NAG Toolbox

nag_interp_4d_scat_shep (e01tk)

1 Purpose

nag_interp_4d_scat_shep (e01tk) generates a four-dimensional interpolant to a set of scattered data points, using a modified Shepard method.

2 Syntax

```
[iq, rq, ifail] = nag_interp_4d_scat_shep(x, f, nw, nq, 'm', m)
[iq, rq, ifail] = e01tk(x, f, nw, nq, 'm', m)
```

3 Description

nag_interp_4d_scat_shep (e01tk) constructs a smooth function $Q(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^4$ which interpolates a set of m scattered data points $(\mathbf{x}_r, f_r)$, for $r = 1, 2, \ldots, m$, using a modification of Shepard's method. The surface is continuous and has continuous first partial derivatives.

The basic Shepard method, which is a generalization of the two-dimensional method described in Shepard (1968), interpolates the input data with the weighted mean

$$Q(\mathbf{x}) = \frac{\sum_{r=1}^{m} w_r(\mathbf{x}) q_r}{\sum_{r=1}^{m} w_r(\mathbf{x})},$$

where $q_r = f_r$, $w_r(\mathbf{x}) = \dfrac{1}{d_r^2}$ and $d_r^2 = \|\mathbf{x} - \mathbf{x}_r\|_2^2$.

The basic method is global in that the interpolated value at any point depends on all the data, but nag_interp_4d_scat_shep (e01tk) uses a modification (see Franke and Nielson (1980) and Renka (1988a)), whereby the method becomes local by adjusting each $w_r(\mathbf{x})$ to be zero outside a hypersphere with centre $\mathbf{x}_r$ and some radius R_w. Also, to improve the performance of the basic method, each q_r above is replaced by a function $q_r(\mathbf{x})$, which is a quadratic fitted by weighted least squares to data local to $\mathbf{x}_r$ and forced to interpolate $(\mathbf{x}_r, f_r)$. In this context, a point $\mathbf{x}$ is defined to be local to another point if it lies within some distance R_q of it.

The efficiency of nag_interp_4d_scat_shep (e01tk) is enhanced by using a cell method for nearest neighbour searching due to Bentley and Friedman (1979) with a cell density of 3.

The radii R_w and R_q are chosen to be just large enough to include N_w and N_q data points, respectively, for user-supplied constants N_w and N_q. Default values of these arguments are provided by the function, and advice on alternatives is given in Section 9.2.

nag_interp_4d_scat_shep (e01tk) is derived from the new implementation of QSHEP3 described by Renka (1988b). It uses the modification for high-dimensional interpolation described by Berry and Minser (1999).

Values of the interpolant $Q(\mathbf{x})$ generated by nag_interp_4d_scat_shep (e01tk), and its first partial derivatives, can subsequently be evaluated for points in the domain of the data by a call to nag_interp_4d_scat_shep_eval (e01tl).

4 References

Bentley J L and Friedman J H (1979) Data structures for range searching *ACM Comput. Surv.* **11** 397–409

Berry M W, Minser K S (1999) Algorithm 798: high-dimensional interpolation using the modified Shepard method *ACM Trans. Math. Software* **25** 353–366

Franke R and Nielson G (1980) Smooth interpolation of large sets of scattered data *Internat. J. Num. Methods Engrg.* **15** 1691–1704

Renka R J (1988a) Multivariate interpolation of large sets of scattered data *ACM Trans. Math. Software* **14** 139–148

Renka R J (1988b) Algorithm 661: QSHEP3D: Quadratic Shepard method for trivariate interpolation of scattered data *ACM Trans. Math. Software* **14** 151–152

Shepard D (1968) A two-dimensional interpolation function for irregularly spaced data *Proc. 23rd Nat. Conf. ACM* 517–523 Brandon/Systems Press Inc., Princeton

5 Parameters

5.1 Compulsory Input Parameters

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

x$(1:4, r)$ must be set to the Cartesian coordinates of the data point $\mathbf{x}_r$, for $r = 1, 2, \ldots, m$.

Constraint: these coordinates must be distinct, and must not all lie on the same three-dimensional hypersurface.

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

f(r) must be set to the data value f_r, for $r = 1, 2, \ldots, m$.

3: **nw** – INTEGER

The number N_w of data points that determines each radius of influence R_w, appearing in the definition of each of the weights w_r, for $r = 1, 2, \ldots, m$ (see Section 3). Note that R_w is different for each weight. If **nw** ≤ 0 the default value **nw** $= \min(32, \mathbf{m} - 1)$ is used instead.

Constraint: **nw** $\leq \min(50, \mathbf{m} - 1)$.

4: **nq** – INTEGER

The number N_q of data points to be used in the least squares fit for coefficients defining the quadratic functions $q_r(\mathbf{x})$ (see Section 3). If **nq** ≤ 0 the default value **nq** $= \min(38, \mathbf{m} - 1)$ is used instead.

Constraint: **nq** ≤ 0 or $14 \leq$ **nq** $\leq \min(50, \mathbf{m} - 1)$.

5.2 Optional Input Parameters

1: **m** – INTEGER

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

m, the number of data points.

Constraint: **m** ≥ 16.

5.3 Output Parameters

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

Integer data defining the interpolant $Q(\mathbf{x})$.

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

Real data defining the interpolant $Q(\mathbf{x})$.

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: **m** ≥ 16.

Constraint: **nq** ≤ 0 or **nq** ≥ 14.

Constraint: **nq** $\leq \min(50, \mathbf{m} - 1)$.

Constraint: **nw** $\leq \min(50, \mathbf{m} - 1)$.

ifail $= 2$

There are duplicate nodes in the dataset.

ifail $= 3$

On entry, all the data points lie on the same three-dimensional hypersurface. No unique solution exists.

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

On successful exit, the function generated interpolates the input data exactly and has quadratic precision. Overall accuracy of the interpolant is affected by the choice of arguments **nw** and **nq** as well as the smoothness of the function represented by the input data.

8 Further Comments

8.1 Timing

The time taken for a call to nag_interp_4d_scat_shep (e01tk) will depend in general on the distribution of the data points and on the choice of N_w and N_q parameters. If the data points are uniformly randomly distributed, then the time taken should be $O(m)$. At worst $O(m^2)$ time will be required.

8.2 Choice of N_w and N_q

Default values of the arguments N_w and N_q may be selected by calling nag_interp_4d_scat_shep (e01tk) with **nw** ≤ 0 and **nq** ≤ 0. These default values may well be satisfactory for many applications.

If non-default values are required they must be supplied to nag_interp_4d_scat_shep (e01tk) through positive values of **nw** and **nq**. Increasing these argument values makes the method less local. This may increase the accuracy of the resulting interpolant at the expense of increased computational cost.

9 Example

This program reads in a set of 30 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 points.

Note that this example is not typical of a realistic problem: the number of data points would normally be larger.

See also Section 10 in nag_interp_4d_scat_shep_eval (e01tl).

9.1 Program Text

```
function e01tk_example

fprintf('e01tk example results\n\n');

x = [0.81, 0.91, 0.13, 0.91, 0.63, 0.10, 0.28, 0.55, 0.96, 0.96, ...
     0.16, 0.97, 0.96, 0.49, 0.80, 0.14, 0.42, 0.92, 0.79, 0.96, ...
     0.66, 0.04, 0.85, 0.93, 0.68, 0.76, 0.74, 0.39, 0.66, 0.17;
     0.15, 0.96, 0.88, 0.49, 0.41, 0.13, 0.93, 0.01, 0.19, 0.32, ...
     0.05, 0.14, 0.73, 0.48, 0.34, 0.24, 0.45, 0.19, 0.32, 0.26, ...
     0.83, 0.70, 0.33, 0.58, 0.29, 0.26, 0.26, 0.68, 0.52, 0.08;
     0.44, 0.00, 0.22, 0.39, 0.72, 0.77, 0.24, 0.04, 0.95, 0.53, ...
     0.16, 0.36, 0.28, 0.58, 0.64, 0.12, 0.03, 0.48, 0.15, 0.93, ...
     0.41, 0.40, 0.15, 0.88, 0.88, 0.09, 0.33, 0.69, 0.17, 0.35;
     0.83, 0.09, 0.21, 0.79, 0.68, 0.47, 0.90, 0.41, 0.66, 0.96, ...
     0.30, 0.72, 0.75, 0.19, 0.57, 0.06, 0.68, 0.67, 0.13, 0.89, ...
     0.17, 0.54, 0.03, 0.81, 0.60, 0.41, 0.64, 0.37, 1.00, 0.71];

f = [6.3900;
    2.5000;
    9.3400;
    7.5200;
    6.9100;
    4.6800;
   45.4000;
    5.4800;
    2.7500;
    7.4300;
    6.0500;
    5.7700;
    8.6800;
    2.3800;
    3.7000;
    1.3400;
   15.1800;
    4.3500;
    1.5000;
    3.4300;
    3.1000;
   14.3300;
    0.3500;
    4.3000;
    3.7700;
    4.1600;
    6.7500;
    5.2200;
   16.2300,
   10.6200];

xe = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9;
      0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9;
      0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9;
```

```
      0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9];

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

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

fprintf('\n   |  Interpolated Evaluation Points         |  Values\n');
fprintf('---|-----------------------------------------+--------\n');
fprintf('i  |  xe(i,1)   xe(i,2)   xe(i,3)   xe(i,4)  | q(i)\n');
fprintf('---|-----------------------------------------+--------\n');
for i=1:9
  fprintf(' %d |%8.4f  %8.4f  %8.4f  %8.4f  %8.4f \n', i, xe(:, i), q(i));
end
```

9.2 Program Results

```
     e01tk example results

   |  Interpolated Evaluation Points         |  Values
---|-----------------------------------------+--------
i  |  xe(i,1)   xe(i,2)   xe(i,3)   xe(i,4)  | q(i)
---|-----------------------------------------+--------
 1 |  0.1000    0.1000    0.1000    0.1000    2.7195
 2 |  0.2000    0.2000    0.2000    0.2000    4.3110
 3 |  0.3000    0.3000    0.3000    0.3000    5.5380
 4 |  0.4000    0.4000    0.4000    0.4000    6.5540
 5 |  0.5000    0.5000    0.5000    0.5000    7.5910
 6 |  0.6000    0.6000    0.6000    0.6000    8.7447
 7 |  0.7000    0.7000    0.7000    0.7000   10.0457
 8 |  0.8000    0.8000    0.8000    0.8000   11.5797
 9 |  0.9000    0.9000    0.9000    0.9000   13.1997
```