NAG Toolbox Chapter Introduction

E02 – Curve and Surface Fitting

Contents

1 Scope of the Chapter

The main aim of this chapter is to assist you in finding a function which approximates a set of data points. Typically the data contain random errors, as of experimental measurement, which need to be smoothed out. To seek an approximation to the data, it is first necessary to specify for the approximating function a mathematical form (a polynomial, for example) which contains a number of unspecified coefficients: the appropriate fitting function then derives for the coefficients the values which provide the best fit of that particular form. The chapter deals mainly with curve and surface fitting (i.e., fitting with functions of one and of two variables) when a polynomial or a cubic spline is used as the fitting function, since these cover the most common needs. However, fitting with other functions and/or more variables can be undertaken by means of general linear or nonlinear functions (some of which are contained in other chapters) depending on whether the coefficients in the function occur linearly or nonlinearly. Cases where a graph rather than a set of data points is given can be treated simply by first reading a suitable set of points from the graph.

The chapter also contains functions for evaluating, differentiating and integrating polynomial and spline curves and surfaces, once the numerical values of their coefficients have been determined.

There is also a function for computing a Padé approximant of a mathematical function (see Section 2.6 and Section 3.8).

2 Background to the Problems

2.1 Preliminary Considerations

In the curve-fitting problems considered in this chapter, we have a dependent variable y and an independent variable x, and we are given a set of data points (x_r, y_r), for $r = 1, 2, \ldots, m$. The preliminary matters to be considered in this section will, for simplicity, be discussed in this context of curve-fitting problems. In fact, however, these considerations apply equally well to surface and higher-dimensional problems. Indeed, the discussion presented carries over essentially as it stands if, for these cases, we interpret x as a vector of several independent variables and correspondingly each x_r as a vector containing the rth data value of each independent variable.

We wish, then, to approximate the set of data points as closely as possible with a specified function, $f(x)$ say, which is as smooth as possible: $f(x)$ may, for example, be a polynomial. The requirements of smoothness and closeness conflict, however, and a balance has to be struck between them. Most often, the smoothness requirement is met simply by limiting the number of coefficients allowed in the fitting function – for example, by restricting the degree in the case of a polynomial. Given a particular number of coefficients in the function in question, the fitting functions of this chapter determine the values of the coefficients such that the 'distance' of the function from the data points is as small as possible. The necessary balance is struck when you compare a selection of such fits having different numbers of coefficients. If the number of coefficients is too low, the approximation to the data will be poor. If the number is too high, the fit will be too close to the data, essentially following the random errors and tending to have unwanted fluctuations between the data points. Between these extremes, there is often a group of fits all similarly close to the data points and then, particularly when least squares polynomials are used, the choice is clear: it is the fit from this group having the smallest number of coefficients.

You are in effect minimizing the smoothness measure (i.e., the number of coefficients) subject to the distance from the data points being acceptably small. Some of the functions, however, do this task themselves. They use a different measure of smoothness (in each case one that is continuous) and minimize it subject to the distance being less than a threshold specified by you. This is a much more automatic process, requiring only some experimentation with the threshold.

2.1.1 Fitting criteria: norms

A measure of the above 'distance' between the set of data points and the function $f(x)$ is needed. The distance from a single data point (x_r, y_r) to the function can simply be taken as

$$\epsilon_r = y_r - f(x_r), \tag{1}$$

and is called the **residual** of the point. (With this definition, the residual is regarded as a function of the coefficients contained in $f(x)$; however, the term is also used to mean the particular value of ϵ_r which

corresponds to the fitted values of the coefficients.) However, we need a measure of distance for the set of data points as a whole. Three different measures are used in the different functions (which measure to select, according to circumstances, is discussed later in this sub-section). With ϵ_r defined in (1), these measures, or **norms**, are

$$\sum_{r=1}^{m}|\epsilon_r|, \tag{2}$$

$$\sqrt{\sum_{r=1}^{m}\epsilon_r^2}, \tag{3}$$

and

$$\max_r|\epsilon_r|, \tag{4}$$

respectively the ℓ_1 norm, the ℓ_2 norm and the ℓ_∞ norm.

Minimization of one or other of these norms usually provides the fitting criterion, the minimization being carried out with respect to the coefficients in the mathematical form used for $f(x)$: with respect to the b_i for example if the mathematical form is the power series in (8) below. The fit which results from minimizing (2) is known as the ℓ_1 fit, or the fit in the ℓ_1 norm: that which results from minimizing (3) is the ℓ_2 fit, the well-known least squares fit (minimizing (3) is equivalent to minimizing the square of (3), i.e., the sum of squares of residuals, and it is the latter which is used in practice), and that from minimizing (4) is the ℓ_∞, or minimax, fit.

Strictly speaking, implicit in the use of the above norms are the statistical assumptions that the random errors in the y_r are independent of one another and that any errors in the x_r are negligible by comparison. From this point of view, the use of the ℓ_2 norm is appropriate when the random errors in the y_r have a Normal distribution, and the ℓ_∞ norm is appropriate when they have a rectangular distribution, as when fitting a table of values rounded to a fixed number of decimal places. The ℓ_1 norm is appropriate when the error distribution has its frequency function proportional to the negative exponential of the modulus of the normalized error – not a common situation.

However, it may be that you are indifferent to these statistical considerations, and simply seek a fit which can be assessed by inspection, perhaps visually from a graph of the results. In this event, the ℓ_1 norm is particularly appropriate when the data are thought to contain some 'wild' points (since fitting in this norm tends to be unaffected by the presence of a small number of such points), though of course in simple situations you may prefer to identify and reject these points. The ℓ_∞ norm should be used only when the maximum residual is of particular concern, as may be the case for example when the data values have been obtained by accurate computation, as of a mathematical function. Generally, however, a function based on least squares should be preferred, as being computationally faster and usually providing more information on which to assess the results. In many problems the three fits will not differ significantly for practical purposes.

Some of the functions based on the ℓ_2 norm do not minimize the norm itself but instead minimize some (intuitively acceptable) measure of smoothness subject to the norm being less than a user-specified threshold. These functions fit with cubic or bicubic splines (see (10) and (14) below) and the smoothing measures relate to the size of the discontinuities in their third derivatives. A much more automatic fitting procedure follows from this approach.

2.1.2 Weighting of data points

The use of the above norms also assumes that the data values y_r are of equal (absolute) accuracy. Some of the functions enable an allowance to be made to take account of differing accuracies. The allowance takes the form of 'weights' applied to the y values so that those values known to be more accurate have a greater influence on the fit than others. These weights, to be supplied by you, should be calculated from estimates of the absolute accuracies of the y values, these estimates being expressed as standard deviations, probable errors or some other measure which has the same dimensions as y. Specifically, for each y_r the corresponding weight w_r should be inversely proportional to the accuracy estimate of y_r. For example, if the percentage accuracy is the same for all y_r, then the absolute accuracy of y_r is proportional to y_r (assuming y_r to be positive, as it usually is in such cases) and so $w_r = K/y_r$, for

$r = 1, 2, \ldots, m$, for an arbitrary positive constant K. (This definition of weight is stressed because often weight is defined as the square of that used here.) The norms (2), (3) and (4) above are then replaced respectively by

$$\sum_{r=1}^{m} |w_r \epsilon_r|, \tag{5}$$

$$\sqrt{\sum_{r=1}^{m} w_r^2 \epsilon_r^2}, \tag{6}$$

and

$$\max_r |w_r \epsilon_r|. \tag{7}$$

Again it is the square of (6) which is used in practice rather than (6) itself.

2.2 Curve Fitting

When, as is commonly the case, the mathematical form of the fitting function is immaterial to the problem, polynomials and cubic splines are to be preferred because their simplicity and ease of handling confer substantial benefits. The **cubic spline** is the more versatile of the two. It consists of a number of cubic polynomial segments joined end to end with continuity in first and second derivatives at the joins. The third derivative at the joins is in general discontinuous. The x values of the joins are called **knots**, or, more precisely, interior knots. Their number determines the number of coefficients in the spline, just as the degree determines the number of coefficients in a polynomial.

2.2.1 Representation of polynomials

Two different forms for representing a polynomial are used in different functions. One is the usual power-series form

$$f(x) \equiv b_0 + b_1 x + b_2 x^2 + \cdots + b_k x^k. \tag{8}$$

The other is the Chebyshev series form

$$f(x) \equiv \tfrac{1}{2} a_0 T_0(x) + a_1 T_1(x) + a_2 T_2(x) + \cdots + a_k T_k(x), \tag{9}$$

where $T_i(x)$ is the Chebyshev polynomial of the first kind of degree i (see page 9 of Cox and Hayes (1973)), and where the range of x has been normalized to run from -1 to $+1$. The use of either form leads theoretically to the same fitted polynomial, but in practice results may differ substantially because of the effects of rounding error. The Chebyshev form is to be preferred, since it leads to much better accuracy in general, both in the computation of the coefficients and in the subsequent evaluation of the fitted polynomial at specified points. This form also has other advantages: for example, since the later terms in (9) generally decrease much more rapidly from left to right than do those in (8), the situation is more often encountered where the last terms are negligible and it is obvious that the degree of the polynomial can be reduced (note that on the interval $-1 \leq x \leq 1$ for all i, $T_i(x)$ attains the value unity but never exceeds it, so that the coefficient a_i gives directly the maximum value of the term containing it). If the power-series form is used it is most advisable to work with the variable x normalized to the range -1 to $+1$, carrying out the normalization before entering the relevant function. This will often substantially improve computational accuracy.

2.2.2 Representation of cubic splines

A cubic spline is represented in the form

$$f(x) \equiv c_1 N_1(x) + c_2 N_2(x) + \cdots + c_p N_p(x), \tag{10}$$

where $N_i(x)$, for $i = 1, 2, \ldots, p$, is a normalized cubic B-spline (see Hayes (1974)). This form, also, has advantages of computational speed and accuracy over alternative representations.

2.3 Surface Fitting

There are now two independent variables, and we shall denote these by x and y. The dependent variable, which was denoted by y in the curve-fitting case, will now be denoted by f. (This is a rather different notation from that indicated for the general-dimensional problem in the first paragraph of Section 2.1, but it has some advantages in presentation.)

Again, in the absence of contrary indications in the particular application being considered, polynomials and splines are the approximating functions most commonly used.

2.3.1 Representation of bivariate polynomials

The type of bivariate polynomial currently considered in the chapter can be represented in either of the two forms

$$f(x,y) \equiv \sum_{i=0}^{k} \sum_{j=0}^{\ell} b_{ij} x^i y^j, \tag{11}$$

and

$$f(x,y) \equiv \sum_{i=0}^{k} \sum_{j=0}^{\ell} a_{ij} T_i(x) T_j(y), \tag{12}$$

where $T_i(x)$ is the Chebyshev polynomial of the first kind of degree i in the parameter x (see page 9 of Cox and Hayes (1973)), and correspondingly for $T_j(y)$. The prime on the two summation signs, following standard convention, indicates that the first term in each sum is halved, as shown for one variable in equation (9). The two forms (11) and (12) are mathematically equivalent, but again the Chebyshev form is to be preferred on numerical grounds, as discussed in Section 2.2.1.

2.3.2 Bicubic splines: definition and representation

The bicubic spline is defined over a rectangle R in the (x,y) plane, the sides of R being parallel to the x and y axes. R is divided into rectangular panels, again by lines parallel to the axes. Over each panel the bicubic spline is a bicubic polynomial, that is it takes the form

$$\sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j. \tag{13}$$

Each of these polynomials joins the polynomials in adjacent panels with continuity up to the second derivative. The constant x values of the dividing lines parallel to the y-axis form the set of interior knots for the variable x, corresponding precisely to the set of interior knots of a cubic spline. Similarly, the constant y values of dividing lines parallel to the x-axis form the set of interior knots for the variable y. Instead of representing the bicubic spline in terms of the above set of bicubic polynomials, however, it is represented, for the sake of computational speed and accuracy, in the form

$$f(x,y) = \sum_{i=1}^{p} \sum_{j=1}^{q} c_{ij} M_i(x) N_j(y), \tag{14}$$

where $M_i(x)$, for $i = 1,2,\ldots,p$, and $N_j(y)$, for $j = 1,2,\ldots,q$, are normalized B-splines (see Hayes and Halliday (1974) for further details of bicubic splines and Hayes (1974) for normalized B-splines).

2.4 General Linear and Nonlinear Fitting Functions

We have indicated earlier that, unless the data-fitting application under consideration specifically requires some other type of fit, a polynomial or a spline is usually to be preferred. Special functions for these types, in one and in two variables, are provided in this chapter. When the application does specify some other form of fitting, however, it may be treated by a function which deals with a general linear function, or by one for a general nonlinear function, depending on whether the coefficients occur linearly or nonlinearly.

The general linear fitting function can be written in the form

$$f(x) \equiv c_1\phi_1(x) + c_2\phi_2(x) + \cdots + c_p\phi_p(x), \tag{15}$$

where x is a vector of one or more independent variables, and the ϕ_i are any given functions of these variables (though they must be linearly independent of one another if there is to be the possibility of a unique solution to the fitting problem). This is not intended to imply that each ϕ_i is necessarily a function of all the variables: we may have, for example, that each ϕ_i is a function of a different single variable, and even that one of the ϕ_i is a constant. All that is required is that a value of each $\phi_i(x)$ can be computed when a value of each independent variable is given.

When the fitting function $f(x)$ is not linear in its coefficients, no more specific representation is available in general than $f(x)$ itself. However, we shall find it helpful later on to indicate the fact that $f(x)$ contains a number of coefficients (to be determined by the fitting process) by using instead the notation $f(x;c)$, where c denotes the vector of coefficients. An example of a nonlinear fitting function is

$$f(x;c) \equiv c_1 + c_2\exp(-c_4x) + c_3\exp(-c_5x), \tag{16}$$

which is in one variable and contains five coefficients. Note that here, as elsewhere in this Chapter Introduction, we use the term 'coefficients' to include all the quantities whose values are to be determined by the fitting process, not just those which occur linearly. We may observe that it is only the presence of the coefficients c_4 and c_5 which makes the form (16) nonlinear. If the values of these two coefficients were known beforehand, (16) would instead be a linear function which, in terms of the general linear form (15), has $p = 3$ and

$$\phi_1(x) \equiv 1, \phi_2(x) \equiv \exp(-c_4x), \text{and } \phi_3(x) \equiv \exp(-c_5x).$$

We may note also that polynomials and splines, such as (9) and (14), are themselves linear in their coefficients. Thus if, when fitting with these functions, a suitable special function is not available (as when more than two independent variables are involved or when fitting in the ℓ_1 norm), it is appropriate to use a function designed for a general linear function.

2.5 Constrained Problems

So far, we have considered only fitting processes in which the values of the coefficients in the fitting function are determined by an unconstrained minimization of a particular norm. Some fitting problems, however, require that further restrictions be placed on the determination of the coefficient values. Sometimes these restrictions are contained explicitly in the formulation of the problem in the form of equalities or inequalities which the coefficients, or some function of them, must satisfy. For example, if the fitting function contains a term $A\exp(-kx)$, it may be required that $k \geq 0$. Often, however, the equality or inequality constraints relate to the value of the fitting function or its derivatives at specified values of the independent variable(s), but these too can be expressed in terms of the coefficients of the fitting function, and it is appropriate to do this if a general linear or nonlinear function is being used. For example, if the fitting function is that given in (10), the requirement that the first derivative of the function at $x = x_0$ be non-negative can be expressed as

$$c_1N_1'(x_0) + c_2N_2'(x_0) + \cdots + c_pN_p'(x_0) \geq 0, \tag{17}$$

where the prime denotes differentiation with respect to x and each derivative is evaluated at $x = x_0$. On the other hand, if the requirement had been that the derivative at $x = x_0$ be exactly zero, the inequality sign in (17) would be replaced by an equality.

Functions which provide a facility for minimizing the appropriate norm subject to such constraints are discussed in Section 3.6.

2.6 Padé Approximants

A Padé approximant to a function $f(x)$ is a rational function (ratio of two polynomials) whose Maclaurin series expansion is the same as that of $f(x)$ up to and including the term in x^k, where k is the sum of the degrees of the numerator and denominator of the approximant. Padé approximation can be a useful technique when values of a function are to be obtained from its Maclaurin series but convergence of the series is unacceptably slow or even nonexistent.

3 Recommendations on Choice and Use of Available Functions

3.1 General

The choice of a function to treat a particular fitting problem will depend first of all on the fitting function and the norm to be used. Unless there is good reason to the contrary, the fitting function should be a polynomial or a cubic spline (in the appropriate number of variables) and the norm should be the ℓ_2 norm (leading to the least squares fit). If some other function is to be used, the choice of function will depend on whether the function is nonlinear (in which case see Section 3.5.2) or linear in its coefficients (see Section 3.5.1), and, in the latter case, on whether the ℓ_1, ℓ_2 or ℓ_∞ norm is to be used. The latter section is appropriate for polynomials and splines, too, if the ℓ_1 or ℓ_∞ norm is preferred, with one exception: there is a special function for fitting polynomial curves in the unweighted ℓ_∞ norm (see Section 3.2.3).

In the case of a polynomial or cubic spline, if there is only one independent variable, you should choose a spline (see Section 3.3) when the curve represented by the data is of complicated form, perhaps with several peaks and troughs. When the curve is of simple form, first try a polynomial (see Section 3.2) of low degree, say up to degree 5 or 6, and then a spline if the polynomial fails to provide a satisfactory fit. (Of course, if third-derivative discontinuities are unacceptable, a polynomial is the only choice.) If the problem is one of surface fitting, the polynomial function (see Section 3.4.1) should be tried first if the data arrangement happens to be appropriate, otherwise one of the spline functions (see Section 3.4.3). If the problem has more than two independent variables, it may be treated by the general linear function in Section 3.5.1, again using a polynomial in the first instance.

Another factor which affects the choice of function is the presence of constraints, as previously discussed on Section 2.5. Indeed this factor is likely to be overriding at present, because of the limited number of functions which have the necessary facility. Consequently those functions have been grouped together for discussion in Section 3.6.

3.1.1 Data considerations

A satisfactory fit cannot be expected by any means if the number and arrangement of the data points do not adequately represent the character of the underlying relationship: sharp changes in behaviour, in particular, such as sharp peaks, should be well covered. Data points should extend over the whole range of interest of the independent variable(s): extrapolation outside the data ranges is most unwise. Then, with polynomials, it is advantageous to have additional points near the ends of the ranges, to counteract the tendency of polynomials to develop fluctuations in these regions. When, with polynomial curves, you can precisely choose the x values of the data, the special points defined in Section 3.2.2 should be selected. With polynomial surfaces, each of these same x values should, where possible, be combined with each of a corresponding set of y values (not necessarily with the same value of n), thus forming a rectangular grid of (x, y)-values. With splines the choice is less critical as long as the character of the relationship is adequately represented. All fits should be tested graphically before accepting them as satisfactory.

For this purpose it should be noted that it is not sufficient to plot the values of the fitted function only at the data values of the independent variable(s); at the least, its values at a similar number of intermediate points should also be plotted, as unwanted fluctuations may otherwise go undetected. Such fluctuations are the less likely to occur the lower the number of coefficients chosen in the fitting function. No firm guide can be given, but as a rough rule, at least initially, the number of coefficients should not exceed half the number of data points (points with equal or nearly equal values of the independent variable, or both independent variables in surface fitting, counting as a single point for this purpose). However, the situation may be such, particularly with a small number of data points, that a satisfactorily close fit to the data cannot be achieved without unwanted fluctuations occurring. In such cases, it is often possible to improve the situation by a transformation of one or more of the variables, as discussed in the next section: otherwise it will be necessary to provide extra data points. Further advice on curve-fitting is given in Cox and Hayes (1973) and, for polynomials only, in Hayes (1970). Much of the advice applies also to surface fitting; see also the function documents.

3.1.2 Transformation of variables

Before starting the fitting, consideration should be given to the choice of a good form in which to deal with each of the variables: often it will be satisfactory to use the variables as they stand, but sometimes the use of the logarithm, square root, or some other function of a variable will lead to a better-behaved relationship. This question is customarily taken into account in preparing graphs and tables of a relationship and the same considerations apply when curve or surface fitting. The practical context will often give a guide. In general, it is best to avoid having to deal with a relationship whose behaviour in one region is radically different from that in another. A steep rise at the left-hand end of a curve, for example, can often best be treated by curve-fitting in terms of $\log(x+c)$ with some suitable value of the constant c. A case when such a transformation gave substantial benefit is discussed in page 60 of Hayes (1970). According to the features exhibited in any particular case, transformation of either dependent variable or independent variable(s) or both may be beneficial. When there is a choice it is usually better to transform the independent variable(s): if the dependent variable is transformed, the weights attached to the data points must be adjusted. Thus (denoting the dependent variable by y, as in the notation for curves) if the y_r to be fitted have been obtained by a transformation $y = g(Y)$ from original data values Y_r, with weights W_r, for $r = 1, 2, \ldots, m$, we must take

$$w_r = W_r/(dy/dY), \tag{18}$$

where the derivative is evaluated at Y_r. Strictly, the transformation of Y and the adjustment of weights are valid only when the data errors in the Y_r are small compared with the range spanned by the Y_r, but this is usually the case.

3.2 Polynomial Curves

3.2.1 Least squares polynomials: arbitrary data points

nag_fit_1dcheb_arb (e02ad) fits to arbitrary data points, with arbitrary weights, polynomials of all degrees up to a maximum degree k, which is a choice. If you are seeking only a low-degree polynomial, up to degree 5 or 6 say, $k = 10$ is an appropriate value, providing there are about 20 data points or more. To assist in deciding the degree of polynomial which satisfactorily fits the data, the function provides the root-mean-square residual s_i for all degrees $i = 1, 2, \ldots, k$. In a satisfactory case, these s_i will decrease steadily as i increases and then settle down to a fairly constant value, as shown in the example

i	s_i
0	3.5215
1	0.7708
2	0.1861
3	0.0820
4	0.0554
5	0.0251
6	0.0264
7	0.0280
8	0.0277
9	0.0297
10	0.0271

If the s_i values settle down in this way, it indicates that the closest polynomial approximation justified by the data has been achieved. The degree which first gives the approximately constant value of s_i (degree 5 in the example) is the appropriate degree to select. (If you are prepared to accept a fit higher than sixth degree you should simply find a high enough value of k to enable the type of behaviour indicated by the example to be detected: thus you should seek values of k for which at least 4 or 5 consecutive values of s_i are approximately the same.) If the degree were allowed to go high enough, s_i would, in most cases, eventually start to decrease again, indicating that the data points are being fitted too closely and that undesirable fluctuations are developing between the points. In some cases, particularly with a small number of data points, this final decrease is not distinguishable from the initial decrease in s_i. In such cases, you may seek an acceptable fit by examining the graphs of several of the polynomials obtained. Failing this, you may (a) seek a transformation of variables which improves the behaviour, (b) try fitting a spline, or (c) provide more data points. If data can be provided simply by

drawing an approximating curve by hand and reading points from it, use the points discussed in Section 3.2.2.

3.2.2 Least squares polynomials: selected data points

When you are at liberty to choose the x values of data points, such as when the points are taken from a graph, it is most advantageous when fitting with polynomials to use the values $x_r = \cos(\pi r/n)$, for $r = 0, 1, \ldots, n$ for some value of n, a suitable value for which is discussed at the end of this section. Note that these x_r relate to the variable x after it has been normalized so that its range of interest is -1 to $+1$. nag_fit_1dcheb_arb (e02ad) may then be used as in Section 3.2.1 to seek a satisfactory fit. However, if the ordinate values are of equal weight, as would often be the case when they are read from a graph, nag_fit_1dcheb_glp (e02af) is to be preferred, as being simpler to use and faster. This latter algorithm provides the coefficients a_j, for $j = 0, 1, \ldots, n$, in the Chebyshev series form of the polynomial of degree n which interpolates the data. In a satisfactory case, the later coefficients in this series, after some initial significant ones, will exhibit a random behaviour, some positive and some negative, with a size about that of the errors in the data or less. All these 'random' coefficients should be discarded, and the remaining (initial) terms of the series be taken as the approximating polynomial. This truncated polynomial is a least squares fit to the data, though with the point at each end of the range given half the weight of each of the other points. The following example illustrates a case in which degree 5 or perhaps 6 would be chosen for the approximating polynomial.

j	a_j
0	9.315
1	−8.030
2	0.303
3	−1.483
4	0.256
5	−0.386
6	0.076
7	0.022
8	0.014
9	0.005
10	0.011
11	−0.040
12	0.017
13	−0.054
14	0.010
15	−0.034
16	−0.001

Basically, the value of n used needs to be large enough to exhibit the type of behaviour illustrated in the above example. A value of 16 is suggested as being satisfactory for very many practical problems, the required cosine values for this value of n being given on page 11 of Cox and Hayes (1973). If a satisfactory fit is not obtained, a spline fit should be tried, or, if you are prepared to accept a higher degree of polynomial, n should be increased: doubling n is an advantageous strategy, since the set of values $\cos(\pi r/n)$, for $r = 0, 1, \ldots, n$, contains all the values of $\cos(\pi r/2n)$, for $r = 0, 1, \ldots, n$, so that the old dataset will then be re-used in the new one. Thus, for example, increasing n from 16 to 32 will require only 16 new data points, a smaller number than for any other increase of n. If data points are particularly expensive to obtain, a smaller initial value than 16 may be tried, provided you are satisfied that the number is adequate to reflect the character of the underlying relationship. Again, the number should be doubled if a satisfactory fit is not obtained.

3.2.3 Minimax space polynomials

nag_1d_minimax_polynomial (e02al) determines the polynomial of given degree which is a minimax space fit to arbitrary data points with equal weights. (If unequal weights are required, the polynomial must be treated as a general linear function and fitted using nag_fit_glin_linf (e02gc).) To arrive at a satisfactory degree it will be necessary to try several different degrees and examine the results graphically. Initial guidance can be obtained from the value of the maximum residual: this will vary

with the degree of the polynomial in very much the same way as does s_i in least squares fitting, but it is much more expensive to investigate this behaviour in the same detail.

The algorithm uses the power-series form of the polynomial so for numerical accuracy it is advisable to normalize the data range of x to $[-1, 1]$.

3.3 Cubic Spline Curves

3.3.1 Least squares cubic splines

nag_fit_1dspline_knots (e02ba) fits to arbitrary data points, with arbitrary weights, a cubic spline with interior knots specified by you. The choice of these knots so as to give an acceptable fit must largely be a matter of trial and error, though with a little experience a satisfactory choice can often be made after one or two trials. It is usually best to start with a small number of knots (too many will result in unwanted fluctuations in the fit, or even in there being no unique solution) and, examining the fit graphically at each stage, to add a few knots at a time at places where the fit is particularly poor. Moving the existing knots towards these places will also often improve the fit. In regions where the behaviour of the curve underlying the data is changing rapidly, closer knots will be needed than elsewhere. Otherwise, positioning is not usually very critical and equally-spaced knots are often satisfactory. See also the next section, however.

A useful feature of the function is that it can be used in applications which require the continuity to be less than the normal continuity of the cubic spline. For example, the fit may be required to have a discontinuous slope at some point in the range. This can be achieved by placing three coincident knots at the given point. Similarly a discontinuity in the second derivative at a point can be achieved by placing two knots there. Analogy with these discontinuous cases can provide guidance in more usual cases: for example, just as three coincident knots can produce a discontinuity in slope, so three close knots can produce a rapid change in slope. The closer the knots are, the more rapid can the change be.

An example set of data is given in Figure 1. It is a rather tricky set, because of the scarcity of data on the right, but it will serve to illustrate some of the above points and to show some of the dangers to be avoided. Three interior knots (indicated by the vertical lines at the top of the diagram) are chosen as a start. We see that the resulting curve is not steep enough in the middle and fluctuates at both ends, severely on the right. The spline is unable to cope with the shape and more knots are needed.

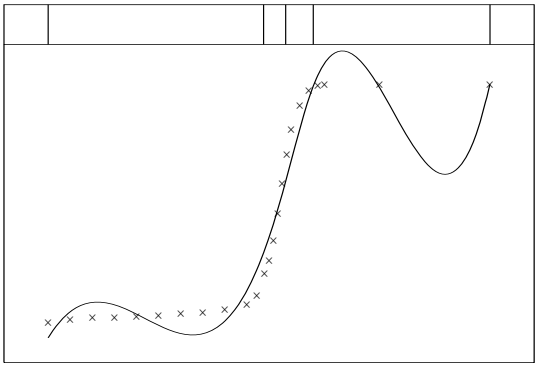


Figure 1

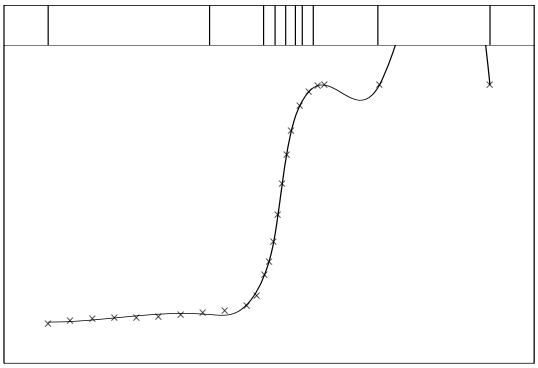


Figure 2

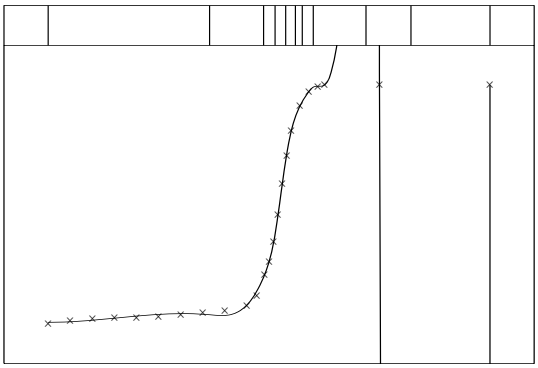


Figure 3

In Figure 2, three knots have been added in the centre, where the data shows a rapid change in behaviour, and one further out at each end, where the fit is poor. The fit is still poor, so a further knot is added in this region and, in Figure 3, disaster ensues in rather spectacular fashion.

The reason is that, at the right-hand end, the fits in Figures , have been interpreted as poor simply because of the fluctuations about the curve underlying the data (or what it is naturally assumed to be). But the fitting process knows only about the data and nothing else about the underlying curve, so it is important to consider only closeness to the data when deciding goodness-of-fit.

Thus, in Figure 1, the curve fits the last two data points quite well compared with the fit elsewhere, so no knot should have been added in this region. In Figure 2, the curve goes exactly through the last two points, so a further knot is certainly not needed here.

Figure 4 shows what can be achieved without the extra knot on each of the flat regions. Remembering that within each knot interval the spline is a cubic polynomial, there is really no need to have more than one knot interval covering each flat region.

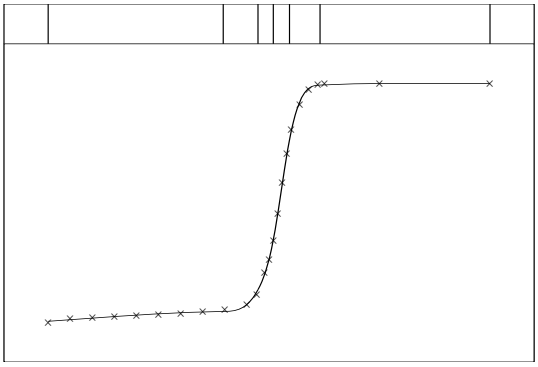


Figure 4

What we have, in fact, in Figures , is a case of too many knots (so too many coefficients in the spline equation) for the number of data points. The warning in the second paragraph of Section 2.1 was that the fit will then be too close to the data, tending to have unwanted fluctuations between the data points. The warning applies locally for splines, in the sense that, in localities where there are plenty of data points, there can be a lot of knots, as long as there are few knots where there are few points, especially near the ends of the interval. In the present example, with so few data points on the right, just the one extra knot in Figure 2 is too many! The signs are clearly present, with the last two points fitted exactly (at least to the graphical accuracy and actually much closer than that) and fluctuations **within** the last two knot-intervals (see Figure 1, where only the final point is fitted exactly and one of the wobbles spans several data points).

The situation in Figure 3 is different. The fit, if computed exactly, **would** still pass through the last two data points, with even more violent fluctuations. However, the problem has become so ill-conditioned that all accuracy has been lost. Indeed, if the last interior knot were moved a tiny amount to the right, there would be no unique solution and an error message would have been caused. **Near-singularity** is, sadly, not picked up by the function, but can be spotted readily in a graph, as Figure 3. B-spline coefficients becoming large, with alternating signs, is another indication. However, it is better to avoid such situations, firstly by providing, whenever possible, data adequately covering the range of interest, and secondly by placing knots only where there is a reasonable amount of data.

The example here could, in fact, have utilized from the start the observation made in the second paragraph of this section, that three close knots can produce a rapid change in slope. The example has two such rapid changes and so requires two sets of three close knots (in fact, the two sets can be so close that one knot can serve in both sets, so only five knots prove sufficient in Figure 4). It should be noted, however, that the rapid turn occurs within the range spanned by the three knots. This is the reason that the six knots in Figure 2 are not satisfactory as they do not quite span the two turns.

Some more examples to illustrate the choice of knots are given in Cox and Hayes (1973).

3.3.2 Automatic fitting with cubic splines

nag_fit_1dspline_auto (e02be) fits cubic splines to arbitrary data points with arbitrary weights but itself chooses the number and positions of the knots. You have to supply only a threshold for the sum of squares of residuals. The function first builds up a knot set by a series of trial fits in the ℓ_2 norm. Then, with the knot set decided, the final spline is computed to minimize a certain smoothing measure subject to satisfaction of the chosen threshold. Thus it is easier to use than nag_fit_1dspline_knots (e02ba) (see the previous section), requiring only some experimentation with this threshold. It should therefore be first choice unless you have a preference for the ordinary least squares fit or, for example, if you wish to experiment with knot positions, trying to keep their number down (nag_fit_1dspline_auto (e02be) aims only to be reasonably frugal with knots).

3.4 Polynomial and Spline Surfaces

3.4.1 Least squares polynomials

nag_fit_2dcheb_lines (e02ca) fits bivariate polynomials of the form (12), with k and ℓ specified by you, to data points in a particular, but commonly occurring, arrangement. This is such that, when the data points are plotted in the plane of the independent variables x and y, they lie on lines parallel to the x-axis. Arbitrary weights are allowed. The matter of choosing satisfactory values for k and ℓ is discussed in Section 9 in nag_fit_2dcheb_lines (e02ca).

3.4.2 Least squares bicubic splines

nag_fit_2dspline_panel (e02da) fits to arbitrary data points, with arbitrary weights, a bicubic spline with its two sets of interior knots specified by you. For choosing these knots, the advice given for cubic splines, in Section 3.3.1 above, applies here too (see also the next section, however). If changes in the behaviour of the surface underlying the data are more marked in the direction of one variable than of the other, more knots will be needed for the former variable than the latter. Note also that, in the surface case, the reduction in continuity caused by coincident knots will extend across the whole spline surface: for example, if three knots associated with the variable x are chosen to coincide at a value L, the spline surface will have a discontinuous slope across the whole extent of the line $x = L$.

With some sets of data and some choices of knots, the least squares bicubic spline will not be unique. This will not occur, with a reasonable choice of knots, if the rectangle R is well covered with data points: here R is defined as the smallest rectangle in the (x, y) plane, with sides parallel to the axes, which contains all the data points. Where the least squares solution is not unique, the minimal least squares solution is computed, namely that least squares solution which has the smallest value of the sum of squares of the B-spline coefficients c_{ij} (see the end of Section 2.3.2 above). This choice of least squares solution tends to minimize the risk of unwanted fluctuations in the fit. The fit will not be reliable, however, in regions where there are few or no data points.

3.4.3 Automatic fitting with bicubic splines

nag_fit_2dspline_sctr (e02dd) fits bicubic splines to arbitrary data points with arbitrary weights but chooses the knot sets itself. You have to supply only a threshold for the sum of squares of residuals. Just like the automatic curve, nag_fit_1dspline_auto (e02be) (see Section 3.3.2), nag_fit_2dspline_sctr (e02dd) then builds up the knot sets and finally fits a spline minimizing a smoothing measure subject to satisfaction of the threshold. Again, this easier to use function is normally to be preferred, at least in the first instance.

nag_fit_2dspline_grid (e02dc) is a very similar function to nag_fit_2dspline_sctr (e02dd) but deals with data points of equal weight which lie on a rectangular mesh in the (x, y) plane. This kind of data allows a very much faster computation and so is to be preferred when applicable. Substantial departures from equal weighting can be ignored if you are not concerned with statistical questions, though the quality of the fit will suffer if this is taken too far. In such cases, you should revert to nag_fit_2dspline_sctr (e02dd).

3.4.4 Large data sets

For fitting large sets of equally weighted, arbitrary data points, nag_fit_2dspline_ts_sctr (e02jd) provides a two stage method where local, least squares, polynomial fits are extended to form a C^1 surface.

3.5 General Linear and Nonlinear Fitting Functions

3.5.1 General linear functions

For the general linear function (15), functions are available for fitting in all three norms. The least squares functions (which are to be preferred unless there is good reason to use another norm – see Section 2.1.1) are in Chapters F04 and F08. The ℓ_∞ function is nag_fit_glin_linf (e02gc). Two functions for the ℓ_1 norm are provided, nag_fit_glin_l1sol (e02ga) and nag_fit_glinc_l1sol (e02gb). Of these two, the former should be tried in the first instance, since it will be satisfactory in most cases, has a much shorter code and is faster. nag_fit_glinc_l1sol (e02gb), however, uses a more stable computational algorithm and therefore may provide a solution when nag_fit_glin_l1sol (e02ga) fails to do so. It also provides a facility for imposing linear inequality constraints on the solution (see Section 3.6).

All the above functions are essentially linear algebra functions, and in considering their use we need to view the fitting process in a slightly different way from hitherto. Taking y to be the dependent variable and x the vector of independent variables, we have, as for equation (1) but with each x_r now a vector,

$$\epsilon_r = y_r - f(x_r), \quad r = 1, 2, \ldots, m.$$

Substituting for $f(x)$ the general linear form (15), we can write this as

$$c_1\phi_1(x_r) + c_2\phi_2(x_r) + \cdots + c_p\phi_p(x_r) = y_r - \epsilon_r, \quad r = 1, 2, \ldots, m. \tag{19}$$

Thus we have a system of linear equations in the coefficients c_j. Usually, in writing these equations, the ϵ_r are omitted and simply taken as implied. The system of equations is then described as an overdetermined system (since we must have $m \geq p$ if there is to be the possibility of a unique solution to our fitting problem), and the fitting process of computing the c_j to minimize one or other of the norms (2), (3) and (4) can be described, in relation to the system of equations, as solving the overdetermined system in that particular norm. In matrix notation, the system can be written as

$$\Phi c = y, \tag{20}$$

where Φ is the m by p matrix whose element in row r and column j is $\phi_j(x_r)$, for $r = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, p$. The vectors c and y respectively contain the coefficients c_j and the data values y_r.

All four functions, however, use the standard notation of linear algebra, the overdetermined system of equations being denoted by

$$Ax = b. \tag{21}$$

(In fact, nag_linsys_real_gen_lsqsol (f04am) can deal with several right-hand sides simultaneously, and thus is concerned with a matrix of right-hand sides, denoted by B, instead of the single vector b, and correspondingly with a matrix X of solutions instead of the single vector x.) The correspondence between this notation and that which we have used for the data-fitting problem (20) is therefore given by

$$\begin{aligned} A &\equiv \Phi, \\ x &\equiv c, \\ b &\equiv y. \end{aligned} \tag{22}$$

Note that the norms used by these functions are the unweighted norms (2), (3) and (4). If you wish to apply weights to the data points, that is to use the norms (5), (6) or (7), the equivalences (22) should be replaced by

$$\begin{aligned} A &\equiv D\Phi, \\ x &\equiv c, \\ b &\equiv Dy, \end{aligned}$$

where D is a diagonal matrix with w_r as the rth diagonal element. Here w_r, for $r = 1, 2, \ldots, m$, is the weight of the rth data point as defined in Section 2.1.2.

3.5.2 Nonlinear functions

Functions for fitting with a nonlinear function in the ℓ_2 norm are provided in Chapter E04. Consult the E04 Chapter Introduction for the appropriate choice of function. Again, however, the notation adopted is different from that we have used for data fitting. In the latter, we denote the fitting function by $f(x; c)$, where x is the vector of independent variables and c is the vector of coefficients, whose values are to be determined. The squared ℓ_2 norm, to be minimized with respect to the elements of c, is then

$$\sum_{r=1}^{m} w_r^2 [y_r - f(x_r; c)]^2 \tag{23}$$

where y_r is the rth data value of the dependent variable, x_r is the vector containing the rth values of the independent variables, and w_r is the corresponding weight as defined in Section 2.1.2.

On the other hand, in the nonlinear least squares functions of Chapter E04, the function to be minimized is denoted by

$$\sum_{i=1}^{m} f_i^2(x), \tag{24}$$

the minimization being carried out with respect to the elements of the vector x. The correspondence between the two notations is given by

$$x \equiv c \text{ and } f_i(x) \equiv w_r[y_r - f(x_r; c)], \quad i = r = 1, 2, \ldots, m.$$

Note especially that the vector x of variables of the nonlinear least squares functions is the vector c of coefficients of the data-fitting problem, and in particular that, if the selected function requires derivatives of the $f_i(x)$ to be provided, these are derivatives of $w_r[y_r - f(x_r; c)]$ with respect to the coefficients of the data-fitting problem.

3.6 Constraints

At present, there are only a limited number of functions which fit subject to constraints. nag_fit_glinc_l1sol (e02gb) allows the imposition of linear inequality constraints (the inequality (17) for example) when fitting with the general linear function in the ℓ_1 norm. In addition, Chapter E04

contains a function, nag_opt_lsq_gencon_deriv (e04us), which can be used for fitting with a nonlinear function in the ℓ_2 norm subject to general equality or inequality constraints.

The remaining two constraint functions relate to fitting with polynomials in the ℓ_2 norm. nag_fit_1dcheb_con (e02ag) deals with polynomial curves and allows precise values of the fitting function and (if required) all its derivatives up to a given order to be prescribed at one or more values of the independent variable. The related surface-fitting nag_fit_2dcheb_lines (e02ca), designed for data on lines as discussed in Section 3.4.1, has a feature which permits precise values of the function and its derivatives to be imposed all along one or more lines parallel to the x or y axes (see the function document for the relationship between these normalized variables and your original variables). In this case, however, the prescribed values cannot be supplied directly to the function: instead, you must provide modified data ordinates $F_{r,s}$ and polynomial factors $\gamma_1(x)$ and $\gamma_2(x)$, as defined on page 95 of Hayes (1970).

3.7 Evaluation, Differentiation and Integration

Functions are available to evaluate, differentiate and integrate polynomials in Chebyshev series form and cubic or bicubic splines in B-spline form. These polynomials and splines may have been produced by the various fitting functions or, in the case of polynomials, from prior calls of the differentiation and integration functions themselves.

nag_fit_1dcheb_eval (e02ae) and nag_fit_1dcheb_eval2 (e02ak) evaluate polynomial curves: the latter has a longer argument list but does not require you to normalize the values of the independent variable and can accept coefficients which are not stored in contiguous locations. nag_fit_2dcheb_eval (e02cb) evaluates polynomial surfaces, nag_fit_1dspline_eval (e02bb) cubic spline curves, nag_fit_2dspline_evalv (e02de) and nag_fit_2dspline_evalm (e02df) bicubic spline surfaces, and nag_fit_2dspline_derivm (e02dh) bicubic spline surfaces with derivatives.

Differentiation and integration of polynomial curves are carried out by nag_fit_1dcheb_deriv (e02ah) and nag_fit_1dcheb_integ (e02aj) respectively. The results are provided in Chebyshev series form and so repeated differentiation and integration are catered for. Values of the derivative or integral can then be computed using the appropriate evaluation function. Polynomial surfaces can be treated by a sequence of calls of one or other of the same two functions, differentiating or integrating the form (12) piece by piece. For example, if, for some given value of j, the coefficients a_{ij}, for $i = 0, 1, \ldots, k$, are supplied to nag_fit_1dcheb_deriv (e02ah), we obtain coefficients $\bar{a}_{ij}$ say, for $i = 0, 1, \ldots, k-1$, which are the coefficients in the derivative with respect to x of the polynomial

$$\sum_{i=0}^{k} a_{ij} T_i(x).$$

If this is repeated for all values of j, we obtain all the coefficients in the derivative of the surface with respect to x, namely

$$\sum_{i=0}^{k-1} \sum_{j=0}^{\ell} \bar{a}_{ij} T_j(y). \tag{25}$$

The derivative of (12), or of (25), with respect to y can be obtained in a corresponding manner. In the latter case, for example, for each value of i in turn we supply the coefficients $\bar{a}_{i0}, \bar{a}_{i1}, \bar{a}_{i2}, \ldots$, to the function. Values of the resulting polynomials, such as (25), can subsequently be computed using nag_fit_2dcheb_eval (e02cb). It is important, however, to note one exception: the process described will not give valid results for differentiating or integrating a surface with respect to y if the normalization of x was made dependent upon y, an option which is available in the fitting function nag_fit_2dcheb_lines (e02ca).

For splines the differentiation and integration functions provided are of a different nature from those for polynomials. nag_fit_1dspline_deriv (e02bc) and nag_fit_1dspline_deriv_vector (e02bf) provide values of a cubic spline curve together with its first three derivatives (the rest, of course, are zero) at a given value, or vector of values, of x. nag_fit_1dspline_integ (e02bd) computes the value of the definite integral of a cubic spline over its whole range. Again the functions can be applied to surfaces, this time of the form (14). For example, if, for each value of j in turn, the coefficients c_{ij}, for $i = 1, 2, \ldots, p$, are supplied to nag_fit_1dspline_deriv (e02bc) with $x = x_0$ and on each occasion we select from the output

the value of the second derivative, d_j say, and if the whole set of d_j are then supplied to the same function with $x = y_0$, the output will contain all the values at (x_0, y_0) of

$$\frac{\partial^2 f}{\partial x^2} \quad \text{and} \quad \frac{\partial^{r+2} f}{\partial x^2 \partial y^r}, \quad r = 1, 2, 3.$$

Equally, if after each of the first p calls of nag_fit_1dspline_deriv (e02bc) we had selected the function value (nag_fit_1dspline_eval (e02bb) would also provide this) instead of the second derivative and we had supplied these values to nag_fit_1dspline_integ (e02bd), the result obtained would have been the value of

$$\int_A^B f(x_0, y)\, dy,$$

where A and B are the end points of the y interval over which the spline was defined.

3.8 Padé Approximants

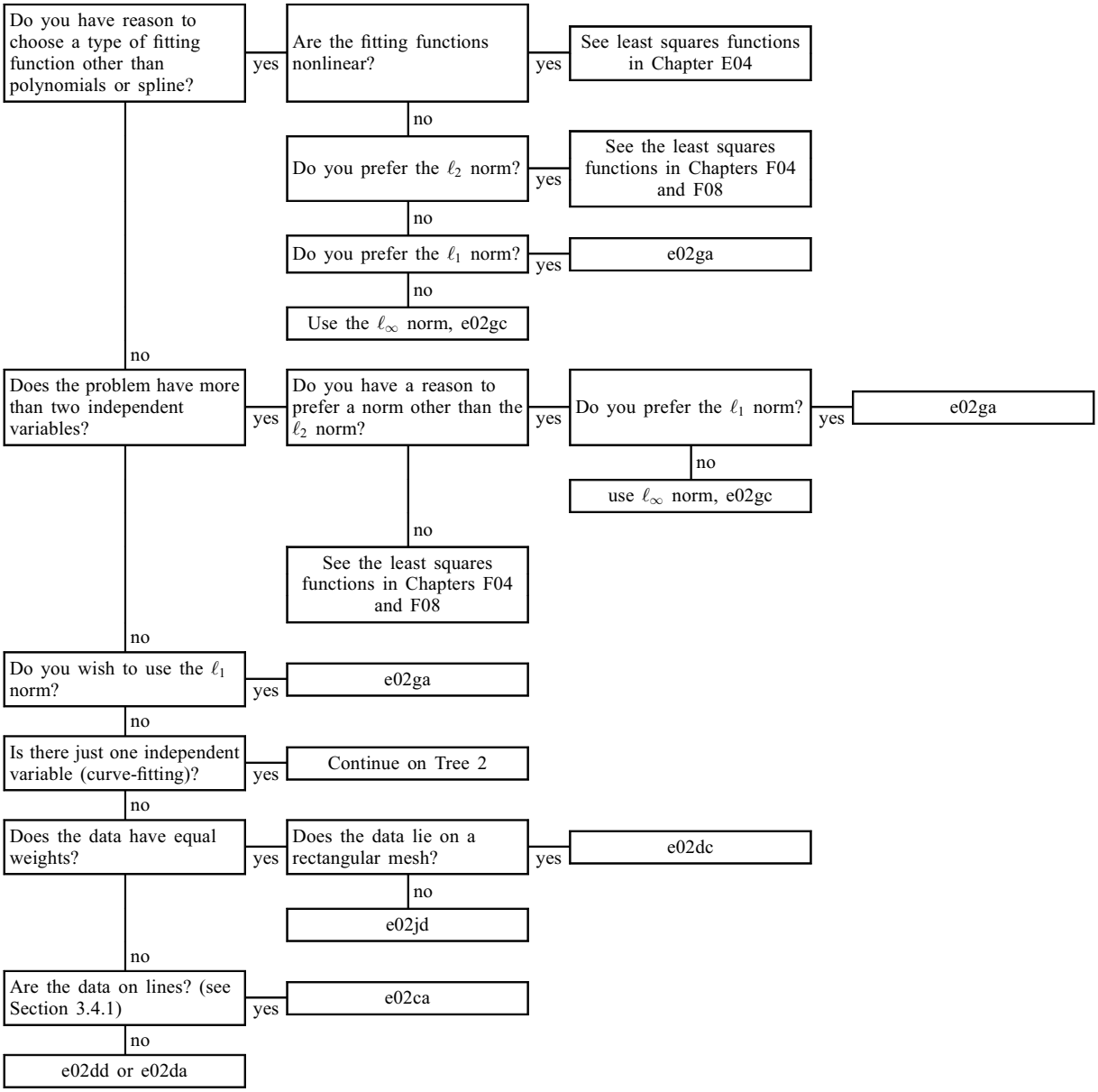
Given two non-negative integers ℓ and m, and the coefficients in the Maclaurin series expansion of a function up to degree $\ell + m$, nag_fit_pade_app (e02ra) calculates the Padé approximant of degree ℓ in the numerator and degree m in the denominator. See Section 3 and Section 10 in nag_fit_pade_app (e02ra) for further advice.

nag_fit_pade_eval (e02rb) is provided to compute values of the Padé approximant, once it has been obtained.

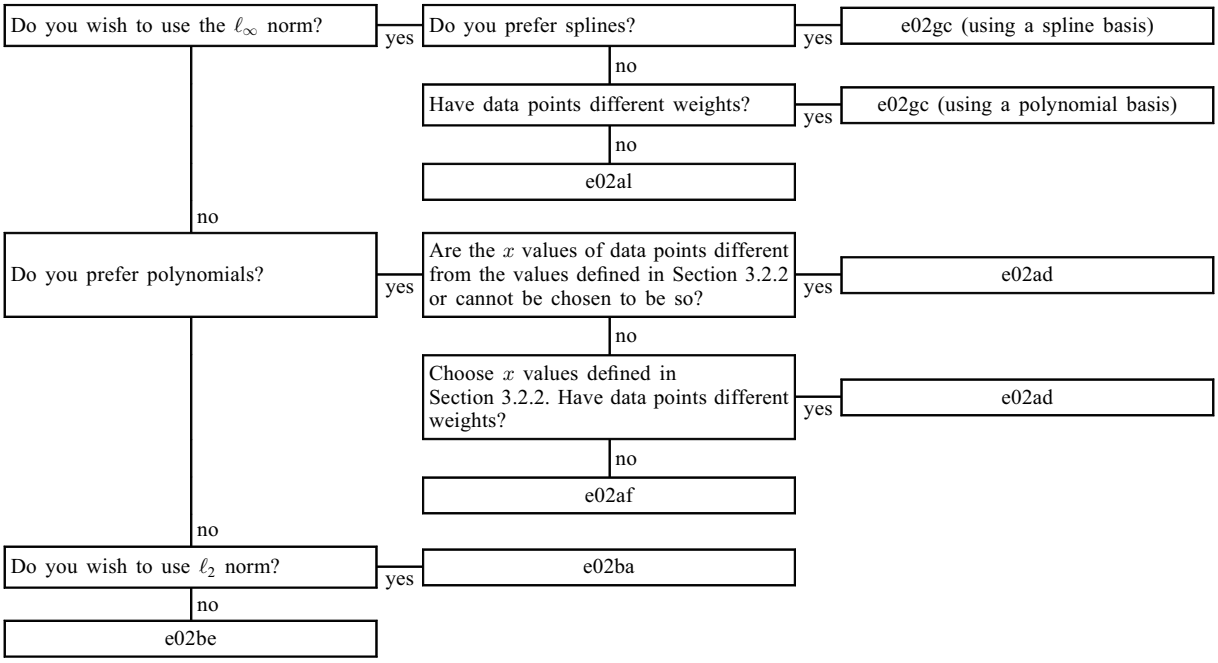
4 Decision Trees

Note: these Decision Trees are concerned with unconstrained fitting; for constrained fitting, consult Section 3.6.

Tree 1


Do you have reason to choose a type of fitting function other than polynomials or spline?
yes
Are the fitting functions nonlinear?
yes
See least squares functions in Chapter E04
no
Do you prefer the ℓ_2 norm?
yes
See the least squares functions in Chapters F04 and F08
no
Do you prefer the ℓ_1 norm?
yes
e02ga
no
Use the ℓ_∞ norm, e02gc
no
Does the problem have more than two independent variables?
yes
Do you have a reason to prefer a norm other than the ℓ_2 norm?
yes
Do you prefer the ℓ_1 norm?
yes
e02ga
no
use ℓ_∞ norm, e02gc
no
See the least squares functions in Chapters F04 and F08
no
Do you wish to use the ℓ_1 norm?
yes
e02ga
no
Is there just one independent variable (curve-fitting)?
yes
Continue on Tree 2
no
Does the data have equal weights?
yes
Does the data lie on a rectangular mesh?
yes
e02dc
no
e02jd
no
Are the data on lines? (see Section 3.4.1)
yes
e02ca
no
e02dd or e02da

Tree 2



5 Functionality Index

6 References

Baker G A (1975) *Essentials of Padé Approximants* Academic Press, New York

Cox M G and Hayes J G (1973) Curve fitting: a guide and suite of algorithms for the non-specialist user *NPL Report NAC26* National Physical Laboratory

Hayes J G (ed.) (1970) *Numerical Approximation to Functions and Data* Athlone Press, London

Hayes J G (1974) Numerical methods for curve and surface fitting *Bull. Inst. Math. Appl.* **10** 144–152

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