NAG Library Function Document
nag_4d_shep_eval (e01tlc)

1 Purpose

nag_4d_shep_eval (e01tlc) evaluates the four-dimensional interpolating function generated by
nag_4d_shep_interp (e01tkc) and its first partial derivatives.

2 Specification

```c
#include <nag.h>
#include <nage01.h>

void nag_4d_shep_eval (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_4d_shep_eval (e01tlc) takes as input the interpolant $Q(x)$, $x \in \mathbb{R}^4$ of a set of scattered data points
$(x_r, f_r)$, for $r = 1, 2, \ldots, m$, as computed by nag_4d_shep_interp (e01tkc), and evaluates the interpolant
and its first partial derivatives at the set of points $x_i$, for $i = 1, 2, \ldots, n$.

nag_4d_shep_eval (e01tlc) must only be called after a call to nag_4d_shep_interp (e01tkc).

nag_4d_shep_eval (e01tlc) 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


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
   nag_4d_shep_interp (e01tkc).

   Constraint: **m** ≥ 16.

2:  
   **x[4 × m]** – const double

   *Input*

   Note: the coordinates of $x_r$ are stored in $x[(r - 1) \times 4] \ldots x[(r - 1) \times 4 + 3]$.

   On entry: **must** be the same array supplied as argument **x** in the preceding call to
   nag_4d_shep_interp (e01tkc). It **must** remain unchanged between calls.

3:  
   **f[m]** – const double

   *Input*

   On entry: **must** be the same array supplied as argument **f** in the preceding call to
   nag_4d_shep_interp (e01tkc). It **must** remain unchanged between calls.
Input

On entry: $$\text{iq}[2 \times m + 1]$$ – const Integer

On entry: must be the same array returned as argument iq in the preceding call to nag_4d_shep_interp (e01tkc). It must remain unchanged between calls.

Input

On entry: $$\text{rq}[15 \times m + 9]$$ – const double

On entry: must be the same array returned as argument rq in the preceding call to nag_4d_shep_interp (e01tkc). It must remain unchanged between calls.

Input

On entry: $$n$$, the number of evaluation points.

Constraint: $$n \geq 1$$.

Input

On entry: $$\text{xe}[4 \times n]$$ – const double

Note: the $$(i, j)$$th element of the matrix is stored in $$\text{xe}[(j - 1) \times 4 + i - 1]$$.

On entry: $$\text{xe}[(r - 1) \times 4], \ldots, \text{xe}[(r - 1) \times 4 + 3]$$ must be set to the evaluation point $$x_i$$, for $$i = 1, 2, \ldots, n$$.

Output

On exit: $$q[i - 1]$$ contains the value of the interpolant, at $$x_i$$, 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_4d_shep_eval (e01tlc) returns with fail.code = NE_BAD_POINT.

Output

On exit: $$\text{qx}[4 \times n]$$ – double

Note: the $$(i, j)$$th element of the matrix is stored in $$\text{qx}[(j - 1) \times 4 + i - 1]$$.

On exit: $$\text{qx}[(i - 1) \times 4 + j - 1]$$ contains the value of the partial derivatives with respect to $$x_j$$ of the interpolant $$Q(x)$$ at $$x_i$$, 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 nag_real_largest_number (X02ALC)), and nag_4d_shep_eval (e01tlc) returns with fail.code = NE_BAD_POINT.

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.

See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument $$(\text{value})$$ 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 = $$(\text{value})$$. 
NE_INT

On entry, \( m = \langle \text{value} \rangle \).
Constraint: \( m \geq 16 \).

On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 1 \).

NE_INT_ARRAY

On entry, values in \( \mathbf{iq} \) appear to be invalid. Check that \( \mathbf{iq} \) has not been corrupted between calls to nag_4d_shep_interp (e01tkc) and nag_4d_shep_eval (e01tlc).

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.

An unexpected error has been triggered by this function. Please contact NAG. See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction for further information.

NE_REAL_ARRAY

On entry, values in \( \mathbf{rq} \) appear to be invalid. Check that \( \mathbf{rq} \) has not been corrupted between calls to nag_4d_shep_interp (e01tkc) and nag_4d_shep_eval (e01tlc).

7 Accuracy

Computational errors should be negligible in most practical situations.

8 Parallelism and Performance

nag_4d_shep_eval (e01tlc) 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 function. 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 nag_4d_shep_eval (e01tlc) 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 nag_4d_shep_interp (e01tkc) to construct an interpolating function \( Q(x) \). It then calls nag_4d_shep_eval (e01tlc) 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 nag_4d_shep_interp (e01tkc).

10.1 Program Text

/* nag_4d_shep_eval (e01tlc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 23, 2010. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage01.h>
#include <nagg05.h>
#include <math.h>

#ifdef __cplusplus
extern "C" {
#endif
static double NAG_CALL funct(double x[]);
#ifdef __cplusplus
}
#endif
#define X(I, J) x[I *4 + J]
#define XE(I, J) xe[I *4 + J]

int main(void)
{
/* Scalars */
Integer exit_status, i, m, n, nq, nw, liq, lrq, lstate, subid;
Integer lseed = 1;
double fun;
Nag_BaseRNG genid;
NagError fail;
/* Arrays */
double *f = 0, *q = 0, *qx = 0, *rq = 0, *xe = 0, *x = 0;
Integer *iq = 0, *state = 0;
Integer seed[1], seed2[1];
exit_status = 0;
INIT_FAIL(fail);
printf("nag_4d_shep_eval (e01tlc) Example Program Results\n");
/* Skip heading in data file */
#ifdef _WIN32
scanf_s("%*[\n ] ");
#else
scanf("%*[\n ] ");
#endif
/* Input the seeds. */
#ifdef _WIN32
scanf_s("%NAG_IFMT%NAG_IFMT%*[\n ]", &seed[0], &seed2[0]);
#else
scanf("%NAG_IFMT%NAG_IFMT%*[\n ]", &seed[0], &seed2[0]);
#endif
/* Choose the base generator */
genid = Nag_Basic;
subid = 0;
/* Get the length of the state array */
lstate = -1;
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. */
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%[\n ] ", &m);
#else
  scanf("%"NAG_IFMT"%[\n ] ", &m);
#endif

/* Allocate memory */
lrq = 21 * m + 11;
liq = 2 * m + 1;
if ( !(f = NAG_ALLOC(m, double)) ||
     !(x = NAG_ALLOC(m*4, double)) ||
     !(rq = NAG_ALLOC(lrq, double)) ||
     !(iq = NAG_ALLOC(liq, Integer)) ||
     !(state = NAG_ALLOC(lstate, Integer)))
{
  printf("Allocation failure\n");
  exit_status = -1;
  goto END;
}

/* 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;
}

/* Generate the data points X */
nag_rand_basic(m*4, state, x, &fail);
if (fail.code != NE_NOERROR) {
  printf("Error from nag_rand_basic (g05sac).\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}

/* Evaluate F */
for (i = 0; i < m; ++i) {
  f[i] = funct(&X(i,0));
}

/* Generate the interpolant. */
nq = 0;
qw = 0;

/* nag_4d_shep_interp (e01tkc).
 * Interpolating functions, modified Shepard’s method, four
 * variables
 */
nag_4d_shep_interp(m, x, f, nw, nq, iq, rq, &fail);
if (fail.code != NE_NOERROR) {
  printf("Error from nag_4d_shep_interp (e01tkc).\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}

/* Input the number of evaluation points. */
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%[\n ] ", &n);
#else
  scanf("%"NAG_IFMT"%[\n ] ", &n);
#endif

/* Allocate memory for nag_4d_shep_eval (e01tlc) */
if (!(q = NAG_ALLOC(n, double)) ||
!(qx = NAG_ALLOC(n*4, double)) ||
!(xe = NAG_ALLOC(n*4, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Generate repeatable evaluation points. */
NAG_CALL nag_rand_init_repeatable(genid, subid, seed2, lseed, state, &lstate, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_init_repeatable (g05kfc).\n", fail.message);
    exit_status = 1;
    goto END;
}

NAG_CALL nag_rand_basic(n*4, state, xe, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_basic (g05sac).\n", fail.message);
    exit_status = 1;
    goto END;
}

NAG_CALL nag_4d_shep_eval(m, x, f, iq, rq, n, xe, q, qx, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_4d_shep_eval (e01tlc).\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("\n i f(x) q(x) |f(x)-q(x)|\n");
for (i = 0; i < n; ++i) {
    fun = funct(&XE(i,0));
    printf("%6NAG_IFMT%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(xe);
NAG_FREE(x);
NAG_FREE(iq);
NAG_FREE(state);

return exit_status;
}

static double NAG_CALL funct(double x[])
{
    double ret_val;
    ret_val = ((1.25+cos(5.4*x[3]))*cos(6.0*x[0])*cos(6.0*x[1]))/
(6.0*(1.0+pow((3.0*x[2]-1.0),2.0)));
    return ret_val;
}

10.2 Program Data

nag_4d_shep_eval (e01tlc) Example Program Data
1762543 43331 : random seeds
30 : m the number of data points
8 : n the number of evaluation points
## 10.3 Program Results

nag_4d_shep_eval (e01tlc) Example Program Results

| i | f(x)  | q(x)  | |f(x)-q(x)| |
|---|-------|-------|----------------|
| 0 | -0.0189 | -0.0394 | 0.0205 |
| 1 | -0.0186 | 0.0967  | 0.1153 |
| 2 | 0.1147   | 0.0606  | 0.0541 |
| 3 | 0.0096   | -0.1313 | 0.1409 |
| 4 | -0.1354 | -0.1878 | 0.0524 |
| 5 | 0.0022   | -0.1595 | 0.1617 |
| 6 | -0.0095 | -0.1179 | 0.1084 |
| 7 | 0.0113   | -0.3950 | 0.4063 |