# NAG Toolbox

# nag_fit_2dspline_ts_sctr (e02jd)

## 1 Purpose

nag_fit_2dspline_ts_sctr (e02jd) computes a spline approximation to a set of scattered data using a two-stage approximation method.

The computational complexity of the method grows linearly with the number of data points; hence large datasets are easily accommodated.

## 2 Syntax

```
[coefs, iopts, opts, ifail] = nag_fit_2dspline_ts_sctr(x, y, f, lsminp, lsmaxp,
nxcels, nycels, iopts, opts, 'n', n)
[coefs, iopts, opts, ifail] = e02jd(x, y, f, lsminp, lsmaxp, nxcels, nycels,
iopts, opts, 'n', n)
```

Before calling nag_fit_2dspline_ts_sctr (e02jd), nag_fit_opt_set (e02zk) must be called with **optstr** set to `"Initialize = nag_fit_2dspline_ts_sctr (e02jd)"`. Settings for optional algorithmic arguments may be specified by calling nag_fit_opt_set (e02zk) before a call to nag_fit_2dspline_ts_sctr (e02jd).

## 3 Description

nag_fit_2dspline_ts_sctr (e02jd) determines a smooth bivariate spline approximation to a set of data points $(x_i, y_i, f_i)$, for $i = 1, 2, \ldots, n$. Here, 'smooth' means $C^1$ or $C^2$. (You may select the degree of smoothing using the optional parameter **Global Smoothing Level**.)

The approximation domain is the bounding box $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, where $x_{\min}$ (respectively $y_{\min}$) and $x_{\max}$ (respectively $y_{\max}$) denote the lowest and highest data values of the $(x_i)$ (respectively $(y_i)$).

The spline is computed by local approximations on a uniform triangulation of the bounding box. These approximations are extended to a smooth spline representation of the surface over the domain. The local approximation scheme is controlled by the optional parameter **Local Method**. The schemes provided are: by least squares polynomial approximation (Davydov and Zeilfelder (2004)); by hybrid polynomial and radial basis function (RBF) approximation (Davydov *et al.* (2006)); or by pure RBF approximation (Davydov *et al.* (2005)).

The two-stage approximation method employed by nag_fit_2dspline_ts_sctr (e02jd) is derived from the TSFIT package of O. Davydov and F. Zeilfelder.

Values of the computed spline can subsequently be computed by calling nag_fit_2dspline_ts_evalv (e02je) or nag_fit_2dspline_ts_evalm (e02jf).

## 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

# 5 Parameters

## 5.1 Compulsory Input Parameters

1: **x**(**n**) – REAL (KIND=nag_wp) array
2: **y**(**n**) – REAL (KIND=nag_wp) array
3: **f**(**n**) – REAL (KIND=nag_wp) array

The $(x_i, y_i, f_i)$ data values to be fitted.

*Constraint*: $\mathbf{x}(j) \neq \mathbf{x}(1)$ for some $j = 2, \ldots, n$ and $\mathbf{y}(k) \neq \mathbf{y}(1)$ for some $k = 2, \ldots, n$; i.e., there are at least two distinct $x$ and $y$ values.

4: **lsminp** – INTEGER
5: **lsmaxp** – INTEGER

Are control parameters for the local approximations.

Each local approximation is computed on a local domain containing one of the triangles in the discretization of the bounding box. The size of each local domain will be adaptively chosen such that if it contains fewer than **lsminp** sample points it is expanded, else if it contains greater than **lsmaxp** sample points a thinning method is applied. **lsmaxp** mainly controls computational cost (in that working with a thinned set of points is cheaper and may be appropriate if the input data is densely distributed), while **lsminp** allows handling of different types of scattered data.

Setting **lsmaxp** $<$ **lsminp**, and therefore forcing either expansion or thinning, may be useful for computing initial coarse approximations. In general smaller values for these arguments reduces cost.

A calibration procedure (experimenting with a small subset of the data to be fitted and validating the results) may be needed to choose the most appropriate values for **lsminp** and **lsmaxp**.

*Constraints*:

$1 \leq$ **lsminp** $\leq$ **n**;
**lsmaxp** $\geq 1$.

6: **nxcels** – INTEGER
7: **nycels** – INTEGER

**nxcels** (respectively **nycels**) is the number of cells in the $x$ (respectively $y$) direction that will be used to create the triangulation of the bounding box of the domain of the function to be fitted.

Greater efficiency generally comes when **nxcels** and **nycels** are chosen to be of the same order of magnitude and are such that **n** is $O($**nxcels** $\times$ **nycels**$)$. Thus for a 'square' triangulation — when **nxcels** $=$ **nycels** — the quantities $\sqrt{\mathbf{n}}$ and **nxcels** should be of the same order of magnitude. See also Section 9.

*Constraints*:

**nxcels** $\geq 1$;
**nycels** $\geq 1$.

8: **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 **iopts must not** be modified in any way either directly or indirectly, by further calls to nag_fit_opt_set (e02zk), before calling either or both of the evaluation routines nag_fit_2dspline_ts_evalv (e02je) and nag_fit_2dspline_ts_evalm (e02jf).

9: **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 **opts must not** be modified in any way either directly or indirectly, by further calls to nag_fit_opt_set (e02zk), before calling either or both of the evaluation routines nag_fit_2dspline_ts_evalv (e02je) and nag_fit_2dspline_ts_evalm (e02jf).

### 5.2 Optional Input Parameters

1: **n** – INTEGER

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

$n$, the number of data values to be fitted.

*Constraint*: **n** $> 1$.

### 5.3 Output Parameters

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

If **ifail** $= 0$ on exit, **coefs** contains the computed spline coefficients.

2: **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).

3: **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).

4: **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: **n** $> 1$.

**ifail** $= 4$

Constraint: $1 \leq$ **lsminp** $\leq$ **n**.

**ifail** $= 5$

Constraint: **lsmaxp** $\geq 1$.

**ifail** $= 6$

Constraint: **nxcels** $\geq 1$.

**ifail** $= 7$

Constraint: **nycels** $\geq 1$.

**ifail** $= 8$

Constraint:
if **Global Smoothing Level** $= 1$,
$lcoefs \geq (((\mathbf{nxcels}+2)\times(\mathbf{nycels}+2)+1)/2)\times 10+1$;
if **Global Smoothing Level** $= 2$,
$lcoefs \geq 28\times(\mathbf{nxcels}+2)\times(\mathbf{nycels}+2)\times 4+1$.

**ifail** $= 9$

Option arrays are not initialized or are corrupted.

**ifail** $= 11$

An unexpected algorithmic failure was encountered. Please contact NAG.

**ifail** $= 12$

On entry, all elements of **x** or of **y** are equal.

**ifail** $= 20$

The selected radial basis function cannot be used with the RBF local method.

**ifail** $= 21$

The value of optional parameter **Polynomial Starting Degree** was invalid.

**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

Technical results on error bounds can be found in Davydov and Zeilfelder (2004), Davydov *et al.* (2006) and Davydov *et al.* (2005).

Local approximation by polynomials of degree $d$ for $n$ data points has optimal approximation order $n^{-(d+1)/2}$. The improved approximation power of hybrid polynomial/RBF and of pure RBF approximations is shown in Davydov *et al.* (2006) and Davydov *et al.* (2005).

The approximation error for $C^1$ global smoothing is $O\left(n^{-2}\right)$. For $C^2$ smoothing the error is $O\left(n^{-7/2}\right)$ when **Supersmooth C2** $=$ YES and $O\left(n^{-3}\right)$ when **Supersmooth C2** $=$ NO.

Whether maximal accuracy is achieved depends on the distribution of the input data and the choices of the algorithmic parameters. The references above contain extensive numerical tests and further technical discussions of how best to configure the method.

## 8 Further Comments

$n$-linear complexity and memory usage can be attained for sufficiently dense input data if the triangulation parameters **nxcels** and **nycels** are chosen as recommended in their descriptions above. For sparse input data on such triangulations, if many expansion steps are required (see **lsminp**) the complexity may rise to be loglinear.

Parts of the pure RBF method used when **Local Method** $=$ RBF have $n$-quadratic memory usage.

Note that if **Local Method** = HYBRID and an initial hybrid approximation is deemed unreliable (see the description of optional parameter **Minimum Singular Value LHA**), a pure polynomial approximation will be used instead on that local domain.

# 9 Example

The Franke function

$$\begin{aligned} f(x,y) &= 0.75\exp\Big(-\Big((9x-2)^2+(9y-2)^2\Big)/4\Big)+ \\ &\quad 0.75\exp\Big(-(9x+1)^2/49-(9y+1)/10\Big)+ \\ &\quad 0.5\exp\Big(-\Big((9x-7)^2+(9y-3)^2\Big)/4\Big)- \\ &\quad 0.2\exp\Big(-(9x-4)^2-(9y-7)^2\Big) \end{aligned}$$

is widely used for testing surface-fitting methods. The example program randomly generates a number of points on this surface. From these a spline is computed and then evaluated at a vector of points and on a mesh.

## 9.1 Program Text

```
function e02jd_example

fprintf('e02jd example results\n\n');

npts  = 15;
xdata = [ 0.0;  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(npts + 4);
ixloc = zeros(numel(x), 1, nag_int_name);
wrk   = zeros(4*npts + 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 random vectors x, y.
  % These are used by bivariate function of R. Franke to create data set.
  % The remaining input data 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 and y values
  [state, x, ifail] = g05sa(n, state);
  [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','e02jd_fig1.png');
  % print(fig1,'-deps','-r75','e02jd_fig1.eps');
```

### 9.2 Program Results

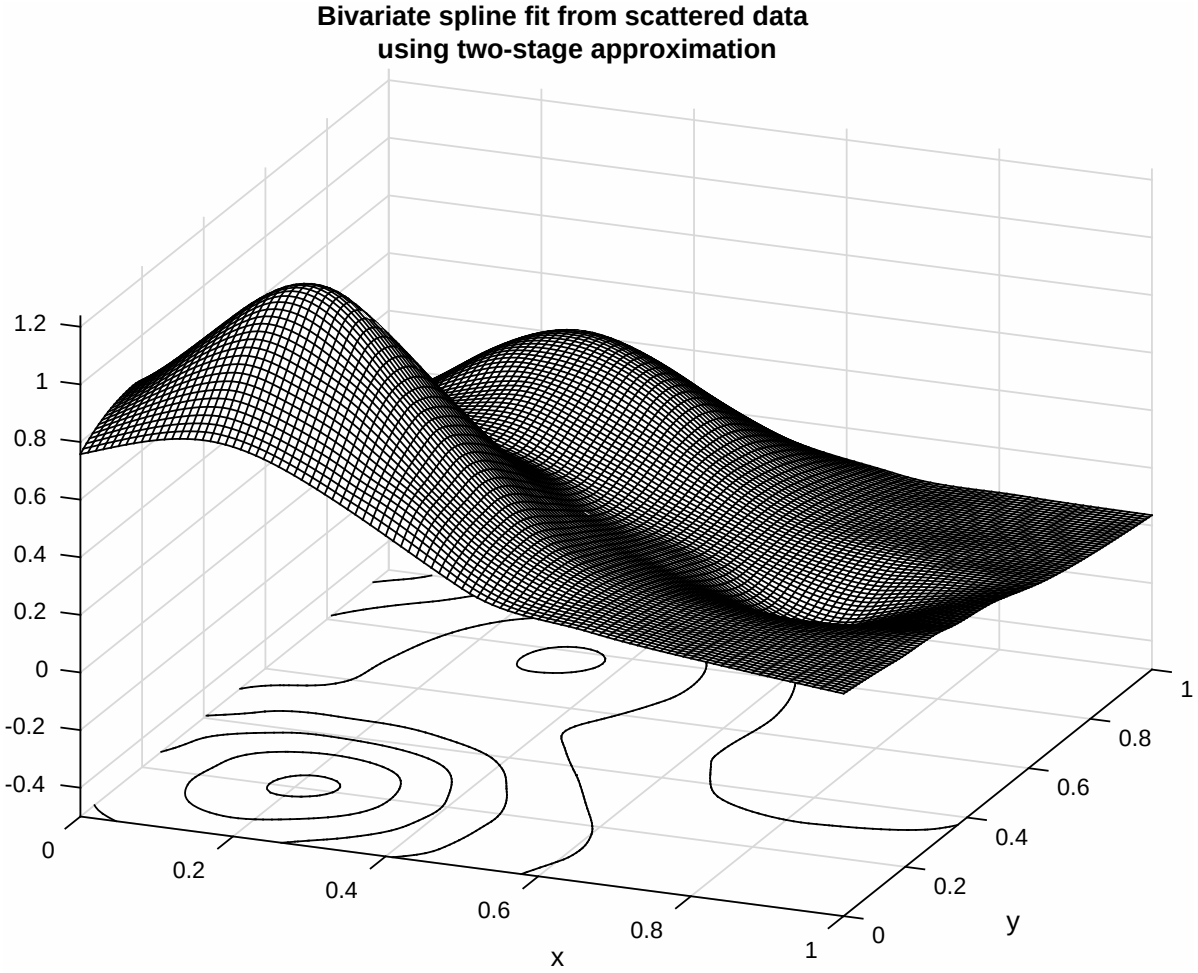
```
     e02jd example results

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
```




## 10 Optional Parameters

Several optional parameters in nag_fit_2dspline_ts_sctr (e02jd) control aspects of the algorithm, methodology used, logic or output. Their values are contained in the arrays **iopts** and **opts**; these must be initialized before calling nag_fit_2dspline_ts_sctr (e02jd) by first calling nag_fit_opt_set (e02zk) with **optstr** set to `"Initialize = nag_fit_2dspline_ts_sctr (e02jd)"`.

Each optional parameter has an associated default value; to set any of them to a non-default value, or to reset any of them to the default value, use nag_fit_opt_set (e02zk). The current value of an optional parameter can be queried using nag_fit_opt_get (e02zl).

The remainder of this section can be skipped if you wish to use the default values for all optional parameters.

The following is a list of the optional parameters available. A full description of each optional parameter is provided in Section 11.1.

**Averaged Spline**
**Global Smoothing Level**
**Interpolation Only RBF**
**Local Method**
**Minimum Singular Value LHA**
**Minimum Singular Value LPA**
**Polynomial Starting Degree**
**Radial Basis Function**
**Scaling Coefficient RBF**
**Separation LRBFA**
**Supersmooth C2**

## 10.1 Description of the Optional Parameters

For each option, we give a summary line, a description of the optional parameter and details of constraints.

The summary line contains:

the keywords;

a parameter value, where the letters $a$, $i$ and $r$ denote options that take character, integer and real values respectively;

the default value.

Keywords and character values are case insensitive.

For nag_fit_2dspline_ts_sctr (e02jd) the maximum length of the parameter **cvalue** used by nag_fit_opt_get (e02zl) is 16.

**Averaged Spline** $a$ Default $=$ NO

When the bounding box is triangulated there are 8 equivalent configurations of the mesh. Setting **Averaged Spline** $=$ YES will use the averaged value of the 8 possible local polynomial approximations over each triangle in the mesh. This usually gives better results but at (about 8 times) higher computational cost.

Constraint: **Averaged Spline** $=$ YES or NO.

**Global Smoothing Level** $i$ Default $= 1$

The smoothness level for the global spline approximation.

**Global Smoothing Level** $= 1$
Will use $C^1$ piecewise cubics.

**Global Smoothing Level** $= 2$
Will use $C^2$ piecewise sextics.

*Constraint*: **Global Smoothing Level** $= 1$ or 2.

**Interpolation Only RBF** $a$ Default $=$ YES

If **Interpolation Only RBF** $=$ YES, each local RBF approximation is computed by interpolation.

If **Interpolation Only RBF** $=$ NO, each local RBF approximation is computed by a discrete least squares approach. This is likely to be more accurate and more expensive than interpolation.

If **Local Method** $=$ HYBRID or POLYNOMIAL, this option setting is ignored.

*Constraint*: **Interpolation Only RBF** $=$ YES or NO.

**Local Method** $a$ Default $=$ POLYNOMIAL

The local approximation scheme to use.

**Local Method** $=$ POLYNOMIAL
Uses least squares polynomial approximations.

**Local Method** $=$ HYBRID
Uses hybrid polynomial and RBF approximations.

**Local Method** $=$ RBF
Uses pure RBF approximations.

In general POLYNOMIAL is less computationally expensive than HYBRID is less computationally expensive than RBF with the reverse ordering holding for accuracy of results.

*Constraint*: **Local Method** $=$ POLYNOMIAL, HYBRID or RBF.

**Minimum Singular Value LHA** $r$ Default $= 1.0$

A tolerance measure for accepting or rejecting a local hybrid approximation (LHA) as reliable.

The solution of a local least squares problem solved on each triangle subdomain is accepted as reliable if the minimum singular value $\sigma$ of the collocation matrix (of polynomial and radial basis function terms) associated with the least squares problem satisfies **Minimum Singular Value LHA** $\leq \sigma$.

In general the approximation power will be reduced as **Minimum Singular Value LHA** is reduced. (A small $\sigma$ indicates that the local data has hidden redundancies which prevent it from carrying enough information for a good approximation to be made.) Setting **Minimum Singular Value LHA** very large may have the detrimental effect that only approximations of low degree are deemed reliable.

A calibration procedure (experimenting with a small subset of the data to be fitted and validating the results) may be needed to choose the most appropriate value for this parameter.

If **Local Method** $=$ POLYNOMIAL or RBF, this option setting is ignored.

*Constraint*: **Minimum Singular Value LHA** $\geq 0.0$.

**Minimum Singular Value LPA** $r$ Default $= 1.0$

A tolerance measure for accepting or rejecting a local polynomial approximation (LPA) as reliable. Clearly this setting is relevant when **Local Method** $=$ POLYNOMIAL, but it also may be used when **Local Method** $=$ HYBRID (see Section 9.)

The solution of a local least squares problem solved on each triangle subdomain is accepted as reliable if the minimum singular value $\sigma$ of the matrix (of Bernstein polynomial values) associated with the least squares problem satisfies **Minimum Singular Value LPA** $\leq \sigma$.

In general the approximation power will be reduced as **Minimum Singular Value LPA** is reduced. (A small $\sigma$ indicates that the local data has hidden redundancies which prevent it from carrying enough information for a good approximation to be made.) Setting **Minimum Singular Value LPA** very large may have the detrimental effect that only approximations of low degree are deemed reliable.

**Minimum Singular Value LPA** will have no effect if **Polynomial Starting Degree** $= 0$, and it will have little effect if the input data is 'smooth' (e.g., from a known function).

A calibration procedure (experimenting with a small subset of the data to be fitted and validating the results) may be needed to choose the most appropriate value for this parameter.

If **Local Method** $=$ RBF, this option setting is ignored.

*Constraint*: **Minimum Singular Value LPA** $\geq 0.0$.

**Polynomial Starting Degree** $i$ Default $= 5$ if **Local Method** $=$ HYBRID, Default $= 1$ otherwise

The degree to be used for the polynomial part in the initial step of each local approximation.

At this initial step the method will attempt to fit with a local approximation having polynomial part of degree **Polynomial Starting Degree**. If **Local Method** = POLYNOMIAL and the approximation is deemed unreliable (according to **Minimum Singular Value LPA**), the degree will be decremented by one and a new local approximation computed, ending with a constant approximation if no other is reliable. If **Local Method** = HYBRID and the approximation is deemed unreliable (according to **Minimum Singular Value LHA**), a pure polynomial approximation of this degree will be tried instead. The method then proceeds as in the POLYNOMIAL case.

**Polynomial Starting Degree** is bounded from above by the maximum possible spline degree, which is 6 (when performing $C^2$ global super-smoothing). Note that the best-case approximation error (see Section 7) for $C^2$ smoothing with **Supersmooth C2** = NO is achieved for local polynomials of degree 5; that is, for this level of global smoothing no further benefit is gained by setting **Polynomial Starting Degree** = 6.

The default value gives a good compromise between efficiency and accuracy. In general the best approximation can be obtained by setting:

If **Local Method** = POLYNOMIAL

if **Global Smoothing Level** = 1, **Polynomial Starting Degree** = 3;

if **Global Smoothing Level** = 2;

if **Supersmooth C2** = NO, **Polynomial Starting Degree** = 5;

otherwise **Polynomial Starting Degree** = 6.

If **Local Method** = HYBRID, **Polynomial Starting Degree** as small as possible.

If **Local Method** = RBF, this option setting is ignored.

*Constraints*:

if **Local Method** = HYBRID,

if **Radial Basis Function** = MQ2, MQ3, TPS or POLYHARMONIC3,
**Polynomial Starting Degree** $\geq 1$;
if **Radial Basis Function** = TPS4 or POLYHARMONIC5,
**Polynomial Starting Degree** $\geq 2$;
if **Radial Basis Function** = TPS6 or POLYHARMONIC7,
**Polynomial Starting Degree** $\geq 3$;
if **Radial Basis Function** = POLYHARMONIC9,
**Polynomial Starting Degree** $\geq 4$.;
otherwise **Polynomial Starting Degree** $\geq 0$;
if **Local Method** = POLYNOMIAL and **Global Smoothing Level** = 1,
**Polynomial Starting Degree** $\leq 3$;
otherwise **Polynomial Starting Degree** $\leq 6$.

| | | |
|---|---|---|
| **Radial Basis Function** | $a$ | Default = MQ |
| **Scaling Coefficient RBF** | $r$ | Default = 1.0 |

**Radial Basis Function** selects the RBF to use in each local RBF approximation, while **Scaling Coefficient RBF** selects the scale factor to use in its evaluation, as described below.

A calibration procedure (experimenting with a small subset of the data to be fitted and validating the results) may be needed to choose the most appropriate scale factor and RBF.

If **Local Method** = POLYNOMIAL, these option settings are ignored.

If **Local Method** = HYBRID or RBF, the following (conditionally) positive definite functions may be chosen.

Define $R = \sqrt{x^2 + y^2}$ and $\rho = R/r$.

| | |
|---|---|
| GAUSS | Gaussian $\exp(-\rho^2)$ |
| IMQ | inverse multiquadric $1/\sqrt{r^2 + R^2}$ |

| | |
|---|---|
| IMQ2 | inverse multiquadric $1/\left(r^2+R^2\right)$ |
| IMQ3 | inverse multiquadric $1/\left(r^2+R^2\right)^{(3/2)}$ |
| IMQ0_5 | inverse multiquadric $1/\left(r^2+R^2\right)^{(1/4)}$ |
| WENDLAND31 | H. Wendland's $C^2$ function $\max(0,1-\rho)^4(4\rho+1)$ |
| WENDLAND32 | H. Wendland's $C^4$ function $\max(0,1-\rho)^6\left(35\rho^2+18\rho+3\right)$ |
| WENDLAND33 | H. Wendland's $C^6$ function $\max(0,1-\rho)^8\left(32\rho^3+25\rho^2+8\rho+1\right)$ |
| BUHMANNC3 | M. Buhmann's $C^3$ function $112/45\rho^{(9/2)}+16/3\rho^{(7/2)}-7\rho^4-14/15\rho^2+1/9$ if $\rho\leq 1$, 0 otherwise |
| MQ | multiquadric $\sqrt{r^2+R^2}$ |
| MQ1_5 | multiquadric $\left(r^2+R^2\right)^{(1.5/2)}$ |
| POLYHARMONIC1_5 | polyharmonic spline $\rho^{1.5}$ |
| POLYHARMONIC1_75 | polyharmonic spline $\rho^{1.75}$ |

If **Local Method** = HYBRID the following conditionally positive definite functions may also be chosen.

| | |
|---|---|
| MQ2 | multiquadric $\left(r^2+R^2\right)\log\left(r^2+R^2\right)$ |
| MQ3 | multiquadric $\left(r^2+R^2\right)^{(3/2)}$ |
| TPS | thin plate spline $\rho^2\log\rho^2$ |
| POLYHARMONIC3 | polyharmonic spline $\rho^3$ |
| TPS4 | thin plate spline $\rho^4\log\rho^2$ |
| POLYHARMONIC5 | polyharmonic spline $\rho^5$ |
| TPS6 | thin plate spline $\rho^6\log\rho^2$ |
| POLYHARMONIC7 | polyharmonic spline $\rho^7$ |
| POLYHARMONIC9 | polyharmonic spline $\rho^9$ |

*Constraints*:

if **Radial Basis Function** = MQ2, MQ3, TPS or POLYHARMONIC3,
**Local Method** = HYBRID and **Polynomial Starting Degree** $\geq 1$;
if **Radial Basis Function** = TPS4 or POLYHARMONIC5,
**Local Method** = HYBRID and **Polynomial Starting Degree** $\geq 2$;
if **Radial Basis Function** = TPS6 or POLYHARMONIC7,
**Local Method** = HYBRID and **Polynomial Starting Degree** $\geq 3$;
if **Radial Basis Function** = POLYHARMONIC9,
**Local Method** = HYBRID and **Polynomial Starting Degree** $\geq 4$;
**Scaling Coefficient RBF** $> 0.0$.

**Separation LRBFA** $r$ Default $= 16.0/$**Scaling Coefficient RBF**

A knot-separation parameter used to control the condition number of the matrix used in each local RBF approximation (LRBFA). A smaller value may mean greater numerical stability but fewer knots.

If **Local Method** = HYBRID or POLYNOMIAL, this option setting is ignored.

*Constraint*: **Separation LRBFA** $> 0.0$.

**Supersmooth C2** $a$ Default $=$ NO

If **Supersmooth C2** $=$ YES, the $C^2$ spline is generated using additional smoothness constraints. This usually gives better results but at higher computational cost.

If **Global Smoothing Level** $= 1$ this option setting is ignored.

*Constraint*: **Supersmooth C2** $=$ YES or NO.

---