nag_2d_scat_interpolant (e01sac)

1. Purpose

The function `nag_2d_scat_interpolant (e01sac)` generates a two-dimensional surface interpolating a set of scattered data points, using either the method of Renka and Cline or a modification of Shepard’s method.

2. Specification

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

void nag_2d_scat_interpolant(Nag_2d_Scat_Method method, Integer m,
                            double x[], double y[], double f[], Nag_Scat_Struct *comm,
                            Nag_E01_Opt *optional, NagError *fail)
```

3. Description

This function constructs an interpolating surface $F(x, y)$ through a set of $m$ scattered data points $(x_r, y_r, f_r)$, for $r = 1, 2, \ldots, m$, using either a method due to Renka and Cline or a modification of Shepard’s method. In the $(x, y)$ plane, the data points must be distinct. The constructed surface is continuous and has continuous first derivatives.

If `method` = `Nag_RC`, the method due to Renka and Cline is used. This involves firstly creating a triangulation with all the $(x, y)$ data points as nodes, the triangulation being as nearly equiangular as possible (see Cline and Renka (1984)). Then gradients in the $x$- and $y$-directions are estimated at node $r$, for $r = 1, 2, \ldots, m$, as the partial derivatives of a quadratic function of $x$ and $y$ which interpolates the data value $f_r$, and which fits the data values at nearby nodes (those within a certain distance chosen by the algorithm) in a weighted least-squares sense. The weights are chosen such that closer nodes have more influence than more distant nodes on derivative estimates at node $r$. The computed partial derivatives, with the $f_r$ values, at the three nodes of each triangle define a piecewise polynomial surface of a certain form which is the interpolant on that triangle. See Renka and Cline (1984) for more detailed information on the algorithm, a development of that by Lawson (1977). The code is derived from Renka (1984).

The interpolant $F(x, y)$ can subsequently be evaluated at $n$ points $(x_k, y_k)$, for $k = 1, 2, \ldots, n$, inside or outside the domain of the data by a call to `nag_2d_scat_eval (e01sbc)`. Points outside the domain are evaluated by extrapolation.

If `method` = `Nag_Shep`, then a modification of Shepard’s method is used. The basic Shepard method, described in Shepard (1968), interpolates the input data with the weighted mean

$$F(x, y) = \frac{\sum_{r=1}^{m} w_r(x, y)f_r}{\sum_{r=1}^{m} w_r(x, y)},$$

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

The basic method is global in that the interpolated value at any point depends on all the data, but this routine uses a modification due to Franke and Nielson (1980), whereby the method becomes local by adjusting each $w_r(x, y)$ to be zero outside a circle with centre $(x_r, y_r)$ and some radius $R_w$. Also, to improve the performance of the basic method, each $f_r$ above is replaced by a function $f_r(x, y)$, which is a quadratic fitted by weighted least-squares to data local to $(x_r, y_r)$ and forced to interpolate $(x_r, y_r, f_r)$. In this context, a point $(x, y)$ 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 part of the routine. If there are less than 5 other points within distance $R_q$ from $(x_r, y_r)$, the quadratic is replaced by a linear function. In cases of rank-deficiency, the minimum norm solution is computed.
The user may specify values for $R_w$ and $R_q$, but it is usually easier to choose instead two integers $N_w$ and $N_q$, from which the routine will compute $R_w$ and $R_q$. These integers can be thought of as the average numbers of data points lying within distances $R_w$ and $R_q$ respectively from each node. Default values are provided, and advice on alternatives is given in Section 6.

The interpolant $F(x, y)$ generated by this routine can subsequently be evaluated for $n$ points $(x_k, y_k)$, for $k = 1, 2, \ldots, n$, in the domain of the data by a call to nag_2d_scat_eval (e01sbc).

4. Parameters

**method**

Input: indicates the method to be used in interpolating the surface.
If method = Nag_RC, the method due to Renka and Cline is used.
If method = Nag_Shep, then a modification of Shepard’s method is used.
Constraint: method = Nag_RC or Nag_Shep.

**m**

Input: the number of data points, $m$.
Constraint: $m \geq 3$.

**x[m]**

Input: the co-ordinates of the $r$th data point, for $r = 1, 2, \ldots, m$. The data points are accepted in any order, but see Section 6.
Constraint: Each node must be unique. Additionally for Nag_RC method the $(x, y)$ nodes must not be all collinear.

**comm**

Pointer to a communication structure of type Nag_Scat_Struct. For method = Nag_RC, this structure contains the computed triangulation and the estimated partial derivatives at the nodes. For method = Nag_Shep, this structure contains the coefficients of the constructed nodal functions.

This structure must be passed unchanged to the interpolant evaluating function nag_2d_scat_eval (e01sbc).

**optional**

Pointer to structure of type Nag_E01_Opt, which may be used only if method = Nag_Shep.
It has the following members:

**rnw – double**

Input: suitable values for the radii $R_w$ and $R_q$, described in Section 3.
Constraint: $0 < \text{optional.rnw} \leq \text{optional.rnq}$.
If this constraint is satisfied then these values are used, otherwise if rnq $\leq 0$ then suitable values of $R_w$ and $R_q$ are computed from other members of this structure, nw and nq.
Output: optional.rnw and optional.rnq contain the actual values of $R_w$ and $R_q$ used by nag_2d_scat_interpolant.

**nw – Integer**

Input: these are integer values which can be used to compute the radii $R_w$ and $R_q$, if the user does not wish to supply appropriate values directly, i.e., optional.rnq $\leq 0$.
Constraint: $0 < \text{optional.nw} \leq \text{optional.nq}$.
If this constraint is satisfied then nag_2d_scat_interpolant computes the appropriate values of the radii $R_w$ and $R_q$. Otherwise, if optional.nq $\leq 0$, then optional.nw and optional.nq are set to 9 and 18 respectively and these values are used to compute the radii $R_w$ and $R_q$.
Output: If the values of optional.rnw and optional.rnq as supplied are used for the radii $R_w$ and $R_q$ then optional.nw and optional.nq are set to zero, otherwise optional.nw and optional.nq contain the actual values used for computing the radii $R_w$ and $R_q$.
If this structure is supplied as a null pointer (Nag_E01_Opt)*0, also defined by Nag as E01_DEFAULT when method = Nag_Shep, then the default values of \( n_w = 9 \) and \( n_q = 18 \) are used to compute the radii.

\textbf{minq} – Integer
Output: the minimum number of data points that lie within radius \texttt{optional.rnw} of any node, and thus define a nodal function. If \texttt{optional.minq} is very small (say, less than 5), then the interpolant may be unsatisfactory in regions where the data points are sparse. \texttt{nag_2d_scat_interpolant} generates a warning in this case. Note that this structure is not referenced when method = Nag_RC. It should however be specified as E01_DEFAULT.

\textbf{fail}
The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. \textbf{Error Indications and Warnings}

\textbf{NE_BAD_PARAM}
On entry, parameter method had an illegal value.

\textbf{NE_INT_ARG_LT}
On entry, \texttt{m} must not be less than 3: \texttt{m} = \langle \text{value} \rangle.

\textbf{NE_ALLOC_FAIL}
Memory allocation failed.

\textbf{NE_ALL_DATA_COLLINEAR}
On entry, all the (x,y) pairs are collinear. Consider specifying method = Nag_Shep or using a one-dimensional interpolating function nag_1d_spline_interpolant (e01bac).

\textbf{NE_DATA_NOT_UNIQUE}
On entry, each data pair is not unique since data points \((x[\langle \text{value} \rangle],y[\langle \text{value} \rangle])\) and \((x[\langle \text{value} \rangle],y[\langle \text{value} \rangle])\) are identical and equal to \((\langle \text{value} \rangle,\langle \text{value} \rangle)\).

\textbf{NE_NUM_PARAM_INVALID}
On entry, either or both of \texttt{optional.nq} and \texttt{optional.nw} are invalid, \texttt{optional.nq} = \langle \text{value} \rangle and \texttt{optional.nw} = \langle \text{value} \rangle. \texttt{optional.nq} and \texttt{optional.nw} must satisfy the following constraints: \(0 < \texttt{optional.nw} \leq \texttt{optional.nq}\).

\textbf{NE_RAD_PARAM_INVALID}
On entry, either or both of \texttt{optional.rnq} and \texttt{optional.rnw} are invalid, \texttt{optional.rnq} = \langle \text{value} \rangle and \texttt{optional.rnw} = \langle \text{value} \rangle. \texttt{optional.rnq} and \texttt{optional.rnw} must satisfy the following constraints: \(0.0 < \texttt{optional.rnw} \leq \texttt{optional.rnq}\).

\textbf{NW_SPARSE_DATA_FIT}
The minimum number of data points \langle \text{value} \rangle that lie within the radius \texttt{optional.rnq} of any node is small enough to indicate that the interpolant may be unsatisfactory in regions where the data points are sparse. Current values of other relevant parameters (available as members of the structure \texttt{optional}, if this has been defined) are \texttt{rnq} = \langle \text{value} \rangle, \texttt{rnw} = \langle \text{value} \rangle, \texttt{nq} = \langle \text{value} \rangle, \texttt{nw} = \langle \text{value} \rangle.

6. \textbf{Further Comments}
If method = Nag_RC, the time taken for a call of \texttt{nag_2d_scat_interpolant} is approximately proportional to the number of data points, \( m \). The routine is more efficient if, before entry, the values in \( x, y, f \) are arranged so that the \( x \) array is in ascending order.

If method = Nag_Shep, the time taken for a call of \texttt{nag_2d_scat_interpolant} is also approximately proportional to the number of data points, \( m \), provided that \( N_q \) is of the same order as its default value (18). However, if \( N_q \) is increased so that the method becomes more global, the time taken becomes approximately proportional to \( m^2 \).

In particular, when method = Nag_Shep, note first that the radii \( R_w \) and \( R_q \), described in Section 3, are computed as \( D \sqrt{\frac{N_q}{m}} \) and \( D \sqrt{\frac{N_q}{m}} \) respectively, where \( D \) is the maximum distance between any pair of data points.
Default values $N_w = 9$ and $N_q = 18$ work quite well when the data points are fairly uniformly distributed. However, for data having some regions with relatively few points or for small data sets ($m < 25$), a larger value of $N_w$ may be needed. This is to ensure a reasonable number of data points within a distance $R_w$ of each node, and to avoid some regions in the data area being left outside all the discs of radius $R_w$ on which the weights $w_r(x, y)$ are non-zero. Maintaining $N_q$ approximately equal to $2N_w$ is usually an advantage.

Note however that increasing $N_w$ and $N_q$ does not improve the quality of the interpolant in all cases. It does increase the computational cost and makes the method less local.

At the end of the program, especially before any repeated call to nag_2d_scat_interpolant, the function nag_2d_scat_free (e01szc) must be called with the communication structure comm as the argument in order to free the memory previously allocated to the pointer members of comm.

6.1. Accuracy

On successful exit, the computational errors should be negligible in most situations but the user should always check the computed surface for acceptability, by drawing contours for instance. The surface always interpolates the input data exactly.

6.2. References


7. See Also

nag_2d_scat_eval (e01sbc)
nag_2d_scat_free (e01szc)

8. Example

This program reads in a set of 30 data points and calls nag_2d_scat_interpolant to construct an interpolating surface. It then calls nag_2d_scat_eval (e01sbc) to evaluate the interpolant at a sample of points on a rectangular grid. The two methods described in Section 3 are used in the construction of the interpolating surface and the subsequent evaluation of the interpolant.

Note that this example is not typical of a realistic problem: the number of data points would normally be larger, and the interpolant would need to be evaluated on a finer grid to obtain an accurate plot, say.

8.1. Program Text

/* nag_2d_scat_interpolant(e01sac) Example Program *
 * *
 * Mark 4, 1996.
 */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nag_e01.h>

#define MMAX 100
#define NMAX 10
#define NXYM NMAX*NMAX
#define TDX MMAX
#define TDQ MMAX+1

main()
{
    Integer i, isel, j, m, n, nx, ny;
    double xhi, xlo, yhi, ylo;
    double f[MMAX], pf[NXYM], px[NXYM], py[NXYM], x[MMAX], y[MMAX];
    Nag_Scat_Struct comm;
    Nag_2d_Scat_Method method;
    Nag_E01_Opt optional;

    Vprintf("e01sac Example Program Results\n");
    /* Skip heading in data file */
    Vscanf("%*[^
");
    /* Input the number of nodes. */
    Vscanf("%ld", &m);
    if (m>=1 && m<=MMAX)
    {
        /* Input the nodes (x,y) and heights, f. */
        for (i=0; i<m; ++i)
            Vscanf("%lf%lf%lf", &x[i], &y[i], &f[i]);

        /* Input the domain for evaluating the interpolant. */
        Vscanf("%ld%lf%lf", &nx, &xlo, &xhi);
        Vscanf("%ld%lf%lf", &ny, &ylo, &yhi);
        for (isel=1; isel<=2; ++isel)
        {
            /* Select the method of interpolation. */
            if (isel==1)
                method = Nag_RC;
            else if (isel==2)
                method = Nag_Shep;

            if (method==Nag_RC)
            {
                Vprintf("\nExample 1: Surface interpolation by method\nof Renka and Cline.\n\n");
                /* Generate the triangulation and gradients using the selected
                * method. */
                e01sac(method, m, x, y, f, &comm, (Nag_E01_Opt *)0, NAGERR_DEFAULT);
            }
            else if (method==Nag_Shep)
            {
                Vprintf("\nExample 2: Surface interpolation by modified\nShepard's method.\n\n");
                /* Compute the nodal function coefficients. */
                optional.nq = 24;
                optional.nw = 12;
                optional.rnq = -1.0;
                e01sac(method, m, x, y, f, &comm, &optional, NAGERR_DEFAULT);
                Vprintf(" optional.rnw =%8.2f optional.rnq =%8.2f\n",
                        optional.rnw, optional.rnq);
                Vprintf(" minimum number of data points that lie within radius\noptional.rnq =%3ld\n", optional.minnq);
            }
            /* Evaluate the interpolant on a rectangular grid at nx*ny points
            * over the domain (xlo to xhi) x (ylo to yhi).
nag_2d_scat_interpolant

NAG C Library Manual

*/

n = 0;
if (nx*ny<=NXYM)
{
  for (j=0; j<ny; ++j)
  {
    for (i=0; i<nx; ++i)
    {
      px[i+nx*j] = ((double)(nx-i-1) / (nx-1)) * xlo +
      ((double)(i) / (nx-1)) * xhi;
      py[i+nx*j] = ((double)(ny-j-1) / (ny-1)) * ylo +
      ((double)(j) / (ny-1)) * yhi;
      ++n;
    }
  }
  if (method == Nag_RC)
    e01sbc(&comm, n, px, py, pf, NAGERR_DEFAULT);
  else if (method == Nag_Shep)
    e01sbc(&comm, n, px, py, pf, NAGERR_DEFAULT);
  Vprintf("\n x\n");
  for (i = 0; i < nx; i++)
    Vprintf("%8.2f", px[i]);
  Vprintf("\n y\n");
  for (i = ny-1; i >= 0; --i)
  {
    Vprintf("%8.2f ", py[nx * i]);
    for (j = 0; j < nx; j++)
      Vprintf("%8.2f", pf[nx * i + j]);
    Vprintf("\n");
  }
  /* Free the memory allocated to the pointers in structure comm. */
  e01szc(&comm);
} else
{
  Vfprintf(stderr, "m is out of range: m = %-3ld\n", m);
  exit(EXIT_FAILURE);
}
} /* main */

8.2. Program Data

e01sac Example Program Data

30
11.16  1.24  22.15
12.85  3.06  22.11
19.85  10.72  7.97
19.72  1.39  16.83
15.91  7.74  15.30
0.00  20.00  34.60
20.87  20.00  5.74
3.45  12.78  41.24
14.26  17.87  10.74
17.43  3.46  18.60
22.80  12.39  5.47
7.58  1.98  29.87
25.00  11.87  4.40
0.00  0.00  58.20
9.66  20.00  4.73
5.22  14.66  40.36
17.25  19.57  6.43
25.00  3.87  8.74
12.13  10.79  13.71
22.23  6.21  10.25
11.52  8.53  15.74
15.20  0.00  21.60
7.54  10.69  19.31
8.3. Program Results

e01sac Example Program Results

Example 1: Surface interpolation by method of Renka and Cline.

<table>
<thead>
<tr>
<th>x</th>
<th>3.00</th>
<th>6.00</th>
<th>9.00</th>
<th>12.00</th>
<th>15.00</th>
<th>18.00</th>
<th>21.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>17.00</td>
<td>41.25</td>
<td>27.62</td>
<td>18.03</td>
<td>12.29</td>
<td>11.68</td>
<td>9.09</td>
</tr>
<tr>
<td>y</td>
<td>14.00</td>
<td>36.66</td>
<td>22.87</td>
<td>14.02</td>
<td>13.44</td>
<td>11.20</td>
<td>6.46</td>
</tr>
<tr>
<td>y</td>
<td>11.00</td>
<td>38.55</td>
<td>25.25</td>
<td>16.72</td>
<td>13.83</td>
<td>13.08</td>
<td>10.71</td>
</tr>
<tr>
<td>y</td>
<td>8.00</td>
<td>37.90</td>
<td>23.97</td>
<td>16.79</td>
<td>16.43</td>
<td>15.46</td>
<td>13.02</td>
</tr>
<tr>
<td>y</td>
<td>5.00</td>
<td>40.49</td>
<td>29.26</td>
<td>22.51</td>
<td>20.72</td>
<td>19.30</td>
<td>16.72</td>
</tr>
<tr>
<td>y</td>
<td>2.00</td>
<td>43.52</td>
<td>33.91</td>
<td>26.59</td>
<td>22.23</td>
<td>21.15</td>
<td>18.67</td>
</tr>
</tbody>
</table>

Example 2: Surface interpolation by modified Shepard’s method.

optional.rnw = 9.49 optional.rnq = 13.42

minimum number of data points that lie within radius optional.rnq = 7

<table>
<thead>
<tr>
<th>x</th>
<th>3.00</th>
<th>6.00</th>
<th>9.00</th>
<th>12.00</th>
<th>15.00</th>
<th>18.00</th>
<th>21.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>17.00</td>
<td>40.23</td>
<td>27.72</td>
<td>21.23</td>
<td>14.59</td>
<td>12.00</td>
<td>9.43</td>
</tr>
<tr>
<td>y</td>
<td>14.00</td>
<td>46.96</td>
<td>37.37</td>
<td>23.74</td>
<td>14.67</td>
<td>13.25</td>
<td>11.29</td>
</tr>
<tr>
<td>y</td>
<td>11.00</td>
<td>39.42</td>
<td>25.42</td>
<td>16.32</td>
<td>13.78</td>
<td>12.60</td>
<td>10.39</td>
</tr>
<tr>
<td>y</td>
<td>8.00</td>
<td>37.50</td>
<td>22.36</td>
<td>18.57</td>
<td>15.63</td>
<td>15.55</td>
<td>13.05</td>
</tr>
<tr>
<td>y</td>
<td>5.00</td>
<td>41.25</td>
<td>31.76</td>
<td>24.74</td>
<td>21.17</td>
<td>18.93</td>
<td>16.83</td>
</tr>
<tr>
<td>y</td>
<td>2.00</td>
<td>44.58</td>
<td>34.35</td>
<td>26.47</td>
<td>22.27</td>
<td>20.98</td>
<td>18.69</td>
</tr>
</tbody>
</table>