NAG Toolbox for MATLAB

e01sa

1 Purpose

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] = e01sa(x, y, f, 'm', m)
```

3 Description

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 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) – double array**
2: **y(m) – double array**
3: **f(m) – double array**

The co-ordinates of the rth data point, for $r = 1,2,\ldots,m$. The data points are accepted in any order, but see Section 8.

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

5.2 Optional Input Parameters

1: **m – int32 scalar**

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

m, the number of data points.

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

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **triang(7 × m) – int32 array**

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

2: **grads(2,m) – double array**

3: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, $\mathbf{m} < 3$.

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 $i \neq j$.

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 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

```
x = [11.16;
     12.85;
     19.85;
```

```
      19.72;
      15.91;
      0;
      20.87;
      3.45;
      14.26;
      17.43;
      22.8;
      7.58;
      25;
      0;
      9.66;
      5.22;
      17.25;
      25;
      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;
      20;
      12.78;
      17.87;
      3.46;
      12.39;
      1.98;
      11.87;
      0;
      20;
      14.66;
      19.57;
      3.87;
      10.79;
      6.21;
      8.529999999999999;
      0;
      10.69;
      13.78;
      15.03;
      8.369999999999999;
      19.63;
      17.13;
      14.36;
      0.33];
f = [22.15;
      22.11;
      7.97;
      16.83;
      15.3;
      34.6;
      5.74;
      41.24;
      10.74;
      18.6;
      5.47;
      29.87;
      4.4;
```

```
     58.2;
     4.73;
     40.36;
     6.43;
     8.74;
     13.71;
     10.25;
     15.74;
     21.6;
     19.31;
     12.11;
     53.1;
     49.43;
     3.25;
     28.63;
     5.52;
     44.08];
[triang, grads, ifail] = e01sa(x, y, f)
```

```
triang =
     array elided
grads =
  Columns 1 through 7
   -1.1404   -0.4357   -1.1617   -1.2770   -0.6278   -3.1328   -0.6231
    1.1024   -0.4120   -0.1997   -0.4772   -1.2760   -4.8673    0.4750
  Columns 8 through 14
   -4.1162   -0.1496   -1.0252   -0.5330   -2.4193   -0.5942   -4.3053
    4.0498   -1.4064   -0.6116   -0.4112   -1.5365   -0.5379   -0.7194
  Columns 15 through 21
   -1.2108   -4.2773   -0.2324   -1.2172   -0.7484   -1.1258   -0.1296
   -4.7026   -0.4552   -1.3563   -0.6099    0.0376   -0.9677   -1.3922
  Columns 22 through 28
   -0.4934   -2.9679   -1.1591   -4.9834   -4.4156   -1.7010   -3.8652
   -0.4380    1.8909   -0.0578   -0.5508    0.1561   -0.2134   -6.0070
  Columns 29 through 30
   -1.1144   -3.5581
   -0.3749   -0.9580
ifail =
           0
```