NAG C Library Function Document

nag_2d_spline_interpolant (e01dac)

1 Purpose

nag_2d_spline_interpolant (e01dac) computes a bicubic spline interpolating surface through a set of data values, given on a rectangular grid in the x-y plane.

2 Specification

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

void nag_2d_spline_interpolant(Integer mx, Integer my, double x[], double y[],
          double f[], Nag_2dSpline *spline, NagError *fail)
```

3 Description

This function determines a bicubic spline interpolant to the set of data points $(x_q, y_r, f_{q,r})$, 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

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

where $M_i(x)$ and $N_j(y)$ denote normalised 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 routine B2IRE in Anthony *et al.* (1982). The method used is described in Section 6.2.

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

Values of the computed spline can subsequently be obtained by calling nag_2d_spline_eval (e02dec) or nag_2d_spline_eval_rect (e02dfc) as described in Section 6.3.

4 Parameters

1: **mx** – Integer *Input*
2: **my** – Integer *Input*

On entry: **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** ≥ 4 and **my** ≥ 4.

3: **x[mx]** – double *Input*
4: **y[my]** – double *Input*

On entry: **x**$[q-1]$ and **y**$[r-1]$ must contain x_q, for $q = 1, 2, \ldots, m_x$, and y_r, for $r = 1, 2, \ldots, m_y$, respectively.

Constraints:

x$[q-1] <$ **x**$[q]$, for $q = 1, 2, \ldots, m_x - 1$,

y$[r-1] <$ **y**$[r]$, for $r = 1, 2, \ldots, m_y - 1$.

5: **f[mx*my]** – double *Input*

On entry: **f**$[m_y \times (q-1) + r - 1]$ must contain $f_{q,r}$, for $q = 1, 2, \ldots, m_x$ and $r = 1, 2, \ldots, m_y$.

6: **spline** – Nag_2dSpline *

Pointer to structure of type **Nag_2dSpline** with the following members:

nx – Integer *Output*
ny – Integer *Output*

On exit: **spline.nx** and **spline.ny** 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.

lamda – double * *Output*

On exit: pointer to which memory of size **spline.nx** is internally allocated. **spline.lamda** contains the complete set of knots λ_i associated with the x variable, i.e., the interior knots **spline.lamda**[4], **spline.lamda**[5], ..., **spline.lamda**[**spline.nx**−5], as well as the additional knots **spline.lamda**[0] = **spline.lamda**[1] = **spline.lamda**[2] = **spline.lamda**[3] = **x**[0] and **spline.lamda**[**spline.nx**−4] = **spline.lamda**[**spline.nx**−3] = **spline.lamda**[**spline.nx**−2] = **spline.lamda**[**spline.nx**−1] = **x**[**mx**−1] needed for the B-spline representation.

mu – double * *Output*

On exit: pointer to which memory of size **spline.ny** is internally allocated. **spline.mu** contains the corresponding complete set of knots μ_i associated with the y variable.

c – double * *Output*

On exit: pointer to which memory of size **mx** × **my** is internally allocated. **spline.c** holds the coefficients of the spline interpolant. **spline.c** $[m_y \times (i-1) + j - 1]$ contains the coefficient c_{ij} described in Section 3.

Note that when the information contained in the pointers **spline.lamda**, **spline.mu** and **spline.c** is no longer of use, or before a new call to nag_2d_spline_interpolant with the same **spline**, the user should free these pointers using the NAG macro `NAG_FREE`. This storage will not have been allocated if this function returns with **fail.code** ≠ **NE_NOERROR**.

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

The NAG error parameter (see the Essential Introduction).

5 Error Indicators and Warnings

NE_INT_ARG_LT

On entry, **mx** must not be less than 4: **mx** = <*value*>.
On entry, **my** must not be less than 4: **my** = <*value*>.

NE_NOT_STRICTLY_INCREASING

The sequence **x** is not strictly increasing: **x**[<*value*>] = <*value*>, **x**[<*value*>] = <*value*>.
The sequence **y** is not strictly increasing: **y**[<*value*>] = <*value*>, **y**[<*value*>] = <*value*>.

NE_ALLOC_FAIL

Memory allocation failed.

NE_DATA_ILL_CONDITIONED

An intermediate set of linear equations is singular, the data is too ill-conditioned to compute B-spline coefficients.

6 Further Comments

The time taken by this routine is approximately proportional to $m_x m_y$.

6.1 Accuracy

The main sources of rounding errors are in steps (2), (3), (6) and (7) of the algorithm described in Section 6.2. It can be shown (Cox (1975a)) 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.

6.2 Outline of Method Used

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

(1) choice of the interior x-knots $\lambda_5, \lambda_6, \ldots, \lambda_{m_x}$ 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 $\mu_5, \mu_6, \ldots, \mu_{m_y}$ as $\mu_i = y_{i-2}$, for $i = 5, 6, \ldots, m_y$,

(6) formation of the system

$$A_y C^T = E^T,$$

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.

6.3 Evaluation of Computed Spline

The values of the computed spline at the points (**tx**[$r-1$],**ty**[$r-1$]), for $r = 1, 2, \ldots$,**n**, may be obtained in the array **ff**, of length at least **n**, by the following call:

```
e02dec(n, tx, ty, ff, &spline, &fail)
```

where **spline** is a structure of type **Nag_2dSpline** which is the output parameter of nag_2d_spline_interpolant.

To evaluate the computed spline on a **kx** by **ky** rectangular grid of points in the x-y plane, which is defined by the x co-ordinates stored in **tx**[$q-1$], for $q = 1, 2, \ldots$,**kx** , and the y co-ordinates stored in **ty**[$r-1$], for $r = 1, 2, \ldots$,**ky**, returning the results in the array **fg** which is of length at least **kx** $\times$ **ky**, the following call may be used:

```
e02dfc(kx, ky, tx, ty, fg, &spline, &fail)
```

where **spline** is a structure of type **Nag_2dSpline** which is the output parameter of nag_2d_spline_interpolant. The result of the spline evaluated at grid point (q, r) is returned in element [**ky** $\times$ $(q-1) + r - 1$] of the array **fg**.

6.4 References

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

Cox M G (1975a) 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

7 See Also

nag_2d_spline_eval (e02dec)
nag_2d_spline_eval_rect (e02dfc)

8 Example

This program 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 (x_q, y_r). It then calls nag_2d_spline_interpolant 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.

8.1 Program Text

```
/* nag_2d_spline_interpolant(e01dac) Example Program
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 *
 * Mark 6 revised, 2000.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nage01.h>
#include <nage02.h>

#define MXMAX 20
#define MYMAX 20
#define F(I,J)  f[my*(I)+(J)]
#define FG(I,J) fg[npy*(I)+(J)]
#define C(I,J)  spline.c[my*(I)+(J)]

main()
{
  Integer i, j, mx, my, npx, npy;
  double  f[MXMAX*MYMAX], x[MXMAX], y[MYMAX];
  double fg[MXMAX*MYMAX], tx[MXMAX], ty[MYMAX];
  double  xhi, yhi, xlo, ylo, step;
  Nag_2dSpline spline;

  Vprintf("e01dac Example Program Results\n");
  Vscanf("%*[^\n]"); /* Skip heading in data file */
  /* Read the number of x points, mx, and the values of the
   * x co-ordinates.
   */
```

```
  Vscanf("%ld%ld",&mx, &my);
  if (mx>MXMAX ||  my>MYMAX)
    {
      Vfprintf(stderr, "mx or my is out of range: mx = %5ld\n,\
my = %5ld\n",mx,my);
      exit(EXIT_FAILURE);
    }
  for (i=0; i<mx;  i++)
    Vscanf("%lf",&x[i]);
  /* Read the number of y points, my, and the values of the
   * y co-ordinates.
   */
  for (i=0; i<my; i++)
    Vscanf("%lf",&y[i]);
  /* Read the function values at the grid points. */
  for (j=0; j<my; j++)
    for (i=0; i<mx; i++)
      Vscanf("%lf",&F(i,j));
  /* Generate the (x,y,f) interpolating bicubic B-spline. */
  e01dac(mx, my, x, y, f, &spline, NAGERR_DEFAULT);

  /* Print the knot sets, lamda and mu. */
  Vprintf("Distinct knots in x direction located  at\n");
  for (j=3; j<spline.nx-3; j++)
    Vprintf("%12.4f%s",spline.lamda[j],((j-3)%5==4 || j==spline.nx-4)
            ? "\n" : " ");
  Vprintf("\nDistinct knots in y direction located  at\n");
  for (j=3; j<spline.ny-3; j++)
    Vprintf("%12.4f%s",spline.mu[j],((j-3)%5==4 || j==spline.ny-4)
            ? "\n" : " ");
  /* Print the spline coefficients. */
  Vprintf("\nThe B-Spline coefficients:\n");
  for (i=0; i<mx; i++)
    {
      for (j=0; j<my; j++)
        Vprintf("%9.4f",C(i,j));
      Vprintf("\n");
    }

  /* Evaluate the spline on a regular rectangular grid at npx*npy
   * points over the domain (xlo to xhi) x (ylo to yhi).
   */
  Vscanf("%ld%lf%lf",&npx,&xlo,&xhi);
  Vscanf("%ld%lf%lf",&npy,&ylo,&yhi);
  if (npx<=MXMAX && npy<=MYMAX)
    {
      step = (xhi-xlo)/(double)(npx-1);
      Vprintf("\nSpline evaluated on a regular mesh \
     (x across, y down): \n     ");
      /* Generate nx equispaced x co-ordinates. */
      for (i=0; i<npx; i++)
        {
          tx[i] = MIN(xlo+i*step,xhi);
          Vprintf("   %5.2f ",tx[i]);
        }
      step = (yhi-ylo)/(npy-1);
      for (i=0; i<npy; i++)
        ty[i] = MIN(ylo+i*step,yhi);
```

```
      /* Evaluate the spline. */
      e02dfc(npx, npy, tx, ty, fg, &spline, NAGERR_DEFAULT);

      /* Print the results. */
      Vprintf("\n");
      for (j=0; j<npy; j++)
        {
          Vprintf("%5.2f",ty[j]);
          for (i=0; i<npx; i++)
            Vprintf("%8.3f ",FG(i,j));
          Vprintf("\n");

        }
      /* Free memory allocated by e01dac */
      NAG_FREE(spline.lamda);
      NAG_FREE(spline.mu);
      NAG_FREE(spline.c);
      exit(EXIT_SUCCESS);
    }
  else
    {
      Vfprintf(stderr, "npx or npy is out of range: npx = %5ld, npy = %5ld\n",
               npx,npy);
      /* Free memory allocated by e01dac */
      NAG_FREE(spline.lamda);
      NAG_FREE(spline.mu);
      NAG_FREE(spline.c);
      exit(EXIT_FAILURE);
    }
}
```

8.2 Program Data

```
e01dac Example Program Data
  7 6
  1.00  1.10  1.30  1.50  1.60  1.80  2.00
  0.00  0.10  0.40  0.70  0.90  1.00
  1.00  1.21  1.69  2.25  2.56  3.24  4.00
  1.10  1.31  1.79  2.35  2.66  3.34  4.10
  1.40  1.61  2.09  2.65  2.96  3.64  4.40
  1.70  1.91  2.39  2.95  3.26  3.94  4.70
  1.90  2.11  2.59  3.15  3.46  4.14  4.90
  2.00  2.21  2.69  3.25  3.56  4.24  5.00
  6  1.0  2.0
  6  0.0  1.0
```

8.3 Program Results

```
e01dac Example Program Results
Distinct knots in x direction located  at
      1.0000       1.3000       1.5000       1.6000       2.0000

Distinct knots in y direction located  at
      0.0000       0.4000       0.7000       1.0000

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):
         1.00     1.20     1.40     1.60     1.80     2.00
 0.00   1.000    1.440    1.960    2.560    3.240    4.000
 0.20   1.200    1.640    2.160    2.760    3.440    4.200
 0.40   1.400    1.840    2.360    2.960    3.640    4.400
 0.60   1.600    2.040    2.560    3.160    3.840    4.600
 0.80   1.800    2.240    2.760    3.360    4.040    4.800
 1.00   2.000    2.440    2.960    3.560    4.240    5.000
```