NAG Toolbox

nag_interp_2d_scat (e01sa)

1 Purpose

nag_interp_2d_scat (e01sa) generates a two-dimensional surface interpolating a set of scattered data points, using the method of Renka and Cline.

2 Syntax

```
[triang, grads, ifail] = nag_interp_2d_scat(x, y, f, 'm', m)
[triang, grads, ifail] = e01sa(x, y, f, 'm', m)
```

3 Description

nag_interp_2d_scat (e01sa) 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 a method due to Renka and Cline. In the (x, y) plane, the data points must be distinct. The constructed surface is continuous and has continuous first derivatives.

The method 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 any point (x, y) inside or outside the domain of the data by a call to nag_interp_2d_scat_eval (e01sb). Points outside the domain are evaluated by extrapolation.

4 References

Cline A K and Renka R L (1984) A storage-efficient method for construction of a Thiessen triangulation *Rocky Mountain J. Math.* **14** 119–139

Lawson C L (1977) Software for C^1 surface interpolation *Mathematical Software III* (ed J R Rice) 161–194 Academic Press

Renka R L (1984) Algorithm 624: triangulation and interpolation of arbitrarily distributed points in the plane *ACM Trans. Math. Software* **10** 440–442

Renka R L and Cline A K (1984) A triangle-based C^1 interpolation method *Rocky Mountain J. Math.* **14** 223–237

5 Parameters

5.1 Compulsory Input Parameters

1: **x**(**m**) – REAL (KIND=nag_wp) array
2: **y**(**m**) – REAL (KIND=nag_wp) array
3: **f**(**m**) – REAL (KIND=nag_wp) array

The coordinates of the rth data point, for $r = 1, 2, \ldots, m$. The data points are accepted in any order, but see Section 9.

Constraint: the (x, y) nodes must not all be collinear, and each node must be unique.

5.2 Optional Input Parameters

1: **m** – INTEGER

Default: the dimension of the arrays **x**, **y**, **f**. (An error is raised if these dimensions are not equal.)

m, the number of data points.

Constraint: **m** ≥ 3.

5.3 Output Parameters

1: **triang**($7 \times$ **m**) – INTEGER array

A data structure defining the computed triangulation, in a form suitable for passing to nag_interp_2d_scat_eval (e01sb).

2: **grads**(**2**, **m**) – REAL (KIND=nag_wp) array

The estimated partial derivatives at the nodes, in a form suitable for passing to nag_interp_2d_scat_eval (e01sb). The derivatives at node r with respect to x and y are contained in **grads**$(1, r)$ and **grads**$(2, r)$ respectively, for $r = 1, 2, \ldots, m$.

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 $= 1$

On entry, **m** < 3.

ifail $= 2$

On entry, all the **(x,y)** pairs are collinear.

ifail $= 3$

On entry, $(\mathbf{x}(i), \mathbf{y}(i)) = (\mathbf{x}(j), \mathbf{y}(j))$ for some $i \neq j$.

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

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

8 Further Comments

The time taken for a call of nag_interp_2d_scat (e01sa) is approximately proportional to the number of data points, m. The function is more efficient if, before entry, the values in **x**, **y** and **f** are arranged so that the **x** array is in ascending order.

9 Example

This example reads in a set of 30 data points and calls nag_interp_2d_scat (e01sa) to construct an interpolating surface. It then calls nag_interp_2d_scat_eval (e01sb) to evaluate the interpolant at a sample of points on a rectangular grid.

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.

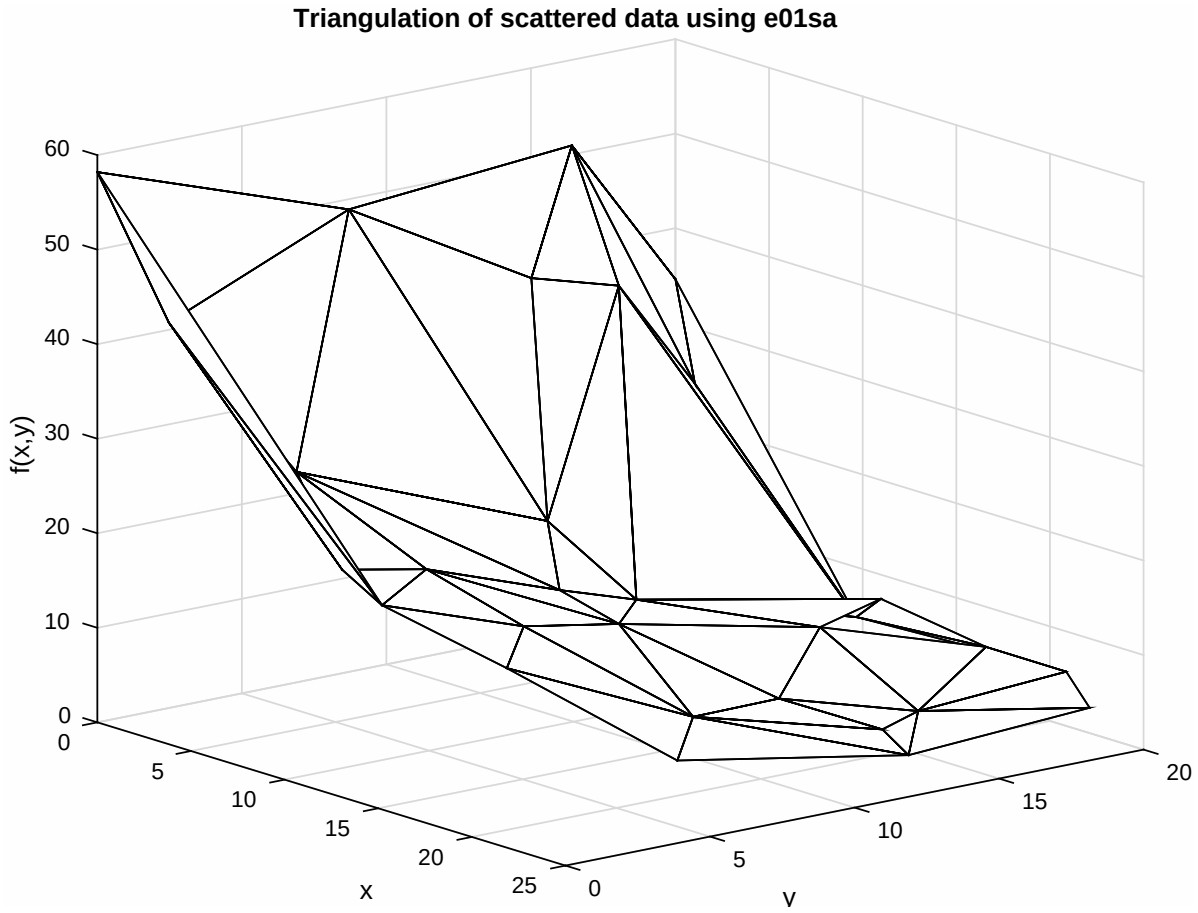
9.1 Program Text

```
function e01sa_example

fprintf('e01sa example results\n\n');

% Scattered Grid Data
x = [11.16; 12.85; 19.85; 19.72; 15.91;  0.00; 20.87;  3.45; 14.26; ...
     17.43; 22.80;  7.58; 25.00;  0.00;  9.66;  5.22; 17.25; 25.00; ...
     12.13; 22.23; 11.52; 15.20;  7.54; 17.32;  2.14;  0.51; 22.69; ...
      5.47; 21.67;  3.31];
y = [ 1.24;  3.06; 10.72;  1.39;  7.74; 20.00; 20.00; 12.78; 17.87; ...
      3.46; 12.39;  1.98; 11.87;  0.00; 20.00; 14.66; 19.57;  3.87; ...
     10.79;  6.21;  8.53;  0.00; 10.69; 13.78; 15.03;  8.37; 19.63; ...
     17.13; 14.36; 0.33];
f = [22.15; 22.11;  7.97; 16.83; 15.30; 34.60;  5.74; 41.24; 10.74; ...
     18.60;  5.47; 29.87;  4.40; 58.20;  4.73; 40.36;  6.43;  8.74; ...
     13.71; 10.25; 15.74; 21.60; 19.31; 12.11; 53.10; 49.43;  3.25; ...
     28.63;  5.52; 44.08];
% Triangulate and obtain details of interpolant
[triang,grads,ifail] = e01sa(x,y,f);

% Convert to for for trimesh plotter.
n       = size(x,1);
[tri,k] = triang2tri(triang,n);
fprintf('Number of triangles in triangulation = %d\n\n',k);

% Display mesh
fig1 = figure;
trimesh(tri(1:k,1:3),x,y,f);
xlabel('x');
ylabel('y');
zlabel('f(x,y)')
title('Triangulation of scattered data using e01sa');
view(51,18);

% Evaluate interpolant at on regular mesh (px,py)
px = [3:3:21];
py = [2:3:17];
for i = 1:6
  for j = 1:7
```

```
    [pf(i,j), ifail] = e01sb( ...
                               x, y, f, triang, grads, px(j), py(i));
  end
end

% Display interpolated values
matrix = 'General';
diag = 'Non-unit';
format = 'F7.2';
mtitl  = 'Spline evaluated on a regular mesh (x across, y down):';
chlab  = 'Character';
rlabs  = cellstr(num2str(py'));
clabs  = cellstr(num2str(px'));
ncols  = nag_int(80);
indent = nag_int(0);
[ifail] =  x04cb( ...
                  matrix, diag, pf, format, mtitl, chlab, ...
                  rlabs, chlab, clabs, ncols, indent);

function [tri,k] = triang2tri(triang,n)
%  Convert triang array to form that MATLAB trimesh function can use

max_t = nag_int(2*n-5);
tri   = nag_int(zeros(max_t,3));
iend  = nag_int(0);
k     = iend;
t     = nag_int([0;0;0]);

for i = 1:n
   % set up loop of nodes connected to node i
   ibeg = iend + 1;
   iend = triang(6*n+i);
   % first connection setup
   t(1) = i;
   t(2) = triang(ibeg);
   % loop over remaining connections
   ibeg = ibeg + 1;
   for j = ibeg:iend
      t(3) = triang(j);
      if t(3)>0
        if t(2)>i || t(3)>i
        % new triangle here
          k = k + 1;
          tri(k,1:3) = t(1:3);
        end
        % shuffle down for next connected node
        t(2) = t(3);
      end
   end
end
```

9.2 Program Results

```
     e01sa example results

Number of triangles in triangulation = 88

 Spline evaluated on a regular mesh (x across, y down):
          3      6      9     12     15     18     21
  2   43.52  33.91  26.59  22.23  21.15  18.67  14.88
  5   40.49  29.26  22.51  20.72  19.30  16.72  12.87
  8   37.90  23.97  16.79  16.43  15.46  13.02   9.30
 11   38.55  25.25  16.72  13.83  13.08  10.71   6.88
 14   47.61  36.66  22.87  14.02  13.44  11.20   6.46
 17   41.25  27.62  18.03  12.29  11.68   9.09   5.37
```


Triangulation of scattered data using e01sa
f(x,y)
60
50
40
30
20
10
0
0
5
10
15
20
25
x
0
5
10
15
20
y