NAG Toolbox

nag_interp_2d_spline_grid (e01da)

1 Purpose

nag_interp_2d_spline_grid (e01da) computes a bicubic spline interpolating surface through a set of data values, given on a rectangular grid in the x-y plane.

2 Syntax

```
[px, py, lamda, mu, c, ifail] = nag_interp_2d_spline_grid(x, y, f, 'mx', mx,
'my', my)
```

```
[px, py, lamda, mu, c, ifail] = e01da(x, y, f, 'mx', mx, 'my', my)
```

3 Description

nag_interp_2d_spline_grid (e01da) determines a bicubic spline interpolant to the set of data points $\left(x_q, y_r, f_{q,r}\right)$, for $q = 1, 2, \ldots, m_x$ and $r = 1, 2, \ldots, m_y$. The spline is given in the B-spline representation

$$s(x,y) = \sum_{i=1}^{m_x} \sum_{j=1}^{m_y} c_{ij} M_i(x) N_j(y),$$

such that

$$s\left(x_q, y_r\right) = f_{q,r},$$

where $M_i(x)$ 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}, and the c_{ij} are the spline coefficients. These knots, as well as the coefficients, are determined by the function, which is derived from the function B2IRE in Anthony *et al.* (1982). The method used is described in Section 9.2.

For further information on splines, see Hayes and Halliday (1974) for bicubic splines and de Boor (1972) for normalized B-splines.

Values and derivatives of the computed spline can subsequently be computed by calling nag_fit_2dspline_evalv (e02de), nag_fit_2dspline_evalm (e02df) or nag_fit_2dspline_derivm (e02dh) as described in Section 9.3.

4 References

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

Cox M G (1975) An algorithm for spline interpolation *J. Inst. Math. Appl.* **15** 95–108

de Boor C (1972) On calculating with B-splines *J. Approx. Theory* **6** 50–62

Hayes J G and Halliday J (1974) The least squares fitting of cubic spline surfaces to general data sets *J. Inst. Math. Appl.* **14** 89–103

5 Parameters

5.1 Compulsory Input Parameters

1: $\mathbf{x}(\mathbf{mx})$ – REAL (KIND=nag_wp) array

2: $\mathbf{y}(\mathbf{my})$ – REAL (KIND=nag_wp) array

$\mathbf{x}(q)$ and $\mathbf{y}(r)$ must contain x_q, for $q = 1, 2, \ldots, m_x$, and y_r, for $r = 1, 2, \ldots, m_y$, respectively.

Constraints:

$\mathbf{x}(q) < \mathbf{x}(q+1)$, for $q = 1, 2, \ldots, m_x - 1$;
$\mathbf{y}(r) < \mathbf{y}(r+1)$, for $r = 1, 2, \ldots, m_y - 1$.

3: $\mathbf{f}(\mathbf{mx} \times \mathbf{my})$ – REAL (KIND=nag_wp) array

$\mathbf{f}(m_y \times (q-1) + r)$ must contain $f_{q,r}$, for $q = 1, 2, \ldots, m_x$ and $r = 1, 2, \ldots, m_y$.

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: $\mathbf{mx} \geq 4$ and $\mathbf{my} \geq 4$.

5.3 Output Parameters

1: **px** – INTEGER

2: **py** – INTEGER

px and **py** contain $m_x + 4$ and $m_y + 4$, the total number of knots of the computed spline with respect to the x and y variables, respectively.

3: $\mathbf{lamda}(\mathbf{mx} + \mathbf{4})$ – REAL (KIND=nag_wp) array

4: $\mathbf{mu}(\mathbf{my} + \mathbf{4})$ – REAL (KIND=nag_wp) array

lamda contains the complete set of knots λ_i associated with the x variable, i.e., the interior knots $\mathbf{lamda}(5), \mathbf{lamda}(6), \ldots, \mathbf{lamda}(\mathbf{px} - 4)$, as well as the additional knots

$$\mathbf{lamda}(1) = \mathbf{lamda}(2) = \mathbf{lamda}(3) = \mathbf{lamda}(4) = \mathbf{x}(1)$$

and

$$\mathbf{lamda}(\mathbf{px} - 3) = \mathbf{lamda}(\mathbf{px} - 2) = \mathbf{lamda}(\mathbf{px} - 1) = \mathbf{lamda}(\mathbf{px}) = \mathbf{x}(\mathbf{mx})$$

needed for the B-spline representation.

5: $\mathbf{c}(\mathbf{mx} \times \mathbf{my})$ – REAL (KIND=nag_wp) array

The coefficients of the spline interpolant. $\mathbf{c}(m_y \times (i-1) + j)$ contains the coefficient c_{ij} described in Section 3.

6: **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:

ifail $= 1$

On entry, **mx** < 4,
or **my** < 4.

ifail $= 2$

On entry, either the values in the **x** array or the values in the **y** array are not in increasing order if not already there.

ifail $= 3$

A system of linear equations defining the B-spline coefficients was singular; the problem is too ill-conditioned to permit solution.

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 main sources of rounding errors are in steps 2, 3, 6 and 7 of the algorithm described in Section 9.2. It can be shown (see Cox (1975)) that the matrix A_x formed in step 2 has elements differing relatively from their true values by at most a small multiple of 3ϵ, where ϵ is the ***machine precision***. A_x is 'totally positive', and a linear system with such a coefficient matrix can be solved quite safely by elimination without pivoting. Similar comments apply to steps 6 and 7. Thus the complete process is numerically stable.

8 Further Comments

8.1 Timing

The time taken by nag_interp_2d_spline_grid (e01da) is approximately proportional to $m_x m_y$.

8.2 Outline of Method Used

The process of computing the spline consists of the following steps:

1. choice of the interior x-knots λ_5, $\lambda_6, \ldots, \lambda_{m_x}$ as $\lambda_i = x_{i-2}$, for $i = 5, 6, \ldots, m_x$,
2. formation of the system

$$A_x E = F,$$

where A_x is a band matrix of order m_x and bandwidth 4, containing in its qth row the values at x_q of the B-splines in x, **f** is the m_x by m_y rectangular matrix of values $f_{q,r}$, and E denotes an m_x by m_y rectangular matrix of intermediate coefficients,

3. use of Gaussian elimination to reduce this system to band triangular form,
4. solution of this triangular system for E,
5. choice of the interior y knots μ_5, $\mu_6, \ldots, \mu_{m_y}$ as $\mu_i = y_{i-2}$, for $i = 5, 6, \ldots, m_y$,

6. formation of the system

$$A_y C^{\mathrm{T}} = E^{\mathrm{T}},$$

   where A_y is the counterpart of A_x for the y variable, and C denotes the m_x by m_y rectangular matrix of values of c_{ij},

7. use of Gaussian elimination to reduce this system to band triangular form,

8. solution of this triangular system for C^{T} and hence C.

For computational convenience, steps 2 and 3, and likewise steps 6 and 7, are combined so that the formation of A_x and A_y and the reductions to triangular form are carried out one row at a time.

8.3 Evaluation of Computed Spline

The values of the computed spline at the points (x_k, y_k), for $k = 1, 2, \ldots, m$, may be obtained in the double array **ff** (see nag_fit_2dspline_evalv (e02de)), of length at least m, by the following call:

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

where $\texttt{M} = m$ and the coordinates x_k, y_k are stored in $\texttt{X}(k)$, $\texttt{Y}(k)$. `LAMDA`, `MU` and `C` have the same values as **lamda**, **mu** and **c** output from nag_interp_2d_spline_grid (e01da). (See nag_fit_2dspline_evalv (e02de).)

To evaluate the computed spline on an m_x by m_y rectangular grid of points in the x-y plane, which is defined by the x coordinates stored in $\texttt{X}(j)$, for $j = 1, 2, \ldots, m_x$, and the y coordinates stored in $\texttt{Y}(k)$, for $k = 1, 2, \ldots, m_y$, returning the results in the double array **ff** (see nag_fit_2dspline_evalm (e02df)) which is of length at least **mx** $\times$ **my**, the following call may be used:

```
[fg, ifail] = e02df(x, y, lamda, mu, c);
```

where $\texttt{MX} = m_x$, $\texttt{MY} = m_y$. `LAMDA`, `MU` and `C` have the same values as **lamda**, **mu** and **c** output from nag_interp_2d_spline_grid (e01da).

The result of the spline evaluated at grid point (j, k) is returned in element $(\texttt{MY} \times (j-1) + k)$ of the array `FG`.

9 Example

This example reads in values of m_x, x_q, for $q = 1, 2, \ldots, m_x$, m_y and y_r, for $r = 1, 2, \ldots, m_y$, followed by values of the ordinates $f_{q,r}$ defined at the grid points $\left(x_q, y_r\right)$.

It then calls nag_interp_2d_spline_grid (e01da) to compute a bicubic spline interpolant of the data values, and prints the values of the knots and B-spline coefficients. Finally it evaluates the spline at a small sample of points on a rectangular grid.

9.1 Program Text

```
     function e01da_example

fprintf('e01da example results\n\n');

x = [1.0   1.10   1.30   1.50   1.60   1.80   2.0];
f = [1.0   1.21   1.69   2.25   2.56   3.24   4.0;
     1.1   1.31   1.79   2.35   2.66   3.34   4.1;
     1.4   1.61   2.09   2.65   2.96   3.64   4.4;
     1.7   1.91   2.39   2.95   3.26   3.94   4.7;
     1.9   2.11   2.59   3.15   3.46   4.14   4.9;
     2.0   2.21   2.69   3.25   3.56   4.24   5.0];
y = [0.0;
     0.1;
     0.4;
     0.7;
     0.9;
     1.0];
```

```
[px, py, lamda, mu, c, ifail] = e01da( ...
                                       x, y, f);

% Display the knot sets, LAMDA and MU.

 fprintf('\n               i   knot lamda(i)      j     knot mu(j)\n');
j = 4:min(px,py)-3;
fprintf('%16d%12.4f%11d%12.4f\n',[j' lamda(j) j' mu(j)]');
if (px>py);
  j = py-2:px-3;
  fprintf('%16d%12.4f\n',[j' lamda(j)]');
elseif (px<py);
  j = px-2:py-3;
  fprintf('%16d%12.4f\n',[j' mu(j)]')
end

% Display the spline coefficients.
c = reshape(c, size(f'));
fprintf('\n');
disp('The B-Spline coefficients:');
disp(c');

% Evaluate spline on regular 6-by-6 mesh
dx = (x(end)-x(1))/5;
dy = (y(end)-y(1))/5;
tx = [x(1):dx:x(end)];
ty = [y(1):dy:y(end)]';

[ff, ifail] = e02df( ...
                     tx, ty, lamda, mu, c);

% Display evaluations as matrix
ff = reshape(ff,[6,6]);

matrix = 'General';
diag = 'Non-unit';
format = 'F8.3';
title  = 'Spline evaluated on a regular mesh (x across, y down):';
chlab  = 'Character';
rlabs  = cellstr(num2str(tx'));
clabs  = cellstr(num2str(ty));
ncols  = nag_int(80);
indent = nag_int(0);
[ifail] =  x04cb( ...
                  matrix, diag, ff, format, title, chlab, ...
                  rlabs, chlab, clabs, ncols, indent);
```

9.2 Program Results

```
     e01da example results

               i   knot lamda(i)      j     knot mu(j)
               4      1.0000          4      0.0000
               5      1.0000          5      0.0000
               6      2.0000          6      1.0000
               7      2.0000          7      1.0000
               8      2.0000

The B-Spline coefficients:
    1.0000    1.1333    1.3667    1.7000    1.9000    2.0000    1.2000
    1.3333    1.5667    1.9000    2.1000    2.2000    1.5833    1.7167
    1.9500    2.2833    2.4833    2.5833    2.1433    2.2767    2.5100
    2.8433    3.0433    3.1433    2.8667    3.0000    3.2333    3.5667
    3.7667    3.8667    3.4667    3.6000    3.8333    4.1667    4.3667
    4.4667    4.0000    4.1333    4.3667    4.7000    4.9000    5.0000

 Spline evaluated on a regular mesh (x across, y down):
            0     0.2     0.4     0.6     0.8       1
   1    1.000   1.440   1.960   2.560   3.240   4.000
```

```
 1.2    1.200   1.640   2.160   2.760   3.440   4.200
 1.4    1.400   1.840   2.360   2.960   3.640   4.400
 1.6    1.600   2.040   2.560   3.160   3.840   4.600
 1.8    1.800   2.240   2.760   3.360   4.040   4.800
   2    2.000   2.440   2.960   3.560   4.240   5.000
```