NAG Toolbox

nag_fit_2dspline_ts_evalv (e02je)

1 Purpose

nag_fit_2dspline_ts_evalv (e02je) calculates a vector of values of a spline computed by nag_fit_2dspline_ts_sctr (e02jd).

2 Syntax

```
[fevalv, ifail] = nag_fit_2dspline_ts_evalv(xevalv, yevalv, coefs, iopts, opts, 'nevalv', nevalv)
[fevalv, ifail] = e02je(xevalv, yevalv, coefs, iopts, opts, 'nevalv', nevalv)
```

3 Description

nag_fit_2dspline_ts_evalv (e02je) calculates values at prescribed points (x_i,y_i), for $i = 1,2,\ldots,n$, of a bivariate spline computed by nag_fit_2dspline_ts_sctr (e02jd). It is derived from the TSFIT package of O. Davydov and F. Zeilfelder.

4 References

Davydov O, Morandi R and Sestini A (2006) Local hybrid approximation for scattered data fitting with bivariate splines *Comput. Aided Geom. Design* **23** 703–721

Davydov O, Sestini A and Morandi R (2005) Local RBF approximation for scattered data fitting with bivariate splines *Trends and Applications in Constructive Approximation* M. G. de Bruin, D. H. Mache, and J. Szabados, Eds **ISNM Vol. 151** Birkhauser 91–102

Davydov O and Zeilfelder F (2004) Scattered data fitting by direct extension of local polynomials to bivariate splines *Advances in Comp. Math.* **21** 223–271

Farin G and Hansford D (2000) *The Essentials of CAGD* Natic, MA: A K Peters, Ltd.

5 Parameters

5.1 Compulsory Input Parameters

1: **xevalv**(**nevalv**) – REAL (KIND=nag_wp) array

The (x_i) values at which the spline is to be evaluated.

Constraint: for all i, **xevalv**(i) must lie inside, or on the boundary of, the spline's bounding box as determined by nag_fit_2dspline_ts_sctr (e02jd).

2: **yevalv**(**nevalv**) – REAL (KIND=nag_wp) array

The (y_i) values at which the spline is to be evaluated.

Constraint: for all i, **yevalv**(i) must lie inside, or on the boundary of, the spline's bounding box as determined by nag_fit_2dspline_ts_sctr (e02jd).

3: **coefs**$(*)$ – REAL (KIND=nag_wp) array

Note: the dimension of this array is dictated by the requirements of associated functions that must have been previously called. This array **must** be the same array passed as argument **coefs** in the previous call to nag_fit_2dspline_ts_sctr (e02jd).

The computed spline coefficients as output from nag_fit_2dspline_ts_sctr (e02jd).

4: **iopts**$(*)$ – INTEGER array

Note: the dimension of this array is dictated by the requirements of associated functions that must have been previously called. This array **must** be the same array passed as argument **iopts** in the previous call to nag_fit_opt_set (e02zk).

The contents of the array **must not** have been modified either directly or indirectly, by a call to nag_fit_opt_set (e02zk), between calls to nag_fit_2dspline_ts_sctr (e02jd) and nag_fit_2dspline_ts_evalv (e02je).

5: **opts**$(*)$ – REAL (KIND=nag_wp) array

Note: the dimension of this array is dictated by the requirements of associated functions that must have been previously called. This array **must** be the same array passed as argument **opts** in the previous call to nag_fit_opt_set (e02zk).

The contents of the array **must not** have been modified either directly or indirectly, by a call to nag_fit_opt_set (e02zk), between calls to nag_fit_2dspline_ts_sctr (e02jd) and nag_fit_2dspline_ts_evalv (e02je).

5.2 Optional Input Parameters

1: **nevalv** – INTEGER

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

n, the number of values at which the spline is to be evaluated.

Constraint: **nevalv** ≥ 1.

5.3 Output Parameters

1: **fevalv**(**nevalv**) – REAL (KIND=nag_wp) array

If **ifail** $= 0$ on exit **fevalv**(i) contains the computed spline value at (x_i, y_i).

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 $= 2$

Constraint: **nevalv** ≥ 1.

ifail $= 9$

Option arrays are not initialized or are corrupted.

ifail $= 10$

The fitting routine has not been called, or the array of coefficients has been corrupted.

ifail $= 13$

Constraint: $\langle value \rangle \leq$ **xevalv**$(i) \leq \langle value \rangle$ for all i.

ifail $= 14$

Constraint: $\langle value \rangle \leq$ **yevalv**$(i) \leq \langle value \rangle$ for all i.

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

nag_fit_2dspline_ts_evalv (e02je) uses the de Casteljau algorithm and thus is numerically stable. See Farin and Hansford (2000) for details.

8 Further Comments

To evaluate a C^1 approximation (i.e., when **Global Smoothing Level** $= 1$), a real array of length $O(1)$ is dynamically allocated by each invocation of nag_fit_2dspline_ts_evalv (e02je). No memory is allocated internally when evaluating a C^2 approximation.

9 Example

See Section 10 in nag_fit_2dspline_ts_sctr (e02jd).

9.1 Program Text

```
function e02je_example

fprintf('e02je example results\n\n');

xdata = [0; 0.5; 1; 1.5; 2; 2.5; 3; 4; 4.5; 5; 5.5; 6; 7; 7.5; 8];
ydata = [-1.1; -0.372; 0.431; 1.69; 2.11; 3.1; 4.23; 4.35; 4.81; 4.61; 4.79; ...
          5.23; 6.35; 7.19; 7.97];
wdata = [1; 1; 1.5; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
cstart = 'c';
sfac  = 0.001;
x     = [6.5178; 7.2463; 1.0159; 7.3070; 5.0589; 0.7803; 2.2280; 4.3751; ...
         7.6601; 7.7191; 1.2609; 7.7647; 7.6573; 3.8830; 6.4022; 1.1351; ...
         3.3741; 7.3259; 6.3377; 7.6759];
nest  = nag_int(numel(xdata) + 4);
ixloc = zeros(numel(x), 1, nag_int_name);
wrk   = zeros(4*numel(xdata) + 16*nest + 41, 1);
iwrk1 = zeros(nest, 1, nag_int_name);
iwrk2 = zeros(3+3*numel(x), 1, nag_int_name);
lamda = zeros(nest, 1);
xord  = nag_int(0);
start = nag_int(0);
deriv = nag_int(3);

% Generate the data to fit.
[x, y, f, lsminp, lsmaxp, nxcels, nycels] = generate_data();

% Initialize the options arrays and set/get some options.
[iopts, opts] = handle_options();

% Compute the spline coefficients.
[coefs, iopts, opts, ifail] = ...
    e02jd(x, y, f, lsminp, lsmaxp, nxcels, nycels, iopts, opts);

% pmin and pmax form the bounding box of the spline. We must not attempt to
% evaluate the spline outside this box.
pmin = [min(x); min(y)];
pmax = [max(x); max(y)];
```

```
% Evaluate the approximation at a vector of values.
evaluate_at_vector(coefs, iopts, opts, pmin, pmax);

% Evaluate the approximation on a mesh.
evaluate_on_mesh(coefs, iopts, opts, pmin, pmax);

function [x, y, f, lsminp, lsmaxp, nxcels, nycels] = generate_data()
  % Generates an x and a y vector of n pseudorandom uniformly distributed
  % values on (0,1]. These are passed to the bivariate function of R. Franke
  % to create the data set to fit.  The remaining input data for
  % e02jd are set to suitable values for this problem,
  % as discussed by Davydov and Zeilfelder.

  n = nag_int(100);

  % Initialize the generator to a repeatable sequence
  [state, ifail] = g05kf(nag_int(1), nag_int(0), nag_int(32958));

  % Generate x values
  [state, x, ifail] = g05sa(n, state);

  % Generate y values
  [state, y, ifail] = g05sa(n, state);

  % Ensure that the bounding box stretches all the way to (0,0) and (1,1)
  x(1) = 0;
  y(1) = 0;
  x(n) = 1;
  y(n) = 1;

  f = 0.75*exp(-((9*x(:)-2).^2    + (9*y(:)-2).^2)/4) + ...
      0.75*exp(-((9*x(:)+1).^2/49 + (9*y(:)+1)/10))   + ...
      0.50*exp(-((9*x(:)-7).^2    + (9*y(:)-3).^2)/4) - ...
      0.20*exp(-((9*x(:)-4).^2    + (9*y(:)-7).^2));

  % Grid size for the approximation
  nxcels = nag_int(6);
  nycels = nag_int(6);

  % Identify the computation.
  fprintf(['\nComputing the coefficients of a C^1 spline',...
           ' approximation to Franke''s function\n']);
  fprintf(' Using a %d by %d grid\n', nxcels, nycels);

  % Local-approximation control parameters.
  lsminp = nag_int(3);
  lsmaxp = nag_int(100);

function [iopts, opts] = handle_options()
  % Initialize the options arrays and demonstrate how to set and get
  % optional parameters.
  opts  = zeros(100, 1);
  iopts = zeros(100, 1, nag_int_name);

  [iopts, opts, ifail] = e02zk( ...
                               'Initialize = e02jd', iopts, opts);

  %  Set some non-default parameters for the local approximation method.
  optstr = strcat('Minimum Singular Value LPA = ', num2str(1/32));
  [iopts, opts, ifail] = e02zk( ...
                               optstr, iopts, opts);
  [iopts, opts, ifail] = e02zk( ...
                               'Polynomial Starting Degree = 3', iopts, opts);

  % Set a non-default parameter for the global approximation method.
  [iopts, opts, ifail] = e02zk( ...
                               'Averaged Spline = Yes', iopts, opts);

  % As an example of how to get the value of an optional parameter,
  % display whether averaging of local approximations is in operation.
```

```
  [~, ~, cvalue, ~, ifail] = e02zl( ...
                                    'Averaged Spline', iopts, opts);
  if strcmp(cvalue, 'YES')
    fprintf(' Using an averaged local approximation\n');
  end

function evaluate_at_vector(coefs, iopts, opts, pmin, pmax)
  % Evaluates the approximation at a (in this case trivial) vector of values.

  xevalv = [0];
  yevalv = [0];

  % Force the points to be within the bounding box of the spline
  for i = 1:numel(xevalv)
    xevalv(i) = max(xevalv(i),pmin(1));
    xevalv(i) = min(xevalv(i),pmax(1));
    yevalv(i) = max(yevalv(i),pmin(2));
    yevalv(i) = min(yevalv(i),pmax(2));
  end

  [fevalv, ifail] = e02je(xevalv, yevalv, coefs, iopts, opts);

  fprintf('\n Values of computed spline at (x_i,y_i):\n\n');
  fprintf('         x_i          y_i   f(x_i,y_i)\n');
  for i = 1:numel(xevalv)
    fprintf('%12.2f %12.2f %12.2f\n', xevalv(i),yevalv(i),fevalv(i));
  end

function evaluate_on_mesh(coefs,iopts,opts,pmin,pmax)
  % Evaluates the approximation on a mesh of n_x * n_y values.
  nxeval = 101;
  nyeval = 101;

  % Define the mesh by its lower-left and upper-right corners.
  ll_corner = [0; 0];
  ur_corner = [1; 1];

  % Set the mesh spacing and the evaluation points.
  % Force the points to be within the bounding box of the spline.
  h = [(ur_corner(1)-ll_corner(1))/(nxeval-1); ...
       (ur_corner(2)-ll_corner(2))/(nyeval-1)];

  xevalm = ll_corner(1) + [0:nxeval-1]*h(1);
  yevalm = ll_corner(2) + [0:nyeval-1]*h(2);

  % Ensure that the evaluation points are in the bounding box
  xevalm = max(pmin(1), xevalm);
  xevalm = min(pmax(1), xevalm);
  yevalm = max(pmin(2), yevalm);
  yevalm = min(pmax(2), yevalm);

  % Evaluate
  [fevalm, ifail] = e02jf(xevalm, yevalm, coefs, iopts, opts);

  print_mesh = false;

  if print_mesh
    fprintf('\nValues of computed spline at (x_i,y_j):\n\n');
    fprintf('         x_i          y_i   f(x_i,y_i)\n');
    for i = 1:nxeval
      for j=1:nyeval
        fprintf('%12.2f %12.2f %12.2f\n', xevalm(i),yevalm(j),fevalm(i, j));
      end
    end
  else
    fprintf('\nOutputting of the function values on the mesh is disabled\n');
  end

  fig1 = figure;
  meshc(yevalm,xevalm,fevalm);
  title({'Bivariate spline fit from scattered data', ...
```

```
        'using two-stage approximation'});
  xlabel('x');
  ylabel('y');
  view(22,28);
  % print(fig1,'-dpng','-r75','e02je_fig1.png');
  % print(fig1,'-deps','-r75','e02je_fig1.eps');
```

9.2 Program Results

```
     e02je example results

Computing the coefficients of a C^1 spline approximation to Franke's function
 Using a 6 by 6 grid
 Using an averaged local approximation

 Values of computed spline at (x_i,y_i):

        x_i          y_i   f(x_i,y_i)
       0.00         0.00         0.76

Outputting of the function values on the mesh is disabled
```