NAG Toolbox

nag_interp_2d_triangulate (e01ea)

1 Purpose

nag_interp_2d_triangulate (e01ea) generates a triangulation for a given set of two-dimensional points using the method of Renka and Cline.

2 Syntax

```
[triang, ifail] = nag_interp_2d_triangulate(x, y, 'n', n)
[triang, ifail] = e01ea(x, y, 'n', n)
```

3 Description

nag_interp_2d_triangulate (e01ea) creates a Thiessen triangulation with a given set of two-dimensional data points as nodes. This triangulation will be as equiangular as possible (Cline and Renka (1984)). 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 computed triangulation is returned in a form suitable for passing to nag_interp_2d_triang_bary_eval (e01eb) which, for a set of nodal function values, computes interpolated values at a set of points.

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**(**n**) – REAL (KIND=nag_wp) array

The x coordinates of the n data points.

2: **y**(**n**) – REAL (KIND=nag_wp) array

The y coordinates of the n data points.

5.2 Optional Input Parameters

1: **n** – INTEGER

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

n, the number of data points.

Constraint: $\mathbf{n} \geq 3$.

5.3 Output Parameters

1: **triang**$(7 \times \mathbf{n})$ – INTEGER array

A data structure defining the computed triangulation, in a form suitable for passing to nag_interp_2d_triang_bary_eval (e01eb). Details of how the triangulation is encoded in **triang** are given in Section 9. These details are most likely to be of use when plotting the computed triangulation which is demonstrated in Section 10.

2: **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 $= 1$

Constraint: **n** ≥ 3.

ifail $= 2$

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

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

Not applicable.

8 Further Comments

The time taken for a call of nag_interp_2d_triangulate (e01ea) is approximately proportional to the number of data points, n. The function is more efficient if, before entry, the (x, y) pairs are arranged in **x** and **y** such that the x values are in ascending order.

The triangulation is encoded in **triang** as follows:

set $j_0 = 0$; for each node, $k = 1, 2, \ldots, n$, (using the ordering inferred from **x** and **y**)

$i_k = j_{k-1} + 1$

$j_k =$ **triang**$(6 \times \mathbf{n} + k)$

triang(j), for $j = i_k, \ldots, j_k$, contains the list of nodes to which node k is connected. If **triang**$(j_k) = 0$ then node k is on the boundary of the mesh.

9 Example

In this example, nag_interp_2d_triangulate (e01ea) creates a triangulation from a set of data points. nag_interp_2d_triang_bary_eval (e01eb) then evaluates the interpolant at a sample of points using this triangulation. Note that this example is not typical of a realistic problem: the number of data points

would normally be larger, so that interpolants can be more accurately evaluated at the fine triangulated grid.

9.1 Program Text

```
     function e01ea_example

fprintf('e01ea 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; 15.2;   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];
% Triangulate on grid
[triang] = e01ea(x, y);

% Convert to form that MATLAB trimesh function can use
n = size(x,1);
t = nag_int([0;0;0]);
max_t = nag_int(2*n-5);
tri = nag_int(zeros(max_t,3));
iend = nag_int(0);
k = nag_int(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

fprintf('Number of triangles in triangulation = %d\n',k);

%display mesh
fig1 = figure;
trimesh(tri(1:k,1:3),x,y)
xlabel('x');
ylabel('y');
title('Triangulation of scattered data using e01ea');

% Interpolate function values on mesh
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];
px = [2.0500 3.7500 5.0000 8.5400 9.1400];
py = [1.7750 3.2500 5.0000 2.0500 4.4500];
```

```
[pf, ifail] = e01eb(...
                    x, y, f, triang, px, py);

fprintf('\n      px        py     Interpolated Value\n');
disp([px' py' pf]);
```

9.2 Program Results

```
     e01ea example results

Number of triangles in triangulation = 88

      px        py     Interpolated Value
    2.0500    1.7750   48.2100
    3.7500    3.2500   41.4195
    5.0000    5.0000   36.1613
    8.5400    2.0500   28.2458
    9.1400    4.4500   24.4543
```