NAG Toolbox

nag_interp_4d_scat_shep_eval (e01tl)

1 Purpose

nag_interp_4d_scat_shep_eval (e01tl) evaluates the four-dimensional interpolating function generated by nag_interp_4d_scat_shep (e01tk) and its first partial derivatives.

2 Syntax

```
[q, qx, ifail] = nag_interp_4d_scat_shep_eval(x, f, iq, rq, xe, 'm', m, 'n', n)
[q, qx, ifail] = e01tl(x, f, iq, rq, xe, 'm', m, 'n', n)
```

3 Description

nag_interp_4d_scat_shep_eval (e01tl) takes as input the interpolant $Q(\mathbf{x})$, $x \in \mathbb{R}^4$ of a set of scattered data points $(\mathbf{x}_r, f_r)$, for $r = 1, 2, \ldots, m$, as computed by nag_interp_4d_scat_shep (e01tk), and evaluates the interpolant and its first partial derivatives at the set of points $\mathbf{x}_i$, for $i = 1, 2, \ldots, n$.

nag_interp_4d_scat_shep_eval (e01tl) must only be called after a call to nag_interp_4d_scat_shep (e01tk).

nag_interp_4d_scat_shep_eval (e01tl) 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

5.1 Compulsory Input Parameters

1: $\mathbf{x}(\mathbf{4}, \mathbf{m})$ – REAL (KIND=nag_wp) array

Note: the coordinates of x_r are stored in $\mathbf{x}(1, r) \ldots \mathbf{x}(4, r)$.

must be the same array supplied as argument **x** in the preceding call to nag_interp_4d_scat_shep (e01tk). It **must** remain unchanged between calls.

2: $\mathbf{f}(\mathbf{m})$ – REAL (KIND=nag_wp) array

must be the same array supplied as argument **f** in the preceding call to nag_interp_4d_scat_shep (e01tk). It **must** remain unchanged between calls.

3: $\mathbf{iq}(\mathbf{2} \times \mathbf{m} + \mathbf{1})$ – INTEGER array

must be the same array returned as argument **iq** in the preceding call to nag_interp_4d_scat_shep (e01tk). It **must** remain unchanged between calls.

4: **rq**($\mathbf{15} \times \mathbf{m} + \mathbf{9}$) – REAL (KIND=nag_wp) array

must be the same array returned as argument **rq** in the preceding call to nag_interp_4d_scat_shep (e01tk). It **must** remain unchanged between calls.

5: **xe**($\mathbf{4}, \mathbf{n}$) – REAL (KIND=nag_wp) array

xe$(1:4, i)$ must be set to the evaluation point $\mathbf{x}_i$, for $i = 1, 2, \dots, n$.

5.2 Optional Input Parameters

1: **m** – INTEGER

Default: the dimension of the array **f** and the first dimension of the array **x**. (An error is raised if these dimensions are not equal.)

must be the same value supplied for argument **m** in the preceding call to nag_interp_4d_scat_shep (e01tk).

Constraint: $\mathbf{m} \geq 16$.

2: **n** – INTEGER

Default: the second dimension of the array **xe**.

n, the number of evaluation points.

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

5.3 Output Parameters

1: **q**($\mathbf{n}$) – REAL (KIND=nag_wp) array

q(i) contains the value of the interpolant, at $\mathbf{x}_i$, for $i = 1, 2, \dots, n$. 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_machine_real_largest (x02al)), and nag_interp_4d_scat_shep_eval (e01tl) returns with **ifail** $= 3$.

2: **qx**($\mathbf{4}, \mathbf{n}$) – REAL (KIND=nag_wp) array

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 $i = 1, 2, \dots, n$, 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_machine_real_largest (x02al)), and nag_interp_4d_scat_shep_eval (e01tl) returns with **ifail** $= 3$.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

Constraint: $\mathbf{m} \geq 16$.

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

ifail $= 2$

On entry, values in **iq** appear to be invalid. Check that **iq** has not been corrupted between calls to nag_interp_4d_scat_shep (e01tk) and nag_interp_4d_scat_shep_eval (e01tl).

On entry, values in **rq** appear to be invalid. Check that **rq** has not been corrupted between calls to nag_interp_4d_scat_shep (e01tk) and nag_interp_4d_scat_shep_eval (e01tl).

ifail $= 3$

On entry, at least one evaluation point lies outside the region of definition of the interpolant.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

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 nag_interp_4d_scat_shep_eval (e01tl) 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

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

at a set of 30 randomly generated data points and calls nag_interp_4d_scat_shep (e01tk) to construct an interpolating function $Q(\mathbf{x})$. It then calls nag_interp_4d_scat_shep_eval (e01tl) 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_interp_4d_scat_shep (e01tk).

9.1 Program Text

```
     function e01tl_example

fprintf('e01tl example results\n\n');

genid = nag_int(1);
subid = nag_int(1);
seed  = [nag_int(1762543)];
seed2 = [nag_int(43331)];

% Initialize the generator to a repeatable sequence
[state, ifail] = g05kf(genid, subid, seed);

% Generate the data points x
[state, x, ifail] = g05sa(nag_int(120), state);

% Put x into the right shape for the e01 routines
x = reshape(x, 4, 30);
```

```
% Create function values
c1 = cos(6*x(1,:));
c2 = cos(6*x(2,:));
c3 = 6 + 6*(3*x(3,:)-1).^2;
c4 = 1.25 + cos(5.4*x(4,:));
f  = c4.*c1.*c2./c3;

% Generate the interpolant
nq = nag_int(0);
nw = nag_int(0);
[iq, rq, ifail] = e01tk(x, f, nw, nq);

% Generate repeatable evaluation points
[state, ifail] = g05kf(genid, subid, seed2);
[state, xe, ifail] = g05sa(nag_int(32), state);
xe = reshape(xe, 4, 8);

% Evaluate the interpolant
[q, qx, ifail] = e01tl(x, f, iq, rq, xe);

% Evaluate function at xe
c1  = cos(6*xe(1,:));
c2  = cos(6*xe(2,:));
c3  = 6 + 6*(3*xe(3,:)-1).^2;
c4  = 1.25 + cos(5.4*xe(4,:));
fun = c4.*c1.*c2./c3;

fprintf('\n i |  f(i)       q(i)   |f(i)-q(i)|\n');
fprintf('---|--------------------+---------+\n');
for i=1:8
  fprintf(' %d |%8.4f  %8.4f  %8.4f \n', i, fun(i), q(i), abs(fun(i)-q(i)));
end
```

9.2 Program Results

```
     e01tl example results

 i |  f(i)       q(i)   |f(i)-q(i)|
---|--------------------+---------+
 1 | -0.0189   -0.0394    0.0205
 2 | -0.0186    0.0967    0.1153
 3 |  0.1147    0.0606    0.0541
 4 |  0.0096   -0.1313    0.1409
 5 | -0.1354   -0.1878    0.0524
 6 |  0.0022   -0.1595    0.1617
 7 | -0.0095   -0.1179    0.1084
 8 |  0.0113   -0.3950    0.4063
```