NAG Toolbox

nag_fit_2dspline_evalm (e02df)

1 Purpose

nag_fit_2dspline_evalm (e02df) calculates values of a bicubic spline from its B-spline representation. The spline is evaluated at all points on a rectangular grid.

2 Syntax

```
[ff, ifail] = nag_fit_2dspline_evalm(x, y, lamda, mu, c, 'mx', mx, 'my', my, 'px', px, 'py', py)
[ff, ifail] = e02df(x, y, lamda, mu, c, 'mx', mx, 'my', my, 'px', px, 'py', py)
```

3 Description

nag_fit_2dspline_evalm (e02df) calculates values of the bicubic spline $s(x,y)$ on a rectangular grid of points in the x-y plane, 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}.

The points in the grid are defined by coordinates x_q, for $q = 1,2,\ldots,m_x$, along the x axis, and coordinates y_r, for $r = 1,2,\ldots,m_y$, along the y axis.

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**(**mx**) – REAL (KIND=nag_wp) array

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

x and **y** must contain x_q, for $q = 1,2,\ldots,m_x$, and y_r, for $r = 1,2,\ldots,m_y$, respectively. These are the x and y coordinates that define the rectangular grid of points at which values of the spline are required.

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

$$\mathbf{lamda}(4) \leq \mathbf{x}(q) < \mathbf{x}(q+1) \leq \mathbf{lamda}(\mathbf{px}-3), \quad q = 1,2,\ldots,m_x-1$$

and

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

.

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**(**px** $-$ 3) $>$ **lamda**(4) and **mu**(**py** $-$ 3) $>$ **mu**(4).

5: **c**((**px** $-$ **4**) $\times$ (**py** $-$ **4**)) – REAL (KIND=nag_wp) array

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

5.2 Optional Input Parameters

1: **mx** – INTEGER

2: **my** – INTEGER

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

mx and **my** must specify m_x and m_y respectively, the number of points along the x and y axis that define the rectangular grid.

Constraint: **mx** ≥ 1 and **my** ≥ 1.

3: **px** – INTEGER

4: **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 **px** $-$ 8 and **py** $-$ 8 are the corresponding numbers of interior knots.

Constraint: **px** ≥ 8 and **py** ≥ 8.

5.3 Output Parameters

1: **ff**(**mx** $\times$ **my**) – REAL (KIND=nag_wp) array

ff(**my** $\times$ $(q-1)+r$) contains the value of the spline at the point (x_q, y_r), for $q = 1,2,\ldots,m_x$ and $r = 1,2,\ldots,m_y$.

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, **mx** < 1,
or **my** < 1,
or **py** < 8,

or **px** < 8.

ifail $= 2$

On entry, *lwrk* is too small,
or *liwrk* is too small.

ifail $= 3$

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

ifail $= 4$

On entry, the restriction **lamda**$(4) \leq \mathbf{x}(1) < \cdots < \mathbf{x}(\mathbf{mx}) \leq$ **lamda**$(\mathbf{px} - 3)$, or the restriction **mu**$(4) \leq \mathbf{y}(1) < \cdots < \mathbf{y}(\mathbf{my}) \leq$ **mu**$(\mathbf{py} - 3)$, is violated.

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 $m_x m_y + 4\big(m_x + m_y\big)$.

9 Example

This example reads in knot sets **lamda**$(1), \ldots,$ **lamda**$(\mathbf{px})$ and **mu**$(1), \ldots,$ **mu**$(\mathbf{py})$, and a set of bicubic spline coefficients c_{ij}. Following these are values for m_x and the x coordinates x_q, for $q = 1, 2, \ldots, m_x$, and values for m_y and the y coordinates y_r, for $r = 1, 2, \ldots, m_y$, defining the grid of points on which the spline is to be evaluated.

9.1 Program Text

```
function e02df_example

fprintf('e02df example results\n\n');

% knots
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];
% Coefficients
c     = [1       1.1333  1.3667  1.7   1.9     2  ...
         1.2     1.3333  1.5667  1.9     2.1   2.2 ...
         1.5833  1.7167  1.95    2.2833  2.4833  2.5833 ...
         2.1433  2.2767  2.51    2.8433  3.0433  3.1433 ...
         2.8667  3       3.2333  3.5667  3.7667  3.8667 ...
         3.4667  3.6     3.8333  4.1667  4.3667  4.4667 ...
         4       4.1333  4.3667  4.7     4.9     5];
```

```
%Evaluation points
x = [1.0  1.1  1.3  1.4  1.5  1.7  2.0];
y = [0 : 0.2 : 1];

% Spline evaluation
[ff, ifail] = e02df( ...
                    x, y, lamda, mu, c);

fprintf(' Spline evaluated on x-y grid (x across, y down):\n');
ff = reshape(ff,[6,7]);
fprintf('%5s%9.1f%9.1f%9.1f%9.1f%9.1f%9.1f%9.1f\n',' ',x);
for i = 1:6
  fprintf('%5.1f%9.3f%9.3f%9.3f%9.3f%9.3f%9.3f%9.3f\n',y(i),ff(i,:));
end
```

9.2 Program Results

```
     e02df example results

 Spline evaluated on x-y grid (x across, y down):
          1.0      1.1      1.3      1.4      1.5      1.7      2.0
  0.0    1.000    1.210    1.690    1.960    2.250    2.890    4.000
  0.2    1.200    1.410    1.890    2.160    2.450    3.090    4.200
  0.4    1.400    1.610    2.090    2.360    2.650    3.290    4.400
  0.6    1.600    1.810    2.290    2.560    2.850    3.490    4.600
  0.8    1.800    2.010    2.490    2.760    3.050    3.690    4.800
  1.0    2.000    2.210    2.690    2.960    3.250    3.890    5.000
```