NAG Library Function Document

nag_4d_shep_interp (e01tkc)

1 Purpose
nag_4d_shep_interp (e01tkc) generates a four-dimensional interpolant to a set of scattered data points, using a modified Shepard method.

2 Specification
#include <nag.h>
#include <nage01.h>

void nag_4d_shep_interp (Integer m, const double x[], const double f[], Integer nw, Integer nq, Integer iq[], double rq[], NagError *fail)

3 Description
nag_4d_shep_interp (e01tkc) constructs a smooth function \( Q(x) \), \( x \in \mathbb{R}^4 \) which interpolates a set of \( m \) scattered data points \((x_r, f_r)\), 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

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

where \( q_r = f_r \), \( w_r(x) = \frac{1}{d^2} \) and \( d^2 = \|x - x_r\|^2 \).

The basic method is global in that the interpolated value at any point depends on all the data, but nag_4d_shep_interp (e01tkc) uses a modification (see Franke and Nielson (1980) and Renka (1988a)), whereby the method becomes local by adjusting each \( w_r(x) \) to be zero outside a hypersphere with centre \( x_r \) and some radius \( R_w \). Also, to improve the performance of the basic method, each \( q_r \) above is replaced by a function \( q_r(x) \), which is a quadratic fitted by weighted least squares to data local to \( x_r \) and forced to interpolate \((x_r, f_r)\). In this context, a point \( x \) is defined to be local to another point if it lies within some distance \( R_q \) of it.

The efficiency of nag_4d_shep_interp (e01tkc) 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_4d_shep_interp (e01tkc) 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 \( Q(x) \) generated by nag_4d_shep_interp (e01tkc), and its first partial derivatives, can subsequently be evaluated for points in the domain of the data by a call to nag_4d_shep_eval (e01tlc).
4 References


5 Arguments

1: \( m \) – Integer  
   \text{Input}  
   \text{On entry: } m, \text{ the number of data points.}  
   \text{Constraint: } m \geq 16.

2: \( x[4 \times m] \) – const double  
   \text{Input}  
   \text{On entry: } x[(j-1) \times 4 + i - 1].  
   \text{Note: } the \ (i,j)\text{th element of the matrix } X \text{ is stored in } x[j-1] \times 4 + i - 1.  
   \text{On entry: } x[(r-1) \times 4], \ldots, x[(r-1) \times 4 + 3] \text{ must be set to the Cartesian coordinates of the data point } x_r, \text{ for } r = 1, 2, \ldots, m.  
   \text{Constraint: } \text{these coordinates must be distinct, and must not all lie on the same three-dimensional hypersurface.}

3: \( f[m] \) – const double  
   \text{Input}  
   \text{On entry: } f[r-1] \text{ must be set to the data value } f_r, \text{ for } r = 1, 2, \ldots, m.

4: \( nw \) – Integer  
   \text{Input}  
   \text{On entry: } the \text{ number } N_w \text{ of data points that determines each radius of influence } R_w, \text{ appearing in the definition of each of the weights } w_r, \text{ for } r = 1, 2, \ldots, m (\text{see Section 3}). \text{Note that } R_w \text{ is different for each weight. If } nw \leq 0 \text{ the default value } nw = \min(32, m - 1) \text{ is used instead.}  
   \text{Constraint: } nw \leq \min(50, m - 1).

5: \( nq \) – Integer  
   \text{Input}  
   \text{On entry: } the \text{ number } N_q \text{ of data points to be used in the least squares fit for coefficients defining the quadratic functions } q_r(x) \text{ (see Section 3). If } nq \leq 0 \text{ the default value } nq = \min(38, m - 1) \text{ is used instead.}  
   \text{Constraint: } nq \leq 0 \text{ or } 14 \leq nq \leq \min(50, m - 1).

6: \( iq[2 \times m + 1] \) – Integer  
   \text{Output}  
   \text{On exit: } integer \text{ data defining the interpolant } Q(x).

7: \( rq[15 \times m + 9] \) – double  
   \text{Output}  
   \text{On exit: } real \text{ data defining the interpolant } Q(x).
6 Error Indicators and Warnings

**NE_ALLOCFAIL**

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument (value) had an illegal value.

**NE_DATA_HYPERSURFACE**

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

**NE_DUPLICATE_NODE**

There are duplicate nodes in the dataset. \( x[(k - 1) \times 4 + i - 1] = x[(k - 1) \times 4 + j - 1] \), for \( i = \text{(value)} \), \( j = \text{(value)} \) and \( k = 1, 2, \ldots, 4 \). The interpolant cannot be derived.

**NE_INT**

On entry, \( m = \text{(value)} \).
Constraint: \( m \geq 16 \).

On entry, \( nq = \text{(value)} \).
Constraint: \( nq \leq 0 \) or \( nq \geq 14 \).

**NE_INT_2**

On entry, \( nq = \text{(value)} \) and \( m = \text{(value)} \).
Constraint: \( nq \leq \min(50, m - 1) \).

On entry, \( nw = \text{(value)} \) and \( m = \text{(value)} \).
Constraint: \( nw \leq \min(50, m - 1) \).

**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 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.

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 Parallelism and Performance

\texttt{nag_4d_shep_interp (e01tkc)} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\texttt{nag_4d_shep_interp (e01tkc)} 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 \texttt{nag_4d_shep_interp (e01tkc)} 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 arguments \(N_w\) and \(N_q\) may be selected by calling \texttt{nag_4d_shep_interp (e01tkc)} with \(nw \leq 0\) and \(nq \leq 0\). These default values, \(nw = \min(32, m - 1)\) and \(nq = \min(38, m - 1)\), may well be satisfactory for many applications.

If non-default values are required they must be supplied to \texttt{nag_4d_shep_interp (e01tkc)} 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.

10 Example

This program reads in a set of 30 data points and calls \texttt{nag_4d_shep_interp (e01tkc)} to construct an interpolating function \(Q(x)\). It then calls \texttt{nag_4d_shep_eval (e01tlc)} 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 \texttt{nag_4d_shep_eval (e01tlc)}.

10.1 Program Text

/* nag_4d_shep_interp (e01tkc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 23, 2010. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage01.h>

define X(I, J) x[I *4 + J]
define QX(I, J) qx[I *4 + J]
define XE(I, J) xe[I *4 + J]

int main(void)
{
    /* Scalars */
    Integer exit_status, i, j, m, n, nq, nw, liq, lrq;
    NagError fail;

    [...]
/* Arrays */
double  *f = 0, *q = 0, *qx = 0, *rq = 0, *xe = 0, *x = 0;
Integer  *iq = 0;

exit_status = 0;
INIT_FAIL(fail);

printf("nag_4d_shep_interp (e01tkc) Example Program Results\n");

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

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

/* Allocate memory */
lrq = 21 * m + 11;
liq = 2 * m + 1;
if (!(f = NAG_ALLOC(m, double)) || 
   !(x = NAG_ALLOC(m*4, double)) || 
   !(rq = NAG_ALLOC(lrq, double)) || 
   !(iq = NAG_ALLOC(liq, Integer)))
   {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
   }

/* Input the data points X and F. */
for (i = 0; i < m; ++i) 
   for (j = 0; j < 4; ++j) 
      {
         #ifdef _WIN32
         scanf_s("%lf", &X(i, j));
         #else
         scanf("%lf", &X(i, j));
         #endif
      }
      #ifdef _WIN32
      scanf_s("%lf%*[\n] ", &f[i]);
      #else
      scanf("%lf%*[\n] ", &f[i]);
      #endif
      
/* Generate the interpolant. */
        nq = 0;
        nw = 0;

        /* nag_4d_shep_interp (e01tkc).
         * Interpolating functions, modified Shepard’s method, four variables
         */
        nag_4d_shep_interp(m, x, f, nw, nq, iq, rq, &fail);
        if (fail.code != NE_NOERROR)
           {
               printf("Error from nag_4d_shep_interp (e01tkc).\n%s\n", fail.message);
               exit_status = 1;
               goto END;
           }

        /* Input the number of evaluation points. */
        endif

        Mark 25
e01tkc.5
/* Allocate memory for nag_4d_shep_eval (e01tlc) */
if (!(q = NAG_ALLOC(n, double)) ||
! (qx = NAG_ALLOC(n*4, double)) ||
! (xe = NAG_ALLOC(n*4, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Input the evaluation points. */
for (i = 0; i < n; ++i) {
    for (j = 0; j < 4; ++j) {
        #ifdef _WIN32
        scanf_s("%lf", &XE(i, j));
        #else
        scanf("%lf", &XE(i, j));
        #endif
    }
    #ifdef _WIN32
    scanf_s("%*[\n"]);
    #else
    scanf("%*[\n"]);
    #endif
}
/* nag_4d_shep_eval (e01tlc). */
*nag_4d_shep_interp (e01tkc), and first*
* derivatives, at oints in xe.*
fail.print = Nag_TRUE;
for (i = 0; i < n; ++i) {
    printf("%5d%8.2f%8.2f%8.2f\n", i, XE(i,0), XE(i,1), XE(i,2), XE(i,3), q[i]);
}
END:
NAG_FREE(f);
NAG_FREE(q);
NAG_FREE(qx);
NAG_FREE(rq);
NAG_FREE(xe);
NAG_FREE(iq);
return exit_status;
}

10.2 Program Data

nag_4d_shep_interp (e01tkc) Example Program Data
30 : number of data points
0.81 0.15 0.44 0.83 6.39 : x, f(x)
0.91 0.96 0.00 0.09 2.50
0.13 0.88 0.22 0.21 9.34
0.91 0.49 0.39 0.79 7.52
0.63 0.41 0.72 0.68 6.91
0.10 0.13 0.77 0.47 4.68
0.28 0.93 0.24 0.90 45.40
0.55 0.01 0.04 0.41 5.48
0.96 0.19 0.95 0.66 2.75
0.96 0.32 0.53 0.96 7.43
### 10.3 Program Results

**nag_4d_shep_interp (e01tkc) Example Program Results**

Evaluation of interpolant at various (4D) points

<table>
<thead>
<tr>
<th>pt.no.</th>
<th>point coordinates</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.10 0.10 0.10 0.10</td>
<td>2.7195</td>
</tr>
<tr>
<td>1</td>
<td>0.20 0.20 0.20 0.20</td>
<td>4.3110</td>
</tr>
<tr>
<td>2</td>
<td>0.30 0.30 0.30 0.30</td>
<td>5.5380</td>
</tr>
<tr>
<td>3</td>
<td>0.40 0.40 0.40 0.40</td>
<td>6.5540</td>
</tr>
<tr>
<td>4</td>
<td>0.50 0.50 0.50 0.50</td>
<td>7.5910</td>
</tr>
<tr>
<td>5</td>
<td>0.60 0.60 0.60 0.60</td>
<td>8.7447</td>
</tr>
<tr>
<td>6</td>
<td>0.70 0.70 0.70 0.70</td>
<td>10.0457</td>
</tr>
<tr>
<td>7</td>
<td>0.80 0.80 0.80 0.80</td>
<td>11.5797</td>
</tr>
<tr>
<td>8</td>
<td>0.90 0.90 0.90 0.90</td>
<td>13.1997</td>
</tr>
</tbody>
</table>