NAG Toolbox

nag_interp_2d_scat_eval (e01sb)

1 Purpose

nag_interp_2d_scat_eval (e01sb) evaluates at a given point the two-dimensional interpolant function computed by nag_interp_2d_scat (e01sa).

2 Syntax

```
[pf, ifail] = nag_interp_2d_scat_eval(x, y, f, triang, grads, px, py, 'm', m)
[pf, ifail] = e01sb(x, y, f, triang, grads, px, py, 'm', m)
```

3 Description

nag_interp_2d_scat_eval (e01sb) takes as input the arguments defining the interpolant $F(x,y)$ of a set of scattered data points (x_r, y_r, f_r), for $r = 1, 2, \ldots, m$, as computed by nag_interp_2d_scat (e01sa), and evaluates the interpolant at the point (px, py).

If (px, py) is equal to (x_r, y_r) for some value of r, the returned value will be equal to f_r.

If (px, py) is not equal to (x_r, y_r) for any r, the derivatives in **grads** will be used to compute the interpolant. A triangle is sought which contains the point (px, py), and the vertices of the triangle along with the partial derivatives and f_r values at the vertices are used to compute the value $F(px, py)$. If the point (px, py) lies outside the triangulation defined by the input arguments, the returned value is obtained by extrapolation. In this case, the interpolating function **f** is extended linearly beyond the triangulation boundary. The method is described in more detail in Renka and Cline (1984) and the code is derived from Renka (1984).

nag_interp_2d_scat_eval (e01sb) must only be called after a call to nag_interp_2d_scat (e01sa).

4 References

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
4: **triang**($\mathbf{7} \times \mathbf{m}$) – INTEGER array
5: **grads**($\mathbf{2}, \mathbf{m}$) – REAL (KIND=nag_wp) array

m, **x**, **y**, **f**, **triang** and **grads** must be unchanged from the previous call of nag_interp_2d_scat (e01sa).

6: **px** – REAL (KIND=nag_wp)
7: **py** – REAL (KIND=nag_wp)

The point (px, py) at which the interpolant is to be evaluated.

5.2 Optional Input Parameters

1: **m** – INTEGER

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

m, **x**, **y**, **f**, **triang** and **grads** must be unchanged from the previous call of nag_interp_2d_scat (e01sa).

5.3 Output Parameters

1: **pf** – REAL (KIND=nag_wp)

The value of the interpolant evaluated at the point (px, py).

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 $= 1$

On entry, **m** < 3.

ifail $= 2$

On entry, the triangulation information held in the array **triang** does not specify a valid triangulation of the data points. **triang** may have been corrupted since the call to nag_interp_2d_scat (e01sa).

ifail $= 3$ (*warning*)

The evaluation point (**px**,**py**) lies outside the nodal triangulation, and the value returned in **pf** is computed by extrapolation.

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

Computational errors should be negligible in most practical situations.

8 Further Comments

The time taken for a call of nag_interp_2d_scat_eval (e01sb) is approximately proportional to the number of data points, m.

The results returned by this function are particularly suitable for applications such as graph plotting, producing a smooth surface from a number of scattered points.

9 Example

See Section 10 in nag_interp_2d_scat (e01sa).

9.1 Program Text

```
     function e01sb_example

fprintf(’e01sb example results\n\n’);

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);

px = [3:3:21];
py = [2:3:17];
% Evaluate interpolant at on regular mesh (px,py)
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’;
title  = ’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, title, chlab, ...
                  rlabs, chlab, clabs, ncols, indent);
```

9.2 Program Results

```
     e01sb example results

 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
```