NAG Library Routine Document

E01TNF

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

E01TNF evaluates the five-dimensional interpolating function generated by E01TMF and its first partial derivatives.

2 Specification

```
SUBROUTINE E01TNF (M, X, F, IQ, RQ, N, XE, Q, QX, IFAIL)
INTEGER            M, IQ(2*M+1), N, IFAIL
REAL (KIND=nag_wp) X(5,M), F(M), RQ(21*M+11), XE(5,N), Q(N), QX(5,N)
```

3 Description

E01TNF takes as input the interpolant $Q(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^5$ of a set of scattered data points $(\mathbf{x}_r, f_r)$, for $r = 1, 2, \ldots, m$, as computed by E01TMF, and evaluates the interpolant and its first partial derivatives at the set of points $\mathbf{x}_i$, for $i = 1, 2, \ldots, n$.

E01TNF must only be called after a call to E01TMF.

E01TNF is derived from the new implementation of QS3GRD described by Renka (1988). It uses the modification for five-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 Arguments

1: M – INTEGER *Input*

On entry: **must** be the same value supplied for argument M in the preceding call to E01TMF.

Constraint: $\mathrm{M} \geq 23$.

2: X(5, M) – REAL (KIND=nag_wp) array *Input*

On entry: **must** be the same array supplied as argument X in the preceding call to E01TMF. It **must** remain unchanged between calls.

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

On entry: **must** be the same array supplied as argument F in the preceding call to E01TMF. 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 argument IQ in the preceding call to E01TMF. It **must** remain unchanged between calls.

5: RQ($21 \times \mathrm{M} + 11$) – REAL (KIND=nag_wp) array *Input*

On entry: **must** be the same array returned as argument RQ in the preceding call to E01TMF. 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($5, \mathrm{N}$) – REAL (KIND=nag_wp) array *Input*

On entry: XE($1 : 5, i$) 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 E01TNF returns with IFAIL $= 3$.

9: QX($5, \mathrm{N}$) – 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 five partial derivatives $j = 1, 2, 3, 4, 5$. 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 E01TNF returns with IFAIL $= 3$.

10: 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, M $= \langle value \rangle$.
Constraint: M ≥ 23.

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

IFAIL $= 2$

On entry, values in IQ appear to be invalid. Check that IQ has not been corrupted between calls to E01TMF and E01TNF.

On entry, values in RQ appear to be invalid. Check that RQ has not been corrupted between calls to E01TMF and E01TNF.

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 3.9 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

Computational errors should be negligible in most practical situations.

8 Parallelism and Performance

E01TNF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

E01TNF 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

The time taken for a call to E01TNF 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

$$f(\mathbf{x}) = \frac{(1.25 + \cos(5.4x_5))\cos(6x_1)\cos(6x_2)\cos(6x_3)}{6 + 6(3x_4 - 1)^2}$$

at a set of 30 randomly generated data points and calls E01TMF to construct an interpolating function $Q(\mathbf{x})$. It then calls E01TNF 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 to the suggested minimum of 4000 improves the interpolation accuracy at the expense of more time.

See also Section 10 in E01TMF.

10.1 Program Text

```
!   E01TNF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module e01tnfe_mod

!     E01TNF 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
      Function funct(x)
!       This function evaluates the 5D function funct.

!       .. Function Return Value ..
        Real (Kind=nag_wp)             :: funct
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: x(5)
!       .. Intrinsic Procedures ..
        Intrinsic                      :: cos
!       .. Executable Statements ..
        funct = ((1.25_nag_wp+cos(5.4_nag_wp*x(5)))*cos(six*x(1))*cos(six*x(2) &
          )*cos(six*x(3)))/(six+six*(three*x(4)-one)**2)

        Return
      End Function funct
    End Module e01tnfe_mod
    Program e01tnfe

!     E01TNF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e01tmf, e01tnf, g05kff, g05saf, nag_wp
      Use e01tnfe_mod, Only: funct, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: lseed = 1
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: fun
      Integer                          :: genid, i, ifail, liq, lrq, lstate,   &
                                          m, n, nq, nw, subid
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: f(:), q(:), qx(:,:), rq(:), x(:,:),  &
                                          xe(:,:)
      Integer, Allocatable             :: iq(:), state(:)
      Integer                          :: seed(lseed), seed2(lseed)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs
!     .. Executable Statements ..
      Write (nout,*) 'E01TNF 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 initializer 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 = 21*m + 11
      Allocate (x(5,m),f(m),iq(liq),rq(lrq))

!     Generate the data points X
      ifail = 0
      Call g05saf(5*m,state,x,ifail)

!     Evaluate F
      Do i = 1, m
        f(i) = funct(x(1,i))
      End Do

!     Generate the interpolant using E01TMF.
      nq = 0
      nw = 0

      ifail = 0
      Call e01tmf(m,x,f,nw,nq,iq,rq,ifail)

!     Input the number of evaluation points.
      Read (nin,*) n
      Allocate (xe(5,n),q(n),qx(5,n))

!     Generate repeatable evaluation points.
      ifail = 0
      Call g05kff(genid,subid,seed2,lseed,state,lstate,ifail)
      ifail = 0
      Call g05saf(5*n,state,xe,ifail)

!     Evaluate the interpolant.
      ifail = 0
      Call e01tnf(m,x,f,iq,rq,n,xe,q,qx,ifail)

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

99999 Format (1X,I4,1X,3F10.4)
99998 Format (4X,'---|',20('-'),'+',15('-'))
99997 Format (/,4X,'I  |',2X,'F(I)',6X,'Q(I)',4X,'|',1X,'|F(I)-Q(I)|')
    End Program e01tnfe
```

10.2 Program Data

```
E01TNF 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

```
 E01TNF Example Program Results

    I  |  F(I)      Q(I)    | |F(I)-Q(I)|
    ---|--------------------+---------------
    1     0.0058    0.0458    0.0401
    2     0.0034    0.5020    0.4986
    3    -0.1096    0.0743    0.1838
    4     0.0875    0.0337    0.0538
    5     0.0015    0.0375    0.0360
    6    -0.0158   -0.1064    0.0906
    7     0.0046   -0.0490    0.0536
    8    -0.0090   -0.0127    0.0037
```