NAG Toolbox

nag_fit_2dspline_evalv (e02de)

1 Purpose

nag_fit_2dspline_evalv (e02de) calculates values of a bicubic spline from its B-spline representation.

2 Syntax

```
[ff, ifail] = nag_fit_2dspline_evalv(x, y, lamda, mu, c, 'm', m, 'px', px, 'py', py)
```

```
[ff, ifail] = e02de(x, y, lamda, mu, c, 'm', m, 'px', px, 'py', py)
```

3 Description

nag_fit_2dspline_evalv (e02de) calculates values of the bicubic spline $s(x,y)$ at prescribed points (x_r, y_r), for $r = 1, 2, \ldots, m$, from its augmented knot sets $\{\lambda\}$ and $\{\mu\}$ and from the coefficients c_{ij}, for $i = 1, 2, \ldots, \mathbf{px} - 4$ and $j = 1, 2, \ldots, \mathbf{py} - 4$, in its B-spline representation

$$s(x,y) = \sum_{ij} c_{ij} M_i(x) N_j(y).$$

Here $M_i(x)$ and $N_j(y)$ denote normalized cubic B-splines, the former defined on the knots λ_i to λ_{i+4} and the latter on the knots μ_j to μ_{j+4}.

This function may be used to calculate values of a bicubic spline given in the form produced by nag_interp_2d_spline_grid (e01da), nag_fit_2dspline_panel (e02da), nag_fit_2dspline_grid (e02dc) and nag_fit_2dspline_sctr (e02dd). It is derived from the function B2VRE in Anthony *et al.* (1982).

4 References

Anthony G T, Cox M G and Hayes J G (1982) *DASL – Data Approximation Subroutine Library* National Physical Laboratory

Cox M G (1978) The numerical evaluation of a spline from its B-spline representation *J. Inst. Math. Appl.* **21** 135–143

5 Parameters

5.1 Compulsory Input Parameters

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

x and **y** must contain x_r and y_r, for $r = 1, 2, \ldots, m$, respectively. These are the coordinates of the points at which values of the spline are required. The order of the points is immaterial.

Constraint: **x** and **y** must satisfy

$$\mathbf{lamda}(4) \leq \mathbf{x}(r) \leq \mathbf{lamda}(\mathbf{px} - 3)$$

and

$$\mathbf{mu}(4) \leq \mathbf{y}(r) \leq \mathbf{mu}(\mathbf{py} - 3), \quad r = 1, 2, \ldots, m.$$

.

The spline representation is not valid outside these intervals.

3: **lamda**(**px**) – REAL (KIND=nag_wp) array
4: **mu**(**py**) – REAL (KIND=nag_wp) array

lamda and **mu** must contain the complete sets of knots $\{\lambda\}$ and $\{\mu\}$ associated with the x and y variables respectively.

Constraint: the knots in each set must be in nondecreasing order, with **lamda**($\mathbf{px} - 3$) > **lamda**(4) and **mu**($\mathbf{py} - 3$) > **mu**(4).

5: **c**(($\mathbf{px} - \mathbf{4}$) × ($\mathbf{py} - \mathbf{4}$)) – REAL (KIND=nag_wp) array

c($(\mathbf{py} - 4) \times (i - 1) + j$) must contain the coefficient c_{ij} described in Section 3, for $i = 1, 2, \ldots, \mathbf{px} - 4$ and $j = 1, 2, \ldots, \mathbf{py} - 4$.

5.2 Optional Input Parameters

1: **m** – INTEGER

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

m, the number of points at which values of the spline are required.

Constraint: $\mathbf{m} \geq 1$.

2: **px** – INTEGER
3: **py** – INTEGER

Default: For **px**, the dimension of the array **lamda**. For **py**, the dimension of the array **mu**.

px and **py** must specify the total number of knots associated with the variables x and y respectively. They are such that $\mathbf{px} - 8$ and $\mathbf{py} - 8$ are the corresponding numbers of interior knots.

Constraint: $\mathbf{px} \geq 8$ and $\mathbf{py} \geq 8$.

5.3 Output Parameters

1: **ff**(**m**) – REAL (KIND=nag_wp) array

ff(r) contains the value of the spline at the point (x_r, y_r), for $r = 1, 2, \ldots, m$.

2: **ifail** – INTEGER

$\mathbf{ifail} = 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

$\mathbf{ifail} = 1$

On entry, $\mathbf{m} < 1$,
or $\mathbf{py} < 8$,
or $\mathbf{px} < 8$.

$\mathbf{ifail} = 2$

On entry, the knots in array **lamda**, or those in array **mu**, are not in nondecreasing order, or **lamda**($\mathbf{px} - 3$) ≤ **lamda**(4), or **mu**($\mathbf{py} - 3$) ≤ **mu**(4).

$\mathbf{ifail} = 3$

On entry, at least one of the prescribed points (x_r, y_r) lies outside the rectangle defined by **lamda**(4), **lamda**($\mathbf{px} - 3$) and **mu**(4), **mu**($\mathbf{py} - 3$).

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

The method used to evaluate the B-splines is numerically stable, in the sense that each computed value of $s(x_r, y_r)$ can be regarded as the value that would have been obtained in exact arithmetic from slightly perturbed B-spline coefficients. See Cox (1978) for details.

8 Further Comments

Computation time is approximately proportional to the number of points, m, at which the evaluation is required.

9 Example

This program reads in knot sets **lamda**$(1), \ldots,$ **lamda**(**px**) and **mu**$(1), \ldots,$ **mu**(**py**), and a set of bicubic spline coefficients c_{ij}. Following these are a value for m and the coordinates (x_r, y_r), for $r = 1, 2, \ldots, m$, at which the spline is to be evaluated.

9.1 Program Text

```
     function e02de_example

fprintf('e02de example results\n\n');

% Spline knots and coefficients
lamda = [1    1     1     1     1.3   1.5   1.6   2     2     2     2];
mu    = [0    0     0     0     0.4   0.7   1     1     1     1];
c = [1      1.2    1.5833 2.1433 2.8667 3.4667 4;
     1.1333 1.3333 1.7167 2.2767 3      3.6    4.1333;
     1.3667 1.5667 1.95   2.51   3.2333 3.8333 4.3667;
     1.7    1.9    2.2833 2.8433 3.5667 4.1667 4.7;
     1.9    2.1    2.4833 3.0433 3.7667 4.3667 4.9;
     2      2.2    2.5833 3.1433 3.8667 4.4667 5];

% Evaluate spline at set of points for displaying

x     = [1    1.1   1.5   1.6   1.9   1.9   2];
y     = [0    0.1   0.7   0.4   0.3   0.8   1];

[ff, ifail] = e02de( ...
                     x, y, lamda, mu, c);

fprintf('        x          y          fit\n');
fprintf('%11.3f%11.3f%11.3f\n',[x; y; ff']);

% Evaluate spline on mesh for figure
mx = [1:0.05:2];
my = [0:0.05:1];
for i = 1:21
  xx(1:21) = mx(i);
  [ff(1:21,i), ifail] = e02de( ...
                               xx, my, lamda, mu, c);
end
```

```
fig1 = figure;
meshc(mx,my,ff);
xlabel('x');
ylabel('y');
title('Least-squares bi-cubic spline surface');
```

9.2 Program Results

```
e02de example results

   x          y          fit
 1.000      0.000      1.000
 1.100      0.100      1.310
 1.500      0.700      2.950
 1.600      0.400      2.960
 1.900      0.300      3.910
 1.900      0.800      4.410
 2.000      1.000      5.000
```