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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

9.1 Timing

The time taken for a call of E02BEF depends on the complexity of the shape of the data, the value of the smoothing factor S, and the number of data points. If E02BEF is to be called for different values of S, much time can be saved by setting START = 'W' after the first call.

9.2 Choice of *S*

If the weights have been correctly chosen (see Section 2.1.2 in the E02 Chapter Introduction), the standard deviation of $w_r y_r$ would be the same for all r, equal to σ, say. In this case, choosing the smoothing factor S in the range $\sigma^2\left(m \pm \sqrt{2m}\right)$, as suggested by Reinsch (1967), is likely to give a good start in the search for a satisfactory value. Otherwise, experimenting with different values of S will be required from the start, taking account of the remarks in Section 3.

In that case, in view of computation time and memory requirements, it is recommended to start with a very large value for S and so determine the least squares cubic polynomial; the value returned in FP, call it θ_0, gives an upper bound for S. Then progressively decrease the value of S to obtain closer fits – say by a factor of 10 in the beginning, i.e., $S = \theta_0/10$, $S = \theta_0/100$, and so on, and more carefully as the approximation shows more details.

The number of knots of the spline returned, and their location, generally depend on the value of S and on the behaviour of the function underlying the data. However, if E02BEF is called with START = 'W', the knots returned may also depend on the smoothing factors of the previous calls. Therefore if, after a number of trials with different values of S and START = 'W', a fit can finally be accepted as satisfactory, it may be worthwhile to call E02BEF once more with the selected value for S but now using START = 'C'. Often, E02BEF then returns an approximation with the same quality of fit but with fewer knots, which is therefore better if data reduction is also important.

9.3 Outline of Method Used

If $S = 0$, the requisite number of knots is known in advance, i.e., $n = m + 4$; the interior knots are located immediately as $\lambda_i = x_{i-2}$, for $i = 5, 6, \ldots, n - 4$. The corresponding least squares spline (see E02BAF) is then an interpolating spline and therefore a solution of the problem.

If $S > 0$, a suitable knot set is built up in stages (starting with no interior knots in the case of a cold start but with the knot set found in a previous call if a warm start is chosen). At each stage, a spline is fitted to the data by least squares (see E02BAF) and θ, the weighted sum of squares of residuals, is computed. If $\theta > S$, new knots are added to the knot set to reduce θ at the next stage. The new knots are located in intervals where the fit is particularly poor, their number depending on the value of S and on the progress made so far in reducing θ. Sooner or later, we find that $\theta \leq S$ and at that point the knot set is accepted. The routine then goes on to compute the (unique) spline which has this knot set and which satisfies the full fitting criterion specified by (2) and (3). The theoretical solution has $\theta = S$. The routine computes the spline by an iterative scheme which is ended when $\theta = S$ within a relative tolerance of 0.001. The main part of each iteration consists of a linear least squares computation of special form, done in a similarly stable and efficient manner as in E02BAF.

An exception occurs when the routine finds at the start that, even with no interior knots $(n = 8)$, the least squares spline already has its weighted sum of squares of residuals $\leq S$. In this case, since this spline (which is simply a cubic polynomial) also has an optimal value for the smoothness measure η, namely zero, it is returned at once as the (trivial) solution. It will usually mean that S has been chosen too large.

For further details of the algorithm and its use, see Dierckx (1981).

9.4 Evaluation of Computed Spline

The value of the computed spline at a given value X may be obtained in the real variable S by the call:

```
CALL E02BBF(N,LAMDA,C,X,S,IFAIL)
```

where N, LAMDA and C are the output arguments of E02BEF.

The values of the spline and its first three derivatives at a given value X may be obtained in the real array S of dimension at least 4 by the call:

```
CALL E02BCF(N,LAMDA,C,X,LEFT,S,IFAIL)
```

where if LEFT $= 1$, left-hand derivatives are computed and if LEFT $\neq 1$, right-hand derivatives are calculated. The value of LEFT is only relevant if X is an interior knot (see E02BCF).

The value of the definite integral of the spline over the interval X(1) to X(M) can be obtained in the real variable DINT by the call:

```
CALL E02BDF(N,LAMDA,C,DINT,IFAIL)
```

(see E02BDF).

10 Example

This example reads in a set of data values, followed by a set of values of S. For each value of S it calls E02BEF to compute a spline approximation, and prints the values of the knots and the B-spline coefficients c_i.

The program includes code to evaluate the computed splines, by calls to E02BBF, at the points x_r and at points mid-way between them. These values are not printed out, however; instead the results are illustrated by plots of the computed splines, together with the data points (indicated by $\times$) and the positions of the knots (indicated by vertical lines): the effect of decreasing S can be clearly seen.

10.1 Program Text

```
    Program e02befe

!     E02BEF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: e02bbf, e02bef, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: fp, s, txr
      Integer                          :: ifail, ioerr, j, lwrk, m, n, nest, r
      Character (1)                    :: start
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: c(:), lamda(:), sp(:), w(:), wrk(:), &
                                          x(:), y(:)
      Integer, Allocatable             :: iwrk(:)
!     .. Executable Statements ..
      Write (nout,*) 'E02BEF Example Program Results'

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

!     Input the number of data points, followed by the data points (X),
!     the function values (Y) and the weights (W).

      Read (nin,*) m
      nest = m + 4
      lwrk = 4*m + 16*nest + 41
      Allocate (x(m),y(m),w(m),iwrk(nest),lamda(nest),wrk(lwrk),c(nest),       &
        sp(2*m-1))

      Do r = 1, m
        Read (nin,*) x(r), y(r), w(r)
      End Do

      start = 'C'

!     Read in successive values of S until end of data file.

data: Do
        Read (nin,*,Iostat=ioerr) s

        If (ioerr<0) Then
          Exit data
        End If

!       Determine the spline approximation.

        ifail = 0
        Call e02bef(start,m,x,y,w,s,nest,n,lamda,c,fp,wrk,lwrk,iwrk,ifail)

!       Evaluate the spline at each X point and midway between
!       X points, saving the results in SP.

        Do r = 1, m

          ifail = 0
          Call e02bbf(n,lamda,c,x(r),sp((r-1)*2+1),ifail)

        End Do

        Do r = 1, m - 1
          txr = (x(r)+x(r+1))/2.0E0_nag_wp

          ifail = 0
```

```
          Call e02bbf(n,lamda,c,txr,sp(r*2),ifail)

        End Do

!       Output the results.

        Write (nout,*)
        Write (nout,99999) 'Calling with smoothing factor S =', s
        Write (nout,*)
        Write (nout,*) '                                                B-Spline'
        Write (nout,*)                                                          &
          '             J       Knot LAMDA(J+2)   Coefficient C(J)'
        Write (nout,99998) 1, c(1)

        Do j = 2, n - 5
          Write (nout,99997) j, lamda(j+2), c(j)
        End Do

        Write (nout,99998) n - 4, c(n-4)
        Write (nout,*)
        Write (nout,99999) 'Weighted sum of squared residuals FP =', fp

        If (fp==0.0E0_nag_wp) Then
          Write (nout,*) '(The spline is an interpolating spline)'
        Else If (n==8) Then
          Write (nout,*)                                                        &
            '(The spline is the weighted least squares cubic polynomial)'
        End If

        Write (nout,*)
        start = 'W'
      End Do data

99999 Format (1X,A,1P,E12.3)
99998 Format (11X,I4,20X,F16.4)
99997 Format (11X,I4,2F18.4)
    End Program e02befe
```

10.2 Program Data

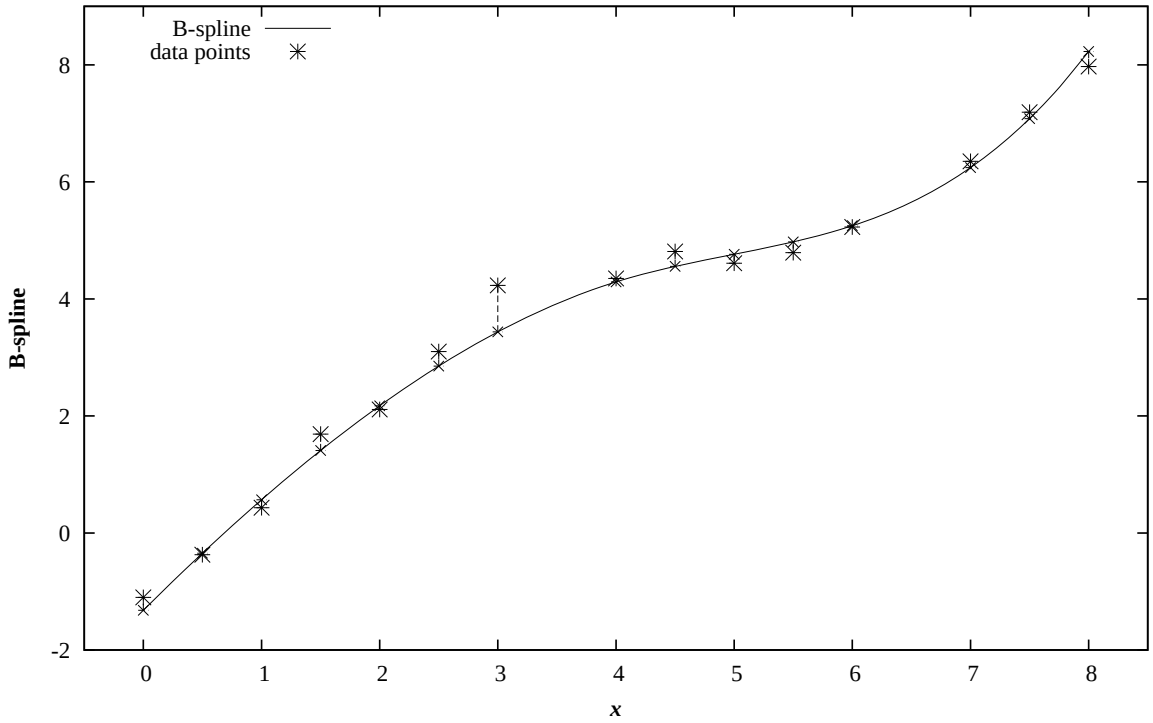
```
E02BEF Example Program Data
  15                            M, the number of data points
  0.0000E+00  -1.1000E+00   1.00  X, Y, W, abscissa, ordinate and weight
  5.0000E-01  -3.7200E-01   2.00
  1.0000E+00   4.3100E-01   1.50
  1.5000E+00   1.6900E+00   1.00
  2.0000E+00   2.1100E+00   3.00
  2.5000E+00   3.1000E+00   1.00
  3.0000E+00   4.2300E+00   0.50
  4.0000E+00   4.3500E+00   1.00
  4.5000E+00   4.8100E+00   2.00
  5.0000E+00   4.6100E+00   2.50
  5.5000E+00   4.7900E+00   1.00
  6.0000E+00   5.2300E+00   3.00
  7.0000E+00   6.3500E+00   1.00
  7.5000E+00   7.1900E+00   2.00
  8.0000E+00   7.9700E+00   1.00  End of data points
  1.0                           S, smoothing factor
  0.5                           S, smoothing factor
  0.1                           S, smoothing factor
```

10.3 Program Results

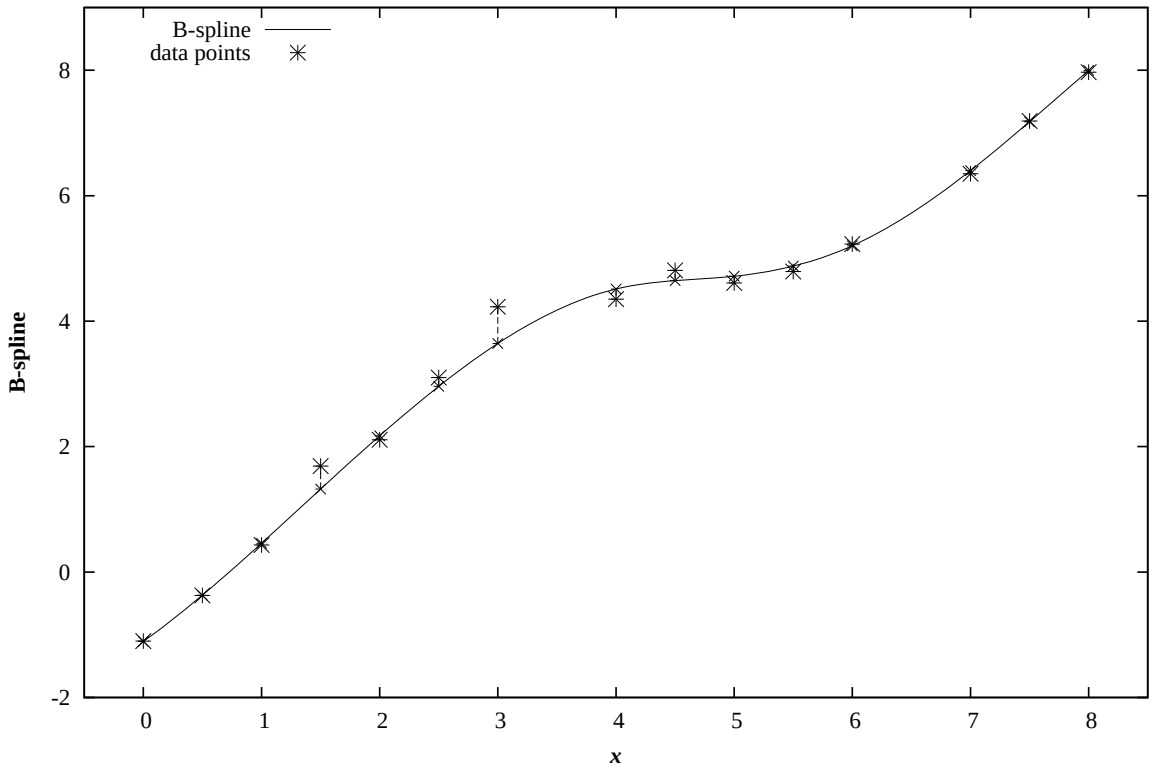
```
 E02BEF Example Program Results

 Calling with smoothing factor S =   1.000E+00

                                        B-Spline
              J       Knot LAMDA(J+2)   Coefficient C(J)
              1                            -1.3201
```

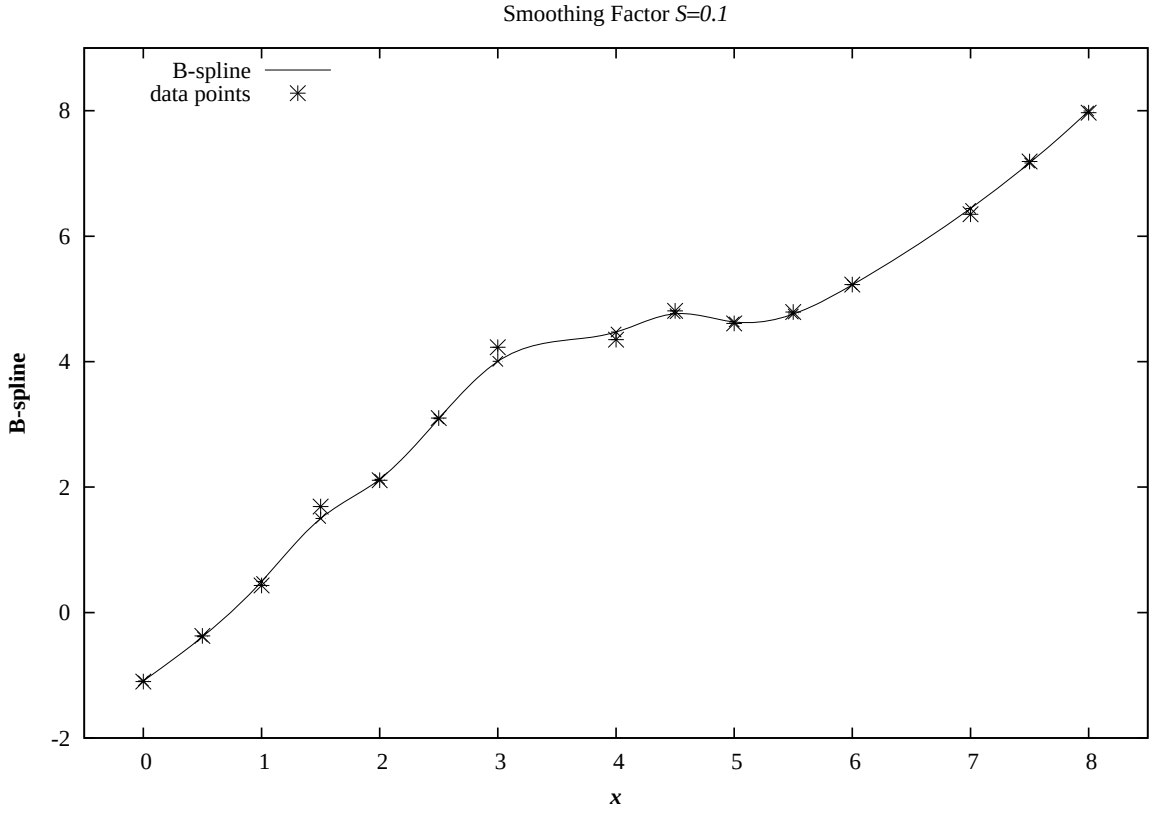
```
             2           0.0000           1.3542
             3           4.0000           5.5510
             4           8.0000           4.7031
             5                            8.2277

 Weighted sum of squared residuals FP =   1.000E+00


 Calling with smoothing factor S =   5.000E-01

                                        B-Spline
             J       Knot LAMDA(J+2)   Coefficient C(J)
             1                           -1.1072
             2           0.0000          -0.6571
             3           1.0000           0.4350
             4           2.0000           2.8061
             5           4.0000           4.6824
             6           5.0000           4.6416
             7           6.0000           5.1976
             8           8.0000           6.9008
             9                            7.9979

 Weighted sum of squared residuals FP =   5.001E-01


 Calling with smoothing factor S =   1.000E-01

                                        B-Spline
             J       Knot LAMDA(J+2)   Coefficient C(J)
             1                           -1.0901
             2           0.0000          -0.6401
             3           1.0000           0.0334
             4           1.5000           1.6390
             5           2.0000           2.1243
             6           3.0000           4.5591
             7           4.0000           4.2174
             8           4.5000           4.9105
             9           5.0000           4.5475
            10           5.5000           4.6960
            11           6.0000           5.7370
            12           8.0000           6.8179
            13                            7.9953

 Weighted sum of squared residuals FP =   9.999E-02
```

Example Program
Calculation and Evaluation of B-splines Representation
Smoothing Factor *S=1.0*



Smoothing Factor *S=0.5*




Smoothing Factor S=0.1
B-spline
data points
B-spline
x
8
6
4
2
0
-2
0
1
2
3
4
5
6
7
8