NAG Library Routine Document

E01ZNF

1 Purpose

E01ZNF evaluates the multi-dimensional interpolating function generated by E01ZMF and its first partial derivatives.

2 Specification

```
SUBROUTINE E01ZNF (D, M, X, F, IQ, RQ, N, XE, Q, QX, IFAIL)

INTEGER            D, M, IQ(2*M+1), N, IFAIL
REAL (KIND=nag_wp) X(D,M), F(M), RQ(*), XE(D,N), Q(N), QX(D,N)
```

3 Description

E01ZNF takes as input the interpolant $Q(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$ of a set of scattered data points $(\mathbf{x}_r, f_r)$, for $r = 1, 2, \ldots, m$, as computed by E01ZMF, and evaluates the interpolant and its first partial derivatives at the set of points $\mathbf{x}_i$, for $i = 1, 2, \ldots, n$.

E01ZNF must only be called after a call to E01ZMF.

E01ZNF 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: D – INTEGER *Input*

On entry: **must** be the same value supplied for parameter D in the preceding call to E01ZMF.

Constraint: $\mathrm{D} \geq 2$.

2: M – INTEGER *Input*

On entry: **must** be the same value supplied for parameter M in the preceding call to E01ZMF.

Constraint: $\mathrm{M} \geq (\mathrm{D}+1) \times (\mathrm{D}+2)/2 + 2$.

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

On entry: **must** be the same array supplied as parameter X in the preceding call to E01ZMF. It **must** remain unchanged between calls.

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

On entry: **must** be the same array supplied as parameter F in the preceding call to E01ZMF. It **must** remain unchanged between calls.

5: 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 E01ZMF. It **must** remain unchanged between calls.

6: RQ($*$) – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array RQ must be at least $((\mathrm{D}+1) \times (\mathrm{D}+2)/2) \times \mathrm{M} + 2 \times \mathrm{D} + 1$.

On entry: **must** be the same array returned as parameter RQ in the preceding call to E01ZMF. It **must** remain unchanged between calls.

7: N – INTEGER *Input*

On entry: n, the number of evaluation points.

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

8: XE(D,N) – REAL (KIND=nag_wp) array *Input*

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

9: 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 E01ZNF returns with IFAIL $= 3$.

10: QX(D,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 partial derivatives $j = 1, 2, \ldots, d$. 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 E01ZNF returns with IFAIL $= 3$.

11: 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: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

IFAIL $= 1$

On entry, $(\mathrm{D}+1) \times (\mathrm{D}+2)/2 \times \mathrm{M} + 2 \times \mathrm{D} + 1$ exceeds the largest machine integer.
D $= \langle value \rangle$ and M $= \langle value \rangle$.

On entry, D $= \langle value \rangle$.
Constraint: $\mathrm{D} \geq 2$.

On entry, M $= \langle value \rangle$ and D $= \langle value \rangle$.
Constraint: $\mathrm{M} \geq (\mathrm{D}+1)(\mathrm{D}+2)/2 + 2$.

On entry, N = ⟨*value*⟩.
Constraint: N ≥ 1.

IFAIL = 2

On entry, values in IQ appear to be invalid. Check that IQ has not been corrupted between calls to E01ZMF and E01ZNF.

On entry, values in RQ appear to be invalid. Check that RQ has not been corrupted between calls to E01ZMF and E01ZNF.

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() = ⟨*value*⟩.

IFAIL = −999

Dynamic memory allocation failed.

7 Accuracy

Computational errors should be negligible in most practical situations.

8 Further Comments

The time taken for a call to E01ZNF 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.

9 Example

This program evaluates the function (in six variables)

$$f(x) = \frac{x_1 x_2 x_3}{1 + 2x_4 x_5 x_6}$$

at a set of randomly generated data points and calls E01ZMF to construct an interpolating function Q_x. It then calls E01ZNF to evaluate the interpolant at a set of points on the line $x_i = x$, for $i = 1, 2, \ldots, 6$. To reduce the time taken by this example, the number of data points is limited. Increasing this value to the suggested minimum of 4000 improves the interpolation accuracy at the expense of more time.

See also Section 9 in E01ZMF.

9.1 Program Text

```
!   E01ZNF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module e01znfe_mod

!     E01ZNF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter        :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter        :: two = 2.0_nag_wp
      Integer, Parameter                   :: d = 6, nin = 5, nout = 6
    Contains
      Function funct(x)
```

```
!       This function evaluates the 6D function funct.

!       .. Function Return Value ..
        Real (Kind=nag_wp)                   :: funct
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: x(d)
!       .. Executable Statements ..
        funct = x(1)*x(2)*x(3)/(one+two*x(4)*x(5)*x(6))

        Return
      End Function funct
    End Module e01znfe_mod
    Program e01znfe

!     E01ZNF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e01zmf, e01znf, g05kff, g05saf, nag_wp
      Use e01znfe_mod, Only: d, 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)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: abs, real
!     .. Executable Statements ..
      Write (nout,*) 'E01ZNF Example Program Results'

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

!     Read in the base generator information and seeds
      Read (nin,*) genid, subid, seed(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 = (d+1)*(d+2)/2*m + 2*d + 1
      Allocate (x(d,m),f(m),iq(liq),rq(lrq))

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

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

```
!     Generate the interpolant using E01ZMF.
      nq = 0
      nw = 0

      ifail = 0
      Call e01zmf(d,m,x,f,nw,nq,iq,rq,ifail)

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

!     Generate a set of evaluation points lying on diagonal line
!     xe(1:d,i) = xe(1,i) = i/(n+1).
      Do i = 1, n
        xe(1:d,i) = real(i,kind=nag_wp)/real(n+1,kind=nag_wp)
      End Do

!     Evaluate the interpolant.
      ifail = 0
      Call e01znf(d,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 e01znfe
```

9.2 Program Data

```
E01ZNF Example Program Data
1  1  1762543     : genid, subid, seed(1)
120               : M, the number of data points
9                 : N, the number of evaluation points
```

9.3 Program Results

```
 E01ZNF Example Program Results

    I  |  F(I)      Q(I)     | |F(I)-Q(I)|
    ---|--------------------+---------------
    1     0.0010    0.0025    0.0015
    2     0.0079    0.0035    0.0043
    3     0.0256    0.0213    0.0043
    4     0.0567    0.0541    0.0026
    5     0.1000    0.0991    0.0009
    6     0.1508    0.1528    0.0019
    7     0.2034    0.2071    0.0037
    8     0.2530    0.2558    0.0028
    9     0.2966    0.2941    0.0024
```