NAG Library Function Document

nag_nd_shep_eval (e01znc)

1 Purpose

nag_nd_shep_eval (e01znc) evaluates the multi-dimensional interpolating function generated by nag_nd_shep_interp (e01zmc) and its first partial derivatives.

2 Specification

```
#include <nag.h>
#include <nage01.h>
void nag_nd_shep_eval (Integer d, Integer m, const double x[],
     const double f[], const Integer iq[], const double rq[], Integer n,
     const double xe[], double q[], double qx[], NagError *fail)
```

3 Description

nag_nd_shep_eval (e01znc) 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 nag_nd_shep_interp (e01zmc), and evaluates the interpolant and its first partial derivatives at the set of points $\mathbf{x}_i$, for $i = 1, 2, \ldots, n$.

nag_nd_shep_eval (e01znc) must only be called after a call to nag_nd_shep_interp (e01zmc).

nag_nd_shep_eval (e01znc) 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 Arguments

1: **d** – Integer *Input*

On entry: **must** be the same value supplied for argument **d** in the preceding call to nag_nd_shep_interp (e01zmc).

Constraint: $\mathbf{d} \geq 2$.

2: **m** – Integer *Input*

On entry: **must** be the same value supplied for argument **m** in the preceding call to nag_nd_shep_interp (e01zmc).

Constraint: $\mathbf{m} \geq (\mathbf{d}+1) \times (\mathbf{d}+2)/2 + 2$.

3: **x**[**d** × **m**] – const double *Input*

Note: the ith ordinate of the point x_j is stored in $\mathbf{x}[(j-1) \times \mathbf{d} + i - 1]$.

On entry: **must** be the same array supplied as argument **x** in the preceding call to nag_nd_shep_interp (e01zmc). It **must** remain unchanged between calls.

4: **f**[**m**] – const double *Input*

On entry: **must** be the same array supplied as argument **f** in the preceding call to nag_nd_shep_interp (e01zmc). It **must** remain unchanged between calls.

5: **iq**[$\mathbf{2} \times \mathbf{m} + \mathbf{1}$] – const Integer *Input*

On entry: **must** be the same array returned as argument **iq** in the preceding call to nag_nd_shep_interp (e01zmc). It **must** remain unchanged between calls.

6: **rq**[dim] – const double *Input*

Note: the dimension, *dim*, of the array **rq** must be at least $((\mathbf{d}+1) \times (\mathbf{d}+2)/2) \times \mathbf{m} + 2 \times \mathbf{d} + 1$.

On entry: **must** be the same array returned as argument **rq** in the preceding call to nag_nd_shep_interp (e01zmc). It **must** remain unchanged between calls.

7: **n** – Integer *Input*

On entry: n, the number of evaluation points.

Constraint: $\mathbf{n} \geq 1$.

8: **xe**[$\mathbf{d} \times \mathbf{n}$] – const double *Input*

Note: the ith ordinate of the point x_j is stored in **xe**[$(j-1) \times \mathbf{d} + i - 1$].

On entry: **xe**[$(j-1) \times \mathbf{d}$], ..., **xe**[$(j-1) \times \mathbf{d} + \mathbf{d} - 1$] must be set to the evaluation point $\mathbf{x}_j$, for $j = 1, 2, \ldots, n$.

9: **q**[**n**] – double *Output*

On exit: **q**[$i-1$] 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 nag_real_largest_number (X02ALC)), and nag_nd_shep_eval (e01znc) returns with **fail.code** = NE_BAD_POINT.

10: **qx**[$\mathbf{d} \times \mathbf{n}$] – double *Output*

Note: the (i, j)th element of the matrix is stored in **qx**[$(j-1) \times \mathbf{d} + i - 1$].

On exit: **qx**[$(j-1) \times \mathbf{d} + i - 1$] contains the value of the partial derivatives with respect to the ith independent variable (dimension) of the interpolant $Q(\mathbf{x})$ at $\mathbf{x}_j$, for $j = 1, 2, \ldots, n$, and for each of the partial derivatives $i = 1, 2, \ldots, d$. 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 nag_real_largest_number (X02ALC)), and nag_nd_shep_eval (e01znc) returns with **fail.code** = NE_BAD_POINT.

11: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_BAD_POINT

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 nag_real_largest_number: nag_real_largest_number = ⟨*value*⟩.

NE_INT

On entry, **d** = ⟨*value*⟩.
Constraint: **d** ≥ 2.

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

NE_INT_2

On entry, $((\mathbf{d}+1)\times(\mathbf{d}+2)/2)\times\mathbf{m}+2\times\mathbf{d}+1$ exceeds the largest machine integer.
d = ⟨*value*⟩ and **m** = ⟨*value*⟩.

On entry, **m** = ⟨*value*⟩ and **d** = ⟨*value*⟩.
Constraint: $\mathbf{m}\geq(\mathbf{d}+1)\times(\mathbf{d}+2)/2+2$.

NE_INT_ARRAY

On entry, values in **iq** appear to be invalid. Check that **iq** has not been corrupted between calls to nag_nd_shep_interp (e01zmc) and nag_nd_shep_eval (e01znc).

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

NE_REAL_ARRAY

On entry, values in **rq** appear to be invalid. Check that **rq** has not been corrupted between calls to nag_nd_shep_interp (e01zmc) and nag_nd_shep_eval (e01znc).

7 Accuracy

Computational errors should be negligible in most practical situations.

8 Parallelism and Performance

nag_nd_shep_eval (e01znc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_nd_shep_eval (e01znc) 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 Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken for a call to nag_nd_shep_eval (e01znc) 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 (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 nag_nd_shep_interp (e01zmc) to construct an interpolating function Q_x. It then calls nag_nd_shep_eval (e01znc) 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 10 in nag_nd_shep_interp (e01zmc).

10.1 Program Text

```
/* nag_nd_shep_eval (e01znc) Example Program.
 *
 * Copyright 2013 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage01.h>
#include <nagg05.h>

static double funct(double *x);

int main(void)
{
  /* Scalars */
  Integer     d = 6, exit_status = 0, lseed = 1, subid = 0;
  Integer     i, j, liq, lrq, lstate, m, n, nq, nw, tmpdsm;
  double      fun;
  /* Arrays */
  double      *f = 0, *q = 0, *qx = 0, *rq = 0, *x = 0, *xe = 0;
  Integer     *iq = 0, *state = 0;
  Integer     seed[] = { 1762543 };
  /* Nag Types */
  Nag_BaseRNG genid = Nag_Basic;
  NagError    fail;

  INIT_FAIL(fail);

  printf("nag_nd_shep_eval (e01znc) Example Program Results\n\n");
  /* Skip heading in data file*/
  scanf("%*[^\n] ");
  /* Set up state array for generating a random sample of data locations
   * using nag_rand_init_repeatable (g05kfc).
   *
   * First get the length of the state array by setting lstate = -1.
   */
  lstate = -1;
  nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
  if (fail.code == NE_NOERROR)
    {
      /* Allocate arrays */
      if (!(state = NAG_ALLOC(lstate, Integer)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }

      /* Then initialise the generator to a repeatable sequence */
      nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate,
```

```
                                &fail);
    }
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Input the number of nodes.*/
  scanf("%ld%*[^\n] ", &m);
  liq = 2 * m + 1;
  lrq = (d + 1) * (d + 2)/2 * m + 2 * d + 1;
  if (
    !(x = NAG_ALLOC(d*m, double)) ||
    !(f = NAG_ALLOC(m, double)) ||
    !(iq = NAG_ALLOC(liq, Integer)) ||
    !(rq = NAG_ALLOC(lrq, double))
    )
    {
      printf("Allocation failure\n");
      exit_status = -2;
      goto END;
    }
  /* Generate d*m  pseudorandom numbers in U(0,1) using
   * nag_rand_basic (g05sac).
   */
  tmpdsm = d*m;
  nag_rand_basic(tmpdsm, state, x, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_rand_basic (g05sac).\n%s\n", fail.message);
      exit_status = 2;
      goto END;
    }

  /* Evaluate f at x */
  for (i = 0; i < m; i++)
    f[i] = funct(&x[i*d]);

  /* Generate the interpolant using nag_nd_shep_interp (e01zmc):
     Interpolating functions, modified Shepard's method, d variables.
   */
  nq = 0;
  nw = 0;
  nag_nd_shep_interp(d, m, x, f, nw, nq, iq, rq, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_nd_shep_interp (e01zmc).\n%s\n", fail.message);
      exit_status = 3;
      goto END;
    }

  /* Input the number of evaluation points and allocate arrays with lengths
     based on this.
   */
  scanf("%ld%*[^\n]", &n);
  if (
    !(xe = NAG_ALLOC(d*n, double)) ||
    !(q = NAG_ALLOC(n, double)) ||
    !(qx = NAG_ALLOC(d*n, double))
    )
    {
      printf("Allocation failure\n");
      exit_status = -3;
      goto END;
    }

  /* Generate a set of evaluation points lying on diagonal line
   * xe(1:d,i) = xe(1,i) = i/(n+1).
```

```
   */
  for (i = 0; i < n; i++)
    for (j = 0; j < d; j++)
      xe[i*d+j] = (double) (i+1)/(double) (n + 1);

  /* Evaluate the interpolant using nag_nd_shep_eval (e01znc), at given
     interpolated values, where interpolant previously computed by e01zmc.
   */
  nag_nd_shep_eval(d, m, x, f, iq, rq, n, xe, q, qx, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_nd_shep_eval (e01znc).\n%s\n", fail.message);
      exit_status = 4;
      goto END;
    }

  /* Print interpolated function values against actual function values
   * at the points on the diagonal line. */
  /* Header */
  printf("    i  |  f(i)      q(i)    | |f(i)-q(i)|\n");
  printf("    ---|--------------------+---------------\n");
  /* Results */
  for (i = 0; i < n; i++)
    {
      fun = funct(&xe[i*d]);
      printf("%5ld %10.4f%10.4f%10.4f\n", i, fun, q[i], fabs(fun-q[i]));
    }

 END:

  NAG_FREE(f);
  NAG_FREE(q);
  NAG_FREE(qx);
  NAG_FREE(rq);
  NAG_FREE(x);
  NAG_FREE(xe);
  NAG_FREE(iq);
  NAG_FREE(state);

  return exit_status;
}
static double funct(double *x)
{
  double funct_return;
  funct_return = x[0]*x[1]*x[2]/(1.0 + 2.0*x[3]*x[4]*x[5]);
  return funct_return;
}
```

10.2 Program Data

```
nag_nd_shep_eval (e01znc) Example Program Data
120               : m, the number of data points
9                 : n, the number of evaluation points
```

10.3 Program Results

```
nag_nd_shep_eval (e01znc) Example Program Results

   i  |  f(i)      q(i)    | |f(i)-q(i)|
   ---|--------------------+---------------
   0     0.0010    0.0025    0.0015
   1     0.0079    0.0035    0.0043
   2     0.0256    0.0213    0.0043
   3     0.0567    0.0541    0.0026
   4     0.1000    0.0991    0.0009
   5     0.1508    0.1528    0.0019
   6     0.2034    0.2071    0.0037
   7     0.2530    0.2558    0.0028
   8     0.2966    0.2941    0.0024
```