NAG Library Function Document
nag_5d_shep_eval (e01tnc)

1 Purpose

nag_5d_shep_eval (e01tnc) evaluates the five-dimensional interpolating function generated by
nag_5d_shep_interp (e01tmc) and its first partial derivatives.

2 Specification

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

void nag_5d_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_5d_shep_eval (e01tnc) takes as input the interpolant \( Q(x) \), \( x \in \mathbb{R}^5 \) of a set of scattered data points
\((x_r, f_r)\), for \( r = 1,2,\ldots,m \), as computed by nag_5d_shep_interp (e01tmc), and evaluates the interpolant
and its first partial derivatives at the set of points \( x_i \), for \( i = 1,2,\ldots,n \).

nag_5d_shep_eval (e01tnc) must only be called after a call to nag_5d_shep_interp (e01tmc).

nag_5d_shep_eval (e01tnc) 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


scattered data ACM Trans. Math. Software 14 151–152

5 Arguments

1: \( m \) – Integer

On entry: \( m \) must be the same value supplied for argument \( m \) in the preceding call to
nag_5d_shep_interp (e01tmc).

Constraint: \( m \geq 23 \).

2: \( x[5 \times m] \) – const double

On entry: \( x \) must be the same array supplied as argument \( x \) in the preceding call to
nag_5d_shep_interp (e01tmc). It must remain unchanged between calls.

Note: the \((i,j)\)th element of the matrix \( X \) is stored in \( x[(j-1) \times 5 + i - 1] \).

3: \( f[m] \) – const double

On entry: \( f \) must be the same array supplied as argument \( f \) in the preceding call to
nag_5d_shep_interp (e01tmc). It must remain unchanged between calls.
4: \[ \text{iq}[2 \times m + 1] \] – const Integer \hspace{1cm} \text{Input}

\text{On entry: must be the same array returned as argument iq in the preceding call to nag_5d_shep_interp (e01tmc). It must remain unchanged between calls.}

5: \[ \text{rq}[21 \times m + 11] \] – const double \hspace{1cm} \text{Input}

\text{On entry: must be the same array returned as argument rq in the preceding call to nag_5d_shep_interp (e01tmc). It must remain unchanged between calls.}

6: \[ n \] – Integer \hspace{1cm} \text{Input}

\text{On entry: } n, \text{ the number of evaluation points.}

\text{Constraint: } n \geq 1.

7: \[ \text{xe}[5 \times n] \] – const double \hspace{1cm} \text{Input}

\text{Note: the } (i,j)\text{th element of the matrix is stored in xe[}(j-1) \times 5 + i - 1].

\text{On entry: xe[} (i-1) \times 5], \ldots, xe[(i-1) \times 5 + 4] \text{ must be set to the evaluation point } x_i, \text{ for } i = 1, 2, \ldots, n.

8: \[ q[n] \] – double \hspace{1cm} \text{Output}

\text{On exit: } q[i-1] \text{ contains the value of the interpolant, at } x_i, \text{ for } i = 1, 2, \ldots, n. \text{ If any of these evaluation points lie outside the region of definition of the interpolant the corresponding entries in } q \text{ are set to the largest machine representable number (see nag_real_largest_number (X02ALC)), and nag_5d_shep_eval (e01tnc) returns with fail.code = NE_BAD_POINT.}

9: \[ qx[5 \times n] \] – double \hspace{1cm} \text{Output}

\text{Note: the } (i,j)\text{th element of the matrix is stored in qx[}(j-1) \times 5 + i - 1].

\text{On exit: qx[} (i-1) \times 5 + j - 1] \text{ contains the value of the partial derivatives with respect to } x_j \text{ of the interpolant } Q(x) \text{ at } x_i, \text{ for } i = 1, 2, \ldots, n, \text{ and for each of the five partial derivatives } j = 1, 2, 3, 4, 5. \text{ 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_5d_shep_eval (e01tnc) returns with fail.code = NE_BAD_POINT.}

10: \[ \text{fail} \] – NagError * \hspace{1cm} \text{Input/Output}

\text{The NAG error argument (see Section 3.6 in the Essential Introduction).}

6 \text{ \hspace{1cm} Error Indicators and Warnings}

\text{NE_ALLOC_FAIL}

\text{Dynamic memory allocation failed.}

See Section 3.2.1.2 in the Essential Introduction for further information.

\text{NE_BAD_PARAM}

\text{On entry, argument } \langle \text{value} \rangle \text{ had an illegal value.}

\text{NE_BAD_POINT}

\text{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 \text{ and } qx \text{ have been set to nag_real_largest_number: nag_real_largest_number = } \langle \text{value} \rangle.
NE_INT
On entry, \( m = \langle \text{value} \rangle \).
Constraint: \( m \geq 23 \).

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

NE_INT_ARRAY
On entry, values in \( iq \) appear to be invalid. Check that \( iq \) has not been corrupted between calls to
\nag_5d_shep_interp (e01tmc)\nand \nag_5d_shep_eval (e01tnc)\n.

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 \( rq \) appear to be invalid. Check that \( rq \) has not been corrupted between calls to
\nag_5d_shep_interp (e01tmc)\nand \nag_5d_shep_eval (e01tnc)\n.

7 Accuracy
Computational errors should be negligible in most practical situations.

8 Parallelism and Performance
\nag_5d_shep_eval (e01tnc)\n\n\is threaded by NAG for parallel execution in multithreaded implementations
\nof the NAG Library.

\nag_5d_shep_eval (e01tnc)\n\makes calls to BLAS and/or LAPACK routines, which may be threaded
\nwithin the vendor library used by this implementation. Consult the documentation for the vendor library
\nfor further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the
\nOpenMP environment used within this function. Please also consult the Users’ Note for your
\nimplementation for any additional implementation-specific information.

9 Further Comments
The time taken for a call to \nag_5d_shep_eval (e01tnc)\nwill 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 \( O(nm) \) time will be required.

10 Example
This program evaluates the function
\[ f(x) = \frac{(1.25 + \cos(5.4x_5)) \cos(6x_1) \cos(6x_2) \cos(6x_3)}{6 + 6(3x_4 - 1)^2} \]
at a set of 30 randomly generated data points and calls \nag_5d_shep_interp (e01tmc)\nto construct an
interpolating function \( Q(x) \). It then calls nag_5d_shep_eval (e01tnc) 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 nag_5d_shep_interp (e01tmc).

10.1 Program Text

/* nag_5d_shep_eval (e01tnc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 23, 2010. */

#include <stdio.h>
#include <nag.h>
#include <nagstdlib.h>
#include <nag01.h>
#include <nag05.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 *5 + J]
#define XE(I, J) xe[I *5 + 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_5d_shep_eval (e01tnc) Example Program Results\n");

#ifdef WIN32
    scanf_s("%*[\n ] ");
#else
    scanf("%*[\n ] ");
#endif

#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

    /* Input the seeds. */
    /* 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\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Input the number of nodes. */
#undef _WIN32
#include "nag.h"

/* Allocate memory */
lrq = 21 * m + 11;
liq = 2 * m + 1;
if (!((f = NAG_ALLOC(m, double)) ||
     (x = NAG_ALLOC(m*5, 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\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Generate the data points X */
nag_rand_basic(m*5, state, x, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_basic (g05sac).\n\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. */

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

/* Allocate memory for nag_5d_shep_eval (e01tnc) */
if (!(q = NAG_ALLOC(n, double)) ||
   !(qx = NAG_ALLOC(n*5, double)) ||
   !(xe = NAG_ALLOC(n*5, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Generate repeatable evaluation points. */
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%s\n", fail.message);
    exit_status = 1;
    goto END;
}
nag_rand_basic(n*5, state, xe, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_basic (g05sac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_5d_shep_eval (e01tnc). */
/* Evaluate interpolant and first derivatives computed by
   nag_5d_shep_interp (e01tmc). */
fail.print = Nag_TRUE;
nag_5d_shep_eval(m, x, f, iq, rq, n, xe, q, qx, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_5d_shep_eval (e01tnc).\n%s\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("%6"NAG_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(iq);
NAG_FREE(state);

return exit_status;
}

static double NAG_CALL funct(double x[])
{
    /* Scalars */
    double ret_val;
    ret_val = ((1.25+cos(5.4*x[4]))*cos(6.0*x[0])*cos(6.0*x[1])
               *cos(6.0*x[2]))/(6.0+6.0*pow((3.0*x[3]-1.0),2.0));
    return ret_val;
}
```
10.2 Program Data

nag_5d_shep_eval (e01tnc) 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_5d_shep_eval (e01tnc) Example Program Results

| i | f(x) | q(x) | |f(x)-Q(x)| |
|---|------|------|------------------|
| 0 | 0.0058 | 0.0464 | 0.0407          |
| 1 | 0.0034 | 0.4855 | 0.4821          |
| 2 | -0.1096 | 0.0724 | 0.1820          |
| 3 | 0.0875 | 0.0320 | 0.0555          |
| 4 | 0.0015 | 0.0373 | 0.0358          |
| 5 | -0.0158 | -0.1170 | 0.1012         |
| 6 | 0.0046 | -0.0484 | 0.0530         |
| 7 | -0.0090 | -0.0134 | 0.0043         |