NAG Library Function Document

nag_2d_triang_eval (e01skc)

1 Purpose

nag_2d_triang_eval (e01skc) evaluates at a given point the two-dimensional interpolant function computed by nag_2d_triang_interp (e01sjc).

2 Specification

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

```
void nag_2d_triang_eval (Integer m, const double x[], const double y[],
     const double f[], const Integer triang[], const double grads[],
     double px, double py, double *pf, NagError *fail)
```

3 Description

nag_2d_triang_eval (e01skc) 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_2d_shep_interp (e01sgc), 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_2d_triang_eval (e01skc) must only be called after a call to nag_2d_shep_interp (e01sgc).

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 Arguments

1: **m** – Integer *Input*

2: **x**[**m**] – const double *Input*

3: **y**[**m**] – const double *Input*

4: **f**[**m**] – const double *Input*

5: **triang**[**7** × **m**] – const Integer *Input*

6: **grads**[**2** × **m**] – const double *Input*

On entry: **m**, **x**, **y**, **f**, **triang** and **grads** must be unchanged from the previous call of nag_2d_triang_interp (e01sjc).

7: **px** – double *Input*

8: **py** – double *Input*

On entry: the point (px, py) at which the interpolant is to be evaluated.

9: **pf** – double * *Output*

On exit: the value of the interpolant evaluated at the point (px, py).

10: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument $\langle$*value*$\rangle$ had an illegal value.

NE_INT

On entry, $\mathbf{m} = \langle$*value*$\rangle$.
Constraint: $\mathbf{m} \geq 3$.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

NE_TRIANG_INVALID

On entry, **triang** does not contain a valid data point triangulation; **triang** may have been corrupted since the call to nag_2d_triang_interp (e01sjc).

NW_VALUE_EXTRAPOLATED

Warning – the evaluation point $(\langle$*value*$\rangle, \langle$*value*$\rangle)$ lies outside the triangulation boundary. The returned value was computed by extrapolation.

7 Accuracy

Computational errors should be negligible in most practical situations.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken for a call of nag_2d_triang_eval (e01skc) 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.

10 Example

See Section 10 in nag_2d_shep_interp (e01sgc).