NAG Library Routine Document

E01TLF

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

E01TLF evaluates the four-dimensional interpolating function generated by E01TKF and its first partial derivatives.

2 Specification

```
SUBROUTINE E01TLF (M, X, F, IQ, RQ, N, XE, Q, QX, IFAIL)
INTEGER            M, IQ(2*M+1), N, IFAIL
REAL (KIND=nag_wp) X(4,M), F(M), RQ(15*M+9), XE(4,N), Q(N), QX(4,N)
```

3 Description

E01TLF takes as input the interpolant $Q(\mathbf{x})$, $x \in \mathbb{R}^4$ of a set of scattered data points $(\mathbf{x}_r, f_r)$, for $r = 1, 2, \ldots, m$, as computed by E01TKF, and evaluates the interpolant and its first partial derivatives at the set of points $\mathbf{x}_i$, for $i = 1, 2, \ldots, n$.

E01TLF must only be called after a call to E01TKF.

E01TLF is derived from the new implementation of QS3GRD described by Renka (1988). It uses the modification for high-dimensional interpolation described by Berry and Minser (1999).

4 References

Berry M W, Minser K S (1999) Algorithm 798: high-dimensional interpolation using the modified Shepard method *ACM Trans. Math. Software* **25** 353–366

Renka R J (1988) Algorithm 661: QSHEP3D: Quadratic Shepard method for trivariate interpolation of scattered data *ACM Trans. Math. Software* **14** 151–152

5 Parameters

1: M – INTEGER *Input*

On entry: **must** be the same value supplied for parameter M in the preceding call to E01TKF.

Constraint: $\mathrm{M} \geq 16$.

2: $\mathrm{X}(4,\mathrm{M})$ – REAL (KIND=nag_wp) array *Input*

Note: the coordinates of x_r are stored in $\mathrm{X}(1,r)\ldots\mathrm{X}(4,r)$.

On entry: **must** be the same array supplied as parameter X in the preceding call to E01TKF. It **must** remain unchanged between calls.

3: $\mathrm{F}(\mathrm{M})$ – REAL (KIND=nag_wp) array *Input*

On entry: **must** be the same array supplied as parameter F in the preceding call to E01TKF. It **must** remain unchanged between calls.

4: IQ$(2 \times \mathrm{M} + 1)$ – INTEGER array *Input*

On entry: **must** be the same array returned as parameter IQ in the preceding call to E01TKF. It **must** remain unchanged between calls.

5: RQ$(15 \times \mathrm{M} + 9)$ – REAL (KIND=nag_wp) array *Input*

On entry: **must** be the same array returned as parameter RQ in the preceding call to E01TKF. It **must** remain unchanged between calls.

6: N – INTEGER *Input*

On entry: n, the number of evaluation points.

Constraint: $\mathrm{N} \geq 1$.

7: XE$(4, \mathrm{N})$ – REAL (KIND=nag_wp) array *Input*

On entry: XE$(1:4, i)$ must be set to the evaluation point $\mathbf{x}_i$, for $i = 1, 2, \ldots, n$.

8: Q(N) – REAL (KIND=nag_wp) array *Output*

On exit: Q(i) contains the value of the interpolant, at $\mathbf{x}_i$, for $i = 1, 2, \ldots, n$. If any of these evaluation points lie outside the region of definition of the interpolant the corresponding entries in Q are set to the largest machine representable number (see X02ALF), and E01TLF returns with $\mathrm{IFAIL} = 3$.

9: QX$(4, \mathrm{N})$ – REAL (KIND=nag_wp) array *Output*

On exit: QX(j, i) contains the value of the partial derivatives with respect to $\mathbf{x}_j$ of the interpolant $Q(\mathbf{x})$ at $\mathbf{x}_i$, for $i = 1, 2, \ldots, n$, and for each of the four partial derivatives $j = 1, 2, 3, 4$. If any of these evaluation points lie outside the region of definition of the interpolant, the corresponding entries in QX are set to the largest machine representable number (see X02ALF), and E01TLF returns with $\mathrm{IFAIL} = 3$.

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.

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 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.**

On exit: $\mathrm{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $\mathrm{IFAIL} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 1$

On entry, $\mathrm{M} = \langle value \rangle$.
Constraint: $\mathrm{M} \geq 16$.

On entry, $\mathrm{N} = \langle value \rangle$.
Constraint: $\mathrm{N} \geq 1$.

IFAIL $= 2$

On entry, values in IQ appear to be invalid. Check that IQ has not been corrupted between calls to E01TKF and E01TLF.

On entry, values in RQ appear to be invalid. Check that RQ has not been corrupted between calls to E01TKF and E01TLF.

IFAIL $= 3$

On entry, at least one evaluation point lies outside the region of definition of the interpolant. At all such points the corresponding values in Q and QX have been set to X02ALF(): X02ALF() $= \langle$*value*$\rangle$.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

Computational errors should be negligible in most practical situations.

8 Parallelism and Performance

E01TLF 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

The time taken for a call to E01TLF will depend in general on the distribution of the data points. If the data points are approximately uniformly distributed, then the time taken should be only $O(n)$. At worst $O(mn)$ time will be required.

10 Example

This program evaluates the function

$$f(x) = \frac{(1.25 + \cos(5.4x_4))\cos(6x_1)\cos(6x_2)}{6 + 6(3x_3 - 1)^2}$$

at a set of 30 randomly generated data points and calls E01TKF to construct an interpolating function $Q(\mathbf{x})$. It then calls E01TLF to evaluate the interpolant at a set of random points.

To reduce the time taken by this example, the number of data points is limited to 30. Increasing this value improves the interpolation accuracy at the expense of more time.

See also Section 10 in E01TKF.

10.1 Program Text

```
!   E01TLF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module e01tlfe_mod

!     E01TLF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                               :: funct
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter        :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter        :: six = 6.0_nag_wp
      Real (Kind=nag_wp), Parameter        :: three = 3.0_nag_wp
      Integer, Parameter, Public           :: nin = 5, nout = 6
    Contains
      Subroutine funct(m,x,f)
!       This subroutine evaluates the 4D function funct.

!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: m
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: f(m)
        Real (Kind=nag_wp), Intent (In)      :: x(4,m)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                   :: c1, c2, c3, c4
        Integer                              :: i
!       .. Intrinsic Procedures ..
        Intrinsic                            :: cos
!       .. Executable Statements ..
        Do i = 1, m
          c1 = cos(six*x(1,i))
          c2 = cos(six*x(2,i))
          c3 = six + six*(three*x(3,i)-one)**2
          c4 = 1.25_nag_wp + cos(5.4_nag_wp*x(4,i))
          f(i) = c4*c1*c2/c3
        End Do
        Return
      End Subroutine funct
    End Module e01tlfe_mod
    Program e01tlfe

!     E01TLF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e01tkf, e01tlf, g05kff, g05saf, nag_wp
      Use e01tlfe_mod, Only: funct, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: lseed = 1
!     .. Local Scalars ..
      Integer                              :: genid, i, ifail, liq, lrq,       &
                                              lstate, m, n, nq, nw, subid
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: f(:), fun(:), q(:), qx(:,:),     &
                                              rq(:), x(:,:), xe(:,:)
      Integer, Allocatable                 :: iq(:), state(:)
      Integer                              :: seed(lseed), seed2(lseed)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: abs
```

```
!     .. Executable Statements ..
      Write (nout,*) 'E01TLF Example Program Results'

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

!     Read in the base generator information and seeds
      Read (nin,*) genid, subid, seed(1), seed2(1)

!     Initial call to initialiser to get size of STATE array
      lstate = 0
      Allocate (state(lstate))
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!     Reallocate STATE
      Deallocate (state)
      Allocate (state(lstate))

!     Initialize the generator to a repeatable sequence
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!     Input the number of nodes.
      Read (nin,*) m
      liq = 2*m + 1
      lrq = 15*m + 9
      Allocate (x(4,m),f(m),iq(liq),rq(lrq))

!     Generate the data points X
      ifail = 0
      Call g05saf(4*m,state,x,ifail)

!     Evaluate F
      Call funct(m,x,f)

!     Generate the interpolant using E01TKF.
      nq = 0
      nw = 0

      ifail = 0
      Call e01tkf(m,x,f,nw,nq,iq,rq,ifail)

!     Input the number of evaluation points.
      Read (nin,*) n
      Allocate (xe(4,n),q(n),qx(4,n),fun(n))

!     Generate repeatable evaluation points.
      ifail = 0
      Call g05kff(genid,subid,seed2,lseed,state,lstate,ifail)
      ifail = 0
      Call g05saf(4*n,state,xe,ifail)

!     Evaluate the interpolant.
      ifail = 0
      Call e01tlf(m,x,f,iq,rq,n,xe,q,qx,ifail)

      Write (nout,99997)
      Write (nout,99998)
      Call funct(n,xe,fun)
      Write (nout,99999)(i,fun(i),q(i),abs(fun(i)-q(i)),i=1,n)

99999 Format (1X,I4,1X,2F10.4,2X,F10.4)
99998 Format (4X,'---+',20('-'),'+',11('-'),'+')
99997 Format (/4X,'I  |',2X,'F(I)',6X,'Q(I)',4X,'|',1X,'|F(I)-Q(I)|')
    End Program e01tlfe
```

10.2 Program Data

```
E01TLF Example Program Data
1  1  1762543  43331            genid, subid, seed(1), seed(2)
30                              M the number of data points
8                               N the number of evaluation points
```

10.3 Program Results

```
 E01TLF Example Program Results

    I  |  F(I)      Q(I)    | |F(I)-Q(I)|
    ---+--------------------+-----------+
    1    -0.0189   -0.0394      0.0205
    2    -0.0186    0.0967      0.1153
    3     0.1147    0.0606      0.0541
    4     0.0096   -0.1313      0.1409
    5    -0.1354   -0.1878      0.0524
    6     0.0022   -0.1595      0.1617
    7    -0.0095   -0.1179      0.1084
    8     0.0113   -0.3950      0.4063
```