NAG Toolbox

nag_interp_nd_scat_shep (e01zm)

1 Purpose

nag_interp_nd_scat_shep (e01zm) generates a multidimensional interpolant to a set of scattered data points, using a modified Shepard method. When the number of dimensions is no more than five, there are corresponding functions in Chapter E01 which are specific to the given dimensionality. nag_interp_2d_scat_shep (e01sg) generates the two-dimensional interpolant, while nag_interp_3d_scat_shep (e01tg), nag_interp_4d_scat_shep (e01tk) and nag_interp_5d_scat_shep (e01tm) generate the three-, four- and five-dimensional interpolants respectively.

2 Syntax

```
[iq, rq, ifail] = nag_interp_nd_scat_shep(x, f, 'd', d, 'm', m, 'nw', nw, 'nq', nq)
[iq, rq, ifail] = e01zm(x, f, 'd', d, 'm', m, 'nw', nw, 'nq', nq)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 24: **nw** and **nq** were made optional.

3 Description

nag_interp_nd_scat_shep (e01zm) constructs a smooth function $Q(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$ 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 $q_r = f_r$, $w_r(\mathbf{x}) = \dfrac{1}{\|\mathbf{x} - \mathbf{x}_r\|_2^2}$.

The basic method is global in that the interpolated value at any point depends on all the data, but nag_interp_nd_scat_shep (e01zm) 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_nd_scat_shep (e01zm) 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 parameters are provided, and advice on alternatives is given in Section 9.2.

nag_interp_nd_scat_shep (e01zm) 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_nd_scat_shep (e01zm), and its first partial derivatives, can subsequently be evaluated for points in the domain of the data by a call to nag_interp_nd_scat_shep_eval (e01zn).

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

$\mathbf{x}(1:\mathbf{d}, r)$ 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 $(d-1)$-dimensional hypersurface.

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

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

5.2 Optional Input Parameters

1: **d** – INTEGER

Default: the first dimension of the array **x**.

d, the number of dimensions.

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

2: **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.

Note: on the basis of experimental results reported in Berry and Minser (1999), when $\mathbf{d} \geq 5$ it is recommended to use $\mathbf{m} \geq 4000$.

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

3: **nw** – INTEGER

Default: $\mathbf{nw} = -1$

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 $\mathbf{nw} \leq 0$ the default value $\mathbf{nw} = \min(2 \times (\mathbf{d}+1) \times (\mathbf{d}+2), \mathbf{m}-1)$ is used instead.

Constraint: $\mathbf{nw} \leq \mathbf{m} - 1$.

4: **nq** – INTEGER

Default: $\mathbf{nq} = -1$

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 $\mathbf{nq} \leq 0$ the default value $\mathbf{nq} = \min((\mathbf{d}+1) \times (\mathbf{d}+2) \times 6/5, \mathbf{m}-1)$ is used instead.

Constraint: $\mathbf{nq} \leq 0$ or $(\mathbf{d}+1) \times (\mathbf{d}+2)/2 - 1 \leq \mathbf{nq} \leq \mathbf{m} - 1$.

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**$(*)$ – REAL (KIND=nag_wp) array

The dimension of the array **rq** will be $((\mathbf{d}+1) \times (\mathbf{d}+2)/2) \times \mathbf{m} + 2 \times \mathbf{d} + 1$

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

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

Constraint: $\mathbf{nq} \leq 0$ or $\mathbf{nq} \geq (\mathbf{d}+1) \times (\mathbf{d}+2)/2 - 1$.

Constraint: $\mathbf{nq} \leq \mathbf{m} - 1$.

Constraint: $\mathbf{nw} \leq \mathbf{m} - 1$.

On entry, $((\mathbf{d}+1) \times (\mathbf{d}+2)/2) \times \mathbf{m} + 2 \times \mathbf{d} + 1$ exceeds the largest machine integer.

ifail $= 2$

There are duplicate nodes in the dataset.

ifail $= 3$

On entry, all the data points lie on the same 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

In experiments undertaken by Berry and Minser (1999), the accuracies obtained for a conditional function resulting in sharp functional transitions were of the order of 10^{-1} at best. In other cases in these experiments, the function generated interpolates the input data with maximum absolute error of the order of 10^{-2}.

8 Further Comments

8.1 Timing

The time taken for a call to nag_interp_nd_scat_shep (e01zm) 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 $O\left(m^2\right)$ time will be required.

8.2 Choice of N_w and N_q

Default values of the parameters N_w and N_q may be selected by calling nag_interp_nd_scat_shep (e01zm) with $\mathbf{nw} \leq 0$ and $\mathbf{nq} \leq 0$. These default values may well be satisfactory for many applications.

If non-default values are required they must be supplied to nag_interp_nd_scat_shep (e01zm) 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. The default values $\mathbf{nw} = \min(2 \times (\mathbf{d}+1) \times (\mathbf{d}+2), \mathbf{m}-1)$ and $\mathbf{nq} = \min((\mathbf{d}+1) \times (\mathbf{d}+2) \times 6/5, \mathbf{m}-1)$ have been chosen on the basis of experimental results reported in Renka (1988a) and Berry and Minser (1999). For further advice on the choice of these arguments see Renka (1988a) and Berry and Minser (1999).

9 Example

This program reads in a set of 30 data points and calls nag_interp_nd_scat_shep (e01zm) to construct an interpolating function $Q(\mathbf{x})$. It then calls nag_interp_nd_scat_shep_eval (e01zn) 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 very much larger.

See also Section 10 in nag_interp_nd_scat_shep_eval (e01zn).

9.1 Program Text

```
function e01zm_example

fprintf('e01zm example results\n\n');

% Data: 30 6-dimensional points and function values
nd = 6;
npts = 30;
x = zeros(npts,nd);
x = [0.81, 0.15, 0.44, 0.83, 0.21, 0.64;
     0.91, 0.96, 0.00, 0.09, 0.98, 0.37;
     0.13, 0.88, 0.22, 0.21, 0.73, 1.00;
     0.91, 0.49, 0.39, 0.79, 0.47, 0.71;
     0.63, 0.41, 0.72, 0.68, 0.65, 0.83;
     0.10, 0.13, 0.77, 0.47, 0.22, 0.09;
```

```
     0.28, 0.93, 0.24, 0.90, 0.96, 0.21;
     0.55, 0.01, 0.04, 0.41, 0.26, 0.79;
     0.96, 0.19, 0.95, 0.66, 0.99, 0.68;
     0.96, 0.32, 0.53, 0.96, 0.84, 0.47;
     0.16, 0.05, 0.16, 0.30, 0.58, 0.90;
     0.97, 0.14, 0.36, 0.72, 0.78, 0.06;
     0.96, 0.73, 0.28, 0.75, 0.28, 0.68;
     0.49, 0.48, 0.58, 0.19, 0.25, 0.67;
     0.80, 0.34, 0.64, 0.57, 0.08, 0.13;
     0.14, 0.24, 0.12, 0.06, 0.63, 0.89;
     0.42, 0.45, 0.03, 0.68, 0.66, 0.17;
     0.92, 0.19, 0.48, 0.67, 0.28, 0.54;
     0.79, 0.32, 0.15, 0.13, 0.40, 0.03;
     0.96, 0.26, 0.93, 0.89, 0.61, 0.81;
     0.66, 0.83, 0.41, 0.17, 0.09, 0.60;
     0.04, 0.70, 0.40, 0.54, 0.37, 0.41;
     0.85, 0.33, 0.15, 0.03, 0.36, 5.77;
     0.93, 0.58, 0.88, 0.81, 0.40, 0.66;
     0.68, 0.29, 0.88, 0.60, 0.47, 0.96;
     0.76, 0.26, 0.09, 0.41, 0.14, 0.30;
     0.74, 0.26, 0.33, 0.64, 0.36, 0.72;
     0.39, 0.68, 0.69, 0.37, 0.12, 0.75;
     0.66, 0.52, 0.17, 1.00, 0.43, 0.19;
     0.17, 0.08, 0.35, 0.71, 0.17, 0.57];
x=x';
f = [6.39;  2.50; 9.34; 7.52; 6.91; 4.68; 45.40; 5.48;  2.75;  7.43; ...
     6.05;  0.41; 8.68; 2.38; 3.70; 1.34; 15.18; 4.35;  1.50;  3.43; ...
     3.10; 14.33; 0.35; 4.30; 3.77; 4.16;  6.75; 5.22; 16.23; 10.62];

% Interpolation points
nip = 6;
xe = zeros(nd,nip);
for i=1:npts
  xe(1:nd,i) = 0.1*i;
end

% Generate the interpolant
[iq, rq, ifail] = e01zm(x, f);

% Evaluate the interpolant
[q, qx, ifail] = e01zn(x, f, iq, rq, xe);

T = array2table(double([xe' q]));
T.Properties.VariableNames = {'x_1','x_2','x_3','x_4','x_5','x_6','q'};

fprintf('Interpolated Values\n\n');
disp(T(1:nip,1:nd+1));
```

9.2 Program Results

```
     e01zm example results

Interpolated Values

    x_1    x_2    x_3    x_4    x_5    x_6       q
    ___    ___    ___    ___    ___    ___    ________

    0.1    0.1    0.1    0.1    0.1    0.1     -6.5071
    0.2    0.2    0.2    0.2    0.2    0.2     -3.5708
    0.3    0.3    0.3    0.3    0.3    0.3    -0.80057
    0.4    0.4    0.4    0.4    0.4    0.4      1.7257
    0.5    0.5    0.5    0.5    0.5    0.5      3.9278
    0.6    0.6    0.6    0.6    0.6    0.6      5.8951
```