nag_3d_shep_interp (e01tgc) generates a three-dimensional interpolant to a set of scattered data points, using a modified Shepard method.

This function constructs a smooth function $Q(x, y, z)$ which interpolates a set of $m$ scattered data points $(x_r, y_r, z_r, f_r)$, 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

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

where $q_r = f_r$ and $w_r(x, y, z) = \frac{1}{d_r^2}$ and $d_r^2 = (x - x_r)^2 + (y - y_r)^2 + (z - z_r)^2$.

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

The efficiency of the function is further enhanced by using a cell method for nearest neighbour searching due to Bentley and Friedman (1979).

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 by the function, and advice on alternatives is given in Section 8.2.

This function is derived from the function QSHEP3 described by Renka (1988b).

Values of the interpolant $Q(x, y, z)$ generated by this function, and its first partial derivatives, can subsequently be evaluated for points in the domain of the data by a call to nag_3d_shep_eval (e01thc).

1 Purpose

nag_3d_shep_interp (e01tgc) generates a three-dimensional interpolant to a set of scattered data points, using a modified Shepard method.

2 Specification

```c
void nag_3d_shep_interp (Integer m, const double x[], const double y[],
const double z[], const double f[], Integer nw, Integer nq, Integer iq[],
double rq[], NagError *fail)
```

3 Description

This function constructs a smooth function $Q(x, y, z)$ which interpolates a set of $m$ scattered data points $(x_r, y_r, z_r, f_r)$, 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

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

where $q_r = f_r$ and $w_r(x, y, z) = \frac{1}{d_r^2}$ and $d_r^2 = (x - x_r)^2 + (y - y_r)^2 + (z - z_r)^2$.

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

The efficiency of the function is further enhanced by using a cell method for nearest neighbour searching due to Bentley and Friedman (1979).

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 by the function, and advice on alternatives is given in Section 8.2.

This function is derived from the function QSHEP3 described by Renka (1988b).

Values of the interpolant $Q(x, y, z)$ generated by this function, and its first partial derivatives, can subsequently be evaluated for points in the domain of the data by a call to nag_3d_shep_eval (e01thc).

4 References


5 Parameters

1: \( m \) – Integer
   \( \text{Input} \)
   \( \text{On entry: } m, \text{ the number of data points.} \)
   \( \text{Constraint: } m \geq 10. \)

2: \( x[m] \) – const double
   \( \text{Input} \)
   \( \text{On entry: } x[r - 1], y[r - 1], z[r - 1] \text{ must be set to the Cartesian coordinates of the data point } (x_r, y_r, z_r), \text{ for } r = 1, 2, \ldots, m. \)
   \( \text{Constraint: these coordinates must be distinct, and must not all be coplanar.} \)

3: \( y[m] \) – const double
   \( \text{Input} \)

4: \( z[m] \) – const double
   \( \text{Input} \)

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

6: \( nw \) – Integer
   \( \text{Input} \)
   \( \text{On entry: the number } N_w \text{ of data points that determines each radius of influence } R_w, \text{ appearing in the definition of each of the weights } w_r, \text{ for } r = 1, 2, \ldots, m \text{ (see Section 3). Note that } R_w \text{ is different for each weight. If } nw \leq 0 \text{ the default value } nw = \min(32, m - 1) \text{ is used instead.} \)
   \( \text{Constraint: } nw \leq \min(40, m - 1). \)

7: \( nq \) – Integer
   \( \text{Input} \)
   \( \text{On entry: the number } N_q \text{ of data points to be used in the least-squares fit for coefficients defining the nodal functions } q_r(x, y, z) \text{ (see Section 3). If } nq \leq 0 \text{ the default value } nq = \min(17, m - 1) \text{ is used instead.} \)
   \( \text{Constraint: } nq \leq 0 \text{ or } 9 \leq nq \leq \min(40, m - 1). \)

8: \( iq[dim] \) – Integer
   \( \text{Output} \)
   \( \text{Note: the dimension, } dim, \text{ of the array } iq \text{ must be at least } 2 \times m + 1. \)
   \( \text{On exit: integer data defining the interpolant } Q(x, y, z). \)

9: \( rq[dim] \) – double
   \( \text{Output} \)
   \( \text{Note: the dimension, } dim, \text{ of the array } rq \text{ must be at least } 10 \times m + 7. \)
   \( \text{On exit: real data defining the interpolant } Q(x, y, z). \)

10: \( fail \) – NagError *
    \( \text{Input/Output} \)
    \( \text{The NAG error parameter (see the Essential Introduction).} \)
6 Error Indicators and Warnings

NE_INT
On entry, \( m = \langle value \rangle \).
Constraint: \( m \geq 10 \).

On entry, \( nq > 0 \) and \( nq < 9 \): \( nq = \langle value \rangle \).

NE_INT_2
On entry, \( nw > \min(40, m - 1) \): \( nw = \langle value \rangle \), \( m = \langle value \rangle \).

On entry, \( nq > \min(40, m - 1) \): \( nq = \langle value \rangle \), \( m = \langle value \rangle \).

NE_DATA_COPLANAR
All nodes are coplanar. There is no unique solution.

NE_DUPLICATE_NODE
There are duplicate nodes in the data set. \( (x[i - 1], y[i - 1], z[i - 1]) = (x[j - 1], y[j - 1], z[j - 1]) \)
for: \( i = \langle value \rangle \) and \( j = \langle value \rangle \). The interpolant cannot be derived.

NE_BAD_PARAM
On entry, parameter \( \langle value \rangle \) had an illegal value.

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 consult NAG for assistance.

7 Accuracy
On successful exit, the function generated interpolates the input data exactly and has quadratic accuracy.

8 Further Comments
8.1 Timing
The time taken for a call to nag_3d_shep_interp (e01tgc) will depend in general on the distribution of the
data points. If \( x, y \) and \( z \) 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 parameters \( N_w \) and \( N_q \) may be selected by calling nag_3d_shep_interp (e01tgc) with
\( nw \leq 0 \) and \( nq \leq 0 \). These default values may well be satisfactory for many applications.

If non-default values are required they must be supplied to nag_3d_shep_interp (e01tgc) through positive
values of \( nw \) and \( nq \). Increasing these parameters makes the method less local. This may increase the
accuracy of the resulting interpolant at the expense of increased computational cost. The default values
\( nw = \min(32, m - 1) \) and \( nq = \min(17, m - 1) \) have been chosen on the basis of experimental results
reported in Renka (1988c). In these experiments the error norm was found to vary smoothly with \( N_w \) and
\( N_q \), generally increasing monotonically and slowly with distance from the optimal pair. The method is not
therefore thought to be particularly sensitive to the parameter values. For further advice on the choice of
these parameters see Renka (1988c).
9 Example

This program reads in a set of 30 data points and calls nag_3d_shep_interp (e01tgc) to construct an interpolating function $Q(x,y,z)$. It then calls nag_3d_shep_eval (e01thc) 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.

9.1 Program Text

/* nag_3d_shep_interp (e01tgc) Example Program. */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */
*
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage01.h>

int main(void)
{
    /* Scalars */
    Integer exit_status, i, m, n, nq, nw, liq, lrq;
    NagError fail;
    /* Arrays */
    double *f = 0, *q = 0, *qx = 0, *qy = 0, *qz = 0, *rq = 0,
        *u = 0, *v = 0, *w = 0, *x = 0, *y = 0, *z = 0;
    Integer *iq = 0;
    exit_status = 0;
    INIT_FAIL(fail);
    Vprintf("e01tgc Example Program Results\n");

    /* Skip heading in data file */
    Vscanf("%*\[^
\] ");

    /* Input the number of nodes. */
    Vscanf("%ld%*[^
\] ", &m);

    if (m > 0)
    {
        /* Allocate memory */
        lrq = 10 * m + 7;
        liq = 2 * m + 1;
        if ( !(f = NAG_ALLOC(m, double)) ||
            !(x = NAG_ALLOC(m, double)) ||
            !(y = NAG_ALLOC(m, double)) ||
            !(z = NAG_ALLOC(m, double)) ||
            !(rq = NAG_ALLOC(lrq, double)) ||
            !(iq = NAG_ALLOC(liq, Integer)) )
        {
            Vprintf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }

        /* Input the data points X,Y,Z and F. */
        for (i = 0; i < m; ++i)
            Vscanf("%lf%lf%lf%*[\n] ", &x[i], &y[i], &z[i], &f[i]);

        /* Generate the interpolant. */
        nq = 0;
        nw = 0;

        e01tgc

        NAG C Library Manual
e01tgc(m, x, y, z, f, nw, nq, iq, rq, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from e01tgc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Input the number of evaluation points. */
Vscanf("%ld%*[^\n] ", &n);

/* Allocate memory for e01thc */
if ( !(q = NAG_ALLOC(n, double)) ||
    !(qx = NAG_ALLOC(n, double)) ||
    !(qy = NAG_ALLOC(n, double)) ||
    !(qz = NAG_ALLOC(n, double)) ||
    !(u = NAG_ALLOC(n, double)) ||
    !(v = NAG_ALLOC(n, double)) ||
    !(w = NAG_ALLOC(n, double)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Input the evaluation points. */
for (i = 0; i < n; ++i)
    Vscanf("%lf%lf%lf%*[^\n] ", &u[i], &v[i], &w[i]);

/* Evaluate the interpolant using e01thc. */
fail.print = TRUE;
e01thc(m, x, y, z, f, iq, rq, n, u, v, w, q, qx, qy, qz, &fail);

Vprintf("\n");
Vprintf(" u(i) v(i) w(i) Q(i)\n");
for (i = 0; i < n; ++i)
    Vprintf("%6ld%10.4f%10.4f%10.4f%10.4f\n", i, u[i], v[i], w[i], q[i]);

END:
if (f) NAG_FREE(f);
if (q) NAG_FREE(q);
if (qx) NAG_FREE(qx);
if (qy) NAG_FREE(qy);
if (qz) NAG_FREE(qz);
if (rq) NAG_FREE(rq);
if (u) NAG_FREE(u);
if (v) NAG_FREE(v);
if (w) NAG_FREE(w);
if (x) NAG_FREE(x);
if (y) NAG_FREE(y);
if (z) NAG_FREE(z);
if (iq) NAG_FREE(iq);

return exit_status;
}

9.2 Program Data

<table>
<thead>
<tr>
<th>M, the number of data points</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
</tr>
<tr>
<td>0.80 0.23 0.37 0.51 X, Y, Z, F data point definition</td>
</tr>
<tr>
<td>0.23 0.88 0.05 1.80</td>
</tr>
<tr>
<td>0.23 0.88 0.05 1.80</td>
</tr>
<tr>
<td>0.18 0.43 0.04 0.11</td>
</tr>
<tr>
<td>0.58 0.95 0.62 2.65</td>
</tr>
<tr>
<td>0.64 0.69 0.20 0.93</td>
</tr>
<tr>
<td>0.88 0.35 0.49 0.72</td>
</tr>
<tr>
<td>0.30 0.10 0.78 -0.11</td>
</tr>
<tr>
<td>0.87 0.09 0.05 0.67</td>
</tr>
</tbody>
</table>
0.04 0.02 0.40 0.00
0.62 0.90 0.43 2.20
0.87 0.96 0.24 3.17
0.62 0.64 0.45 0.74
0.86 0.13 0.47 0.64
0.87 0.60 0.46 1.07
0.49 0.43 0.13 0.22
0.12 0.61 0.00 0.41
0.02 0.71 0.82 0.58
0.62 0.93 0.44 2.48
0.49 0.54 0.04 0.37
0.36 0.56 0.39 0.35
0.62 0.42 0.97 -0.20
0.01 0.72 0.45 0.78
0.41 0.36 0.52 0.11
0.17 0.99 0.65 2.82
0.51 0.29 0.59 0.14
0.85 0.05 0.04 0.61
0.20 0.20 0.87 -0.25
0.04 0.67 0.04 0.59
0.31 0.63 0.18 0.50
0.88 0.27 0.07 0.71 End of data points
<table>
<thead>
<tr>
<th>N, the number of evaluation points</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10 0.10 0.10 U, V, W evaluation point definition</td>
</tr>
<tr>
<td>0.20 0.20 0.20</td>
</tr>
<tr>
<td>0.30 0.30 0.30</td>
</tr>
<tr>
<td>0.40 0.40 0.40</td>
</tr>
<tr>
<td>0.50 0.50 0.50</td>
</tr>
<tr>
<td>0.60 0.60 0.60 End of evaluation points</td>
</tr>
</tbody>
</table>

### 9.3 Program Results

e01tgc Example Program Results

<table>
<thead>
<tr>
<th>i</th>
<th>u(i)</th>
<th>v(i)</th>
<th>w(i)</th>
<th>Q(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.1000</td>
<td>0.1000</td>
<td>0.1000</td>
<td>0.2630</td>
</tr>
<tr>
<td>1</td>
<td>0.2000</td>
<td>0.2000</td>
<td>0.2000</td>
<td>0.1182</td>
</tr>
<tr>
<td>2</td>
<td>0.3000</td>
<td>0.3000</td>
<td>0.3000</td>
<td>0.0811</td>
</tr>
<tr>
<td>3</td>
<td>0.4000</td>
<td>0.4000</td>
<td>0.4000</td>
<td>0.1552</td>
</tr>
<tr>
<td>4</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.3019</td>
</tr>
<tr>
<td>5</td>
<td>0.6000</td>
<td>0.6000</td>
<td>0.6000</td>
<td>0.5712</td>
</tr>
</tbody>
</table>