NAG Library Function Document

nag_nd_shep_interp (e01zmc)

1 Purpose

nag_nd_shep_interp (e01zmc) 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_2d_shep_interp (e01sgc) generates the two-dimensional interpolant, while nag_3d_shep_interp (e01tgc), nag_4d_shep_interp (e01tkc) and nag_5d_shep_interp (e01tmc) generate the three-, four- and five-dimensional interpolants respectively.

2 Specification

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

```
void nag_nd_shep_interp (Integer d, Integer m, const double x[],
     const double f[], Integer nw, Integer nq, Integer iq[], double rq[],
     NagError *fail)
```

3 Description

nag_nd_shep_interp (e01zmc) 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_nd_shep_interp (e01zmc) 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_nd_shep_interp (e01zmc) 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_nd_shep_interp (e01zmc) 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_nd_shep_interp (e01zmc), and its first partial derivatives, can subsequently be evaluated for points in the domain of the data by a call to nag_nd_shep_eval (e01znc).

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 Arguments

1: **d** – Integer *Input*

On entry: d, the number of dimensions.

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

2: **m** – Integer *Input*

On entry: 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: $\mathbf{x}[\mathbf{d} \times \mathbf{m}]$ – const double *Input*

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

On entry: the **d** components of the first data point must be stored in elements $0, 1, \ldots, \mathbf{d}-1$ of **x**. The second data point must be stored in elements $\mathbf{d}, \mathbf{d}+1, \ldots, 2 \times \mathbf{d} - 1$ of **x**, and so on. In general, the **m** data points must be stored in $\mathbf{x}[i \times \mathbf{d} + j]$, for $i = 0, 1, \ldots, \mathbf{m}-1$ and $j = 0, 1, \ldots, \mathbf{d}-1$.

Constraint: these coordinates must be distinct, and must not all lie on the same $(d-1)$-dimensional hypersurface.

4: $\mathbf{f}[\mathbf{m}]$ – const double *Input*

On entry: $\mathbf{f}[r-1]$ must be set to the data value f_r, for $r = 1, 2, \ldots, m$.

5: **nw** – Integer *Input*

On entry: 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.

Suggested value: $\mathbf{nw} = -1$.

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

6: **nq** – Integer *Input*

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

Suggested value: $\mathbf{nq} = -1$.

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

7: **iq**$[\mathbf{2} \times \mathbf{m} + \mathbf{1}]$ – Integer *Output*

On exit: integer data defining the interpolant $Q(\mathbf{x})$.

8: **rq**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **rq** must be at least $((\mathbf{d}+1) \times (\mathbf{d}+2)/2) \times \mathbf{m} + 2 \times \mathbf{d} + 1$.

On exit: real data defining the interpolant $Q(\mathbf{x})$.

9: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_DATA_HYPERSURFACE

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

NE_DUPLICATE_NODE

There are duplicate nodes in the dataset. $\mathbf{x}[(i-1) \times \mathbf{d} + k - 1] = \mathbf{x}[(j-1) \times \mathbf{d} + k - 1]$, for $i = \langle value \rangle$, $j = \langle value \rangle$ and $k = 1, 2, \ldots, \mathbf{d}$. The interpolant cannot be derived.

NE_INT

On entry, $\mathbf{d} = \langle value \rangle$.
Constraint: $\mathbf{d} \ge 2$.

NE_INT_2

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

On entry, $\mathbf{m} = \langle value \rangle$ and $\mathbf{d} = \langle value \rangle$.
Constraint: $\mathbf{m} \ge (\mathbf{d}+1) \times (\mathbf{d}+2)/2 + 2$.

On entry, $\mathbf{nq} = \langle value \rangle$ and $\mathbf{d} = \langle value \rangle$.
Constraint: $\mathbf{nq} \leq 0$ or $\mathbf{nq} \geq (\mathbf{d}+1) \times (\mathbf{d}+2)/2 - 1$.

On entry, $\mathbf{nq} = \langle value \rangle$ and $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{nq} \leq \mathbf{m} - 1$.

On entry, $\mathbf{nw} = \langle value \rangle$ and $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{nw} \leq \mathbf{m} - 1$.

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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

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 Parallelism and Performance

nag_nd_shep_interp (e01zmc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_nd_shep_interp (e01zmc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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

9.1 Timing

The time taken for a call to nag_nd_shep_interp (e01zmc) 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.

9.2 Choice of N_w and N_q

Default values of the parameters N_w and N_q may be selected by calling nag_nd_shep_interp (e01zmc) 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_nd_shep_interp (e01zmc) 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).

10 Example

This program reads in a set of 30 data points and calls nag_nd_shep_interp (e01zmc) to construct an interpolating function $Q(\mathbf{x})$. It then calls nag_nd_shep_eval (e01znc) 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_nd_shep_eval (e01znc).

10.1 Program Text

```
/* nag_nd_shep_interp (e01zmc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage01.h>

int main(void)
{
  /* Scalars */
  Integer exit_status = 0;
  Integer d, i, j, liq, lrq, m, n, nq, nw;
  /* Arrays */
  double *f = 0, *q = 0, *qx = 0, *rq = 0, *x = 0, *xe = 0;
  Integer *iq = 0;
  /* NAG types */
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_nd_shep_interp (e01zmc) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

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

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

  /* Input the data points X and F. */
  for (i = 0; i < m; i++) {
#ifdef _WIN32
```

```
    for (j = 0; j < d; j++)
      scanf_s("%lf", &x[i * d + j]);
#else
    for (j = 0; j < d; j++)
      scanf("%lf", &x[i * d + j]);
#endif
#ifdef _WIN32
    scanf_s("%lf%*[^\n] ", &f[i]);
#else
    scanf("%lf%*[^\n] ", &f[i]);
#endif
  }

  /* Generate the interpolant using nag_nd_shep_interp (e01zmc):
     Interpolating functions, modified Shepard's method, d variables. */
  nq = 0;
  nw = 0;
  nag_nd_shep_interp(d, m, x, f, nw, nq, iq, rq, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_nd_shep_interp (e01zmc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Input the number of evaluation points and allocate arrays with lengths
     based on this. */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n]", &n);
#else
  scanf("%" NAG_IFMT "%*[^\n]", &n);
#endif
  if (!(xe = NAG_ALLOC((d) * (n), double)) ||
      !(q = NAG_ALLOC((n), double)) || !(qx = NAG_ALLOC((d) * (n), double)))
  {
    printf("Allocation failure\n");
    exit_status = -2;
    goto END;
  }

  /* Input the evaluation points. */
  for (i = 0; i < n; i++) {
#ifdef _WIN32
    for (j = 0; j < d; j++)
      scanf_s("%lf", &xe[i * d + j]);
#else
    for (j = 0; j < d; j++)
      scanf("%lf", &xe[i * d + j]);
#endif
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif
  }
  /* Evaluate the interpolant using nag_nd_shep_eval (e01znc), at given
   * interpolated values, where interpolant previously computed by
   * nag_nd_shep_interp (e01zmc). */
  nag_nd_shep_eval(d, m, x, f, iq, rq, n, xe, q, qx, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_nd_shep_eval (e01znc).\n%s\n", fail.message);
    exit_status = 2;
    goto END;
  }

  /* Print evaluation points and interpolated function values at those points */

  /* Header */
  printf("\n%4s|%39s%23s|%8s", "", "Interpolated Evaluation Points", "",
         "Values\n");
  printf(" ---|---------------------------------------------------------------");
  printf("+-------\n");
  printf(" i  |  ");
  for (i = 0; i < d; i++)
    printf("XE(%1" NAG_IFMT ",i)   ", i);
  printf("|  q[i]\n");
  printf(" ---|----------------------------------------------------------------");
  printf("--------\n");

  /* Results */
  for (i = 0; i < n; i++) {
    printf(" %1" NAG_IFMT " ", i);
    for (j = 0; j < d; j++)
      printf("%10.4f", xe[i * d + j]);
    printf("  %10.4f\n", q[i]);
  }

END:
  NAG_FREE(f);
  NAG_FREE(q);
  NAG_FREE(qx);
  NAG_FREE(rq);
  NAG_FREE(x);
  NAG_FREE(xe);
  NAG_FREE(iq);

  return exit_status;
}
```

10.2 Program Data

```
nag_nd_shep_interp (e01zmc) Example Program Data
 6   30                                    : d, m
 0.81  0.15  0.44  0.83  0.21  0.64  6.39
 0.91  0.96  0.00  0.09  0.98  0.37  2.50
 0.13  0.88  0.22  0.21  0.73  1.00  9.34
 0.91  0.49  0.39  0.79  0.47  0.71  7.52
 0.63  0.41  0.72  0.68  0.65  0.83  6.91
 0.10  0.13  0.77  0.47  0.22  0.09  4.68
 0.28  0.93  0.24  0.90  0.96  0.21 45.40
 0.55  0.01  0.04  0.41  0.26  0.79  5.48
 0.96  0.19  0.95  0.66  0.99  0.68  2.75
 0.96  0.32  0.53  0.96  0.84  0.47  7.43
 0.16  0.05  0.16  0.30  0.58  0.90  6.05
 0.97  0.14  0.36  0.72  0.78  0.06  0.41
 0.96  0.73  0.28  0.75  0.28  0.68  8.68
 0.49  0.48  0.58  0.19  0.25  0.67  2.38
 0.80  0.34  0.64  0.57  0.08  0.13  3.70
 0.14  0.24  0.12  0.06  0.63  0.89  1.34
 0.42  0.45  0.03  0.68  0.66  0.17 15.18
 0.92  0.19  0.48  0.67  0.28  0.54  4.35
 0.79  0.32  0.15  0.13  0.40  0.03  1.50
 0.96  0.26  0.93  0.89  0.61  0.81  3.43
 0.66  0.83  0.41  0.17  0.09  0.60  3.10
 0.04  0.70  0.40  0.54  0.37  0.41 14.33
 0.85  0.33  0.15  0.03  0.36  5.77  0.35
 0.93  0.58  0.88  0.81  0.40  0.66  4.30
 0.68  0.29  0.88  0.60  0.47  0.96  3.77
 0.76  0.26  0.09  0.41  0.14  0.30  4.16
 0.74  0.26  0.33  0.64  0.36  0.72  6.75
 0.39  0.68  0.69  0.37  0.12  0.75  5.22
 0.66  0.52  0.17  1.00  0.43  0.19 16.23
 0.17  0.08  0.35  0.71  0.17  0.57 10.62 : x[], f[]
6                                          : n, no. of evaluation points
 0.10  0.10  0.10  0.10  0.10  0.10
 0.20  0.20  0.20  0.20  0.20  0.20
 0.30  0.30  0.30  0.30  0.30  0.30
 0.40  0.40  0.40  0.40  0.40  0.40
 0.50  0.50  0.50  0.50  0.50  0.50
 0.60  0.60  0.60  0.60  0.60  0.60       : xe[1:n*d]
```

10.3 Program Results

```
nag_nd_shep_interp (e01zmc) Example Program Results

    |         Interpolated Evaluation Points                         | Values
 ---|----------------------------------------------------------------+-------
 i  |  XE(0,i)   XE(1,i)   XE(2,i)   XE(3,i)   XE(4,i)   XE(5,i)    |  q[i]
 ---|------------------------------------------------------------------------
 0     0.1000    0.1000    0.1000    0.1000    0.1000    0.1000      -7.2059
 1     0.2000    0.2000    0.2000    0.2000    0.2000    0.2000      -3.9343
 2     0.3000    0.3000    0.3000    0.3000    0.3000    0.3000      -0.9674
 3     0.4000    0.4000    0.4000    0.4000    0.4000    0.4000       1.6680
 4     0.5000    0.5000    0.5000    0.5000    0.5000    0.5000       3.9251
 5     0.6000    0.6000    0.6000    0.6000    0.6000    0.6000       5.9318
```