Spline2 V6.0 Tutorial and User Manual – Unix version

Date: 1 February 2002
Author: Barend J. Thijsse
Delft University of Technology
Laboratory of Materials Science
2628 AL Delft, Netherlands
E-mail: B.J.Thijsse@tnw.tudelft.nl

0. How to navigate this document

- **The purpose of *spline2*** is explained in Section 1. The first two paragraphs are essential.
- Section 1a briefly describes spline functions. It is non-essential reading for the impatient user.
- Autocorrelation is introduced in Section 2. This can be skipped when time pressure is heavy.
- **How to use *spline2***, Section 3, is essential reading.
- Section 4 describes the runtime options of spline2. Skip on first reading, but know that it's there.
- Section 4a illustrates how spline2 handles the example datafiles.
- **How to use the utility program *evalsp***, Section 5, is essential reading.
- The numerical methods and the underlying algorithms are explained in Section 6.

1. Purpose of spline2

Spline2 is a program for freestyle data fitting. Simply said, spline2 *constructs a smooth curve through datapoints containing noise* (Fig.1). This smooth curve, which is considered to be the best approximation of the "trend" in the data (i.e., the true functional relationship), is made available to the user as an analytic function. The great advantage of this is that data can be easily analyzed and manipulated in terms of this analytic representation – examples of such operations are: differentiation, determination of roots and inflection points, Fourier transformation, or substraction of two datasets having different abscissae. These and other operations can be carried out using the accomanying utility program *evalsp*.

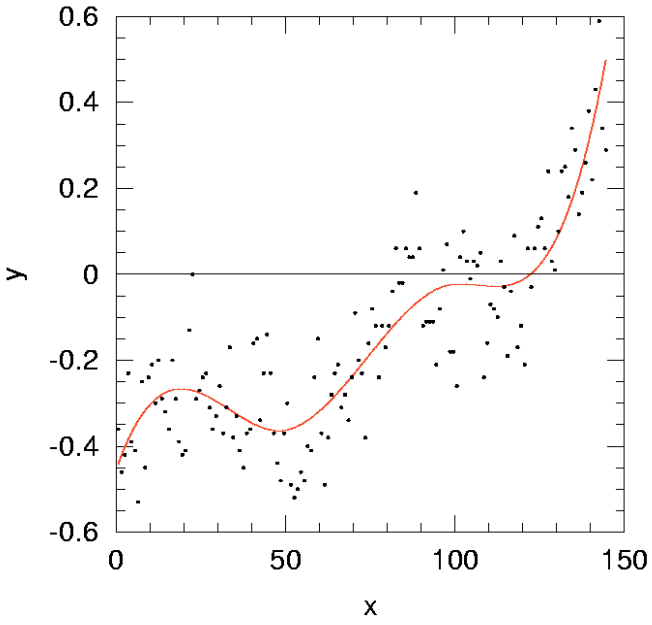

Fig. 1. Datafile *example3* and result of *spline2 –s –q* (see Sec. 3).

It is important to note that the user does not have to decide on the type of function that is used to represent the data trend (e.g., exponential, polynomial, goniometric, or combinations thereof). The program generates *spline* functions for this purpose, which are extremely flexible and are capable of representing virtually any conceivable trend. Splines are briefly described in Sec. 1a and fully discussed in Ref. [1]. *There are two things that you cannot do with spline2: (1) Extrapolation outside the data range, and (2) Fitting a theoretical curve to the data in order to assess the validity of the theory and to estimate parameter values.*

Expressed mathematically, spline2 does the following. It is given datapoints x_i, y_i (i = 1, 2, .. N), of which the values y_i are assumed to contain random errors,

$$y_i = f(x_i) + \varepsilon_i. \qquad (1)$$

The function $y = f(x)$ is the true relationship that the user wants to recover, and ε_i is the "noise component" of y_i, having the following statistical properties,

$$E(\varepsilon_i) = 0, \qquad (2)$$
$$E(\varepsilon_i^2) = \sigma_i^2. \qquad (3)$$

(Here E denotes the expectation value.) The variance σ_i^2 of the random error in y_i, a quantity that is sometimes poorly known to the user, may be different for each datapoint. Note that the random errors can have any magnitude, even as small as rounding errors in calculated data. By systematically generating trial spline functions $S(x)$ and carefully analyzing the fit-residuals r_i,

$$r_i = y_i - S(x_i), \qquad (3a)$$

spline2 ultimately decides in which of these trial cases the statistical properties of the fit-residuals are the most consistent with those of ε_i. This then logically implies that the spline in question is the one that resembles $f(x)$ as closely as possible.

An important additional criterion is that given the choice between statistically equivalent candidate splines, spline2 will select the simplest spline. In mathematical language: the spline with the smallest number of *knots* (this term is dicussed below).

1a. Splines, intervals, and knots

A spline function, here generically denoted as $S(x)$, is a piecewise polynomial function. To visualize this, one should imagine that the datarange $x_1 ... x_N$ is divided up – as needed – in a certain number of *intervals*. The breakpoints between the intervals are called (interior) *knots*. The outer datapoints are also knots. Each polynomial piece of the spline covers an interval. The polynomial pieces are constructed in such a way that in each interior knot the polynomial pieces to the left and to the right of the knot have the same value, the same first derivative, the same second derivative, and so on. Only the highest derivative, a constant, is different on both sides of a knot. Taken together, these polynomial pieces form a smooth curve, of which the "flexibility" is highly dependent on the number of knots and their distribution along the x-axis. The principal task of spline2 is to determine the optimal number and distribution of knots to approximate the trend in the data under consideration.

2. Autocorrelation

Before proceeding further we first have to discuss the concept of autocorrelation, since a basic understanding of this is needed in order to understand how spline2 works and what its output tells you.

The noise present in many data can be considered "white". This term expresses the fact that the random errors in the data are independent of each other,

$$E(\varepsilon_i \varepsilon_j) = 0 \qquad (i \neq j). \qquad (4)$$

However, it is not safe to assume that this is always true. In many cases, often without the user knowing it, some kind of filtering or averaging process has acted on the data "after" the noise has originated. Consider the following example. Suppose your data x_i, y_i are 3-point running averages of (unknown) "parent" data x_i, z_i, as expressed by

$$y_i = \tfrac{1}{3}(z_{i-1} + z_i + z_{i+1}). \tag{5}$$

If the z-values contain white noise,

$$z_i = g(x_i) + \varepsilon_i, \tag{6}$$
$$E(\varepsilon_i) = 0, \tag{7}$$
$$E(\varepsilon_i^2) = \sigma_i^2, \tag{8}$$
$$E(\varepsilon_i \varepsilon_j) = 0 \qquad (i \neq j). \tag{9}$$

this does not mean that your y-values also contain white noise. In fact, it is not difficult to show that – when the error variances σ_i^2 in the z-values do not vary too much from one point to the next – *your* random errors r_i have the following properties:

$$E(r_i) = 0, \tag{10}$$
$$E(r_i^2) = \sigma_i^2 - \tfrac{1}{3}\sigma_i^2, \tag{11}$$
$$E(r_i r_{i+n}) \begin{cases} E(r_i^2)(1 - \tfrac{1}{3}|n|) & (|n| < 3) \\ 0 & (|n| \geq 3) \end{cases}. \tag{12}$$

Two things emerge from this. First, the amplitude of the noise has reduced by a factor of $\sqrt{3}$ (Eq. (11)). [This noise reduction is the reason why people average and smooth their data in the first place.] Second, and more importantly, your noise is no longer white. Eq. (12) shows that the random errors in neighboring datapoints have become correlated. If you would not be aware of this (it is easily overlooked in visual inspection of the data), freestyle curve fitting may lead to very incorrect results, since keeping the quantity $E(r_i r_{i+n})$ at the correct value is an important part of the spline2 algorithms, as we will see in Sec. 6.

Starting with version 6.0, spline2 automatically searches for these autocorrelation effects, so that user decisions are no longer necessary. *You have to use the –s option for this, however*. If not told otherwise, spline2 assumes that the autocorrelation function is of the following type:

$$\frac{E(\varepsilon_i \varepsilon_{i+n})}{E(\varepsilon_i^2)} = \exp(-(x_{i+n} - x_i)/\lambda). \tag{13}$$

This exponentially decaying form is a good approximation for many practical cases. The quantity λ is called the *autocorrelation length*. Spline2 determines the value of λ that agrees best with your data. Uncorrelated noise has $\lambda = 0$.

3. Using spline2

The easiest way to get familiar with spline2 is to start working with example data right away. (First-time users may first need to compile spline2 and evalsp. See the appendix for this.)

The recommended way to run spline2 is to use the –s and –q options. The datafile called example3 is a good one to start with. Type:

```
spline2 example3 -s -q > /dev/null
```

The –s option stands for “search for autocorrelation” and the –q option stands for “quiet”. The –q option generates output for evalsp. At first it is best to throw it away, as is done here. More on other options later.

You should get the following output on the screen (line numbers are for reference only):

```
(1)  ..... spline2 version 6.0 .....
(2)  Name of data file: example3
(3)  Format: x y
(4)  Number of datapoints participating in spline-fit = 145
(5)  Degree of spline = 3
(6)  Spline approximations are tested according to the Durbin-Watson test
(7)  Automatic search for autocorrelation in residuals
(8)  Assumed autocorrelation function of residuals: exponential: exp(-x/ksi)
(9)  Average datapoint spacing <delta x> = 1
(10) Splines with non-optimized breakpoints are allowed
(11) ksi-sweep: ++++++++++++++++ end sweep
(12) Spline-fit: rms=0.108386, dws=1.98444, l=3 (eqi) ksi=0.8 acffit=0.168573
```

The first 11 lines of output serve as a feedback to the user, showing what spline2 has been doing. The last line (12) is the *results* line, summarizing what spline2 has found. The spline itself is written to a file named *splres* (see Fig. 1 for a plot). We will discuss these issues in turn.

The datafile and the user estimates of the uncertainties in the data. Lines (2)-(4) give feedback about the data fed to spline2. The name of the datafile and the number of datapoints need no explanation. The Format line indicates that the data file is a 2-column file, where each line contains the x, y value pair of a single datapoint. The two numbers should be separated by “white space”, i.e., by any number of spaces or tabs. *In the datafile the x-values should appear in increasing order*. This is the standard datafile format. The only other allowed format is “x y s”. In this 3-column format each value pair x_i, y_i is followed by a positive number s_i, being an estmate of the uncertainty σ_i in the i^{th} point,

s_i is the user estmate of σ_i. (14)

For a 2-column datafile the uncertainty estimates s_i of all datapoints are given the arbitrary value 1. What this actually means is that all datapoints are given the same relative weight in the fit. This principle of relativity also applies to the 3-column case, in which the estimates s_i are specified: only the *relative* values of s_i matter, not the absolute values.

Line (5) gives the **polynomial degree** of the spline. Cubic splines are standard, as they suffice for most purposes. The degree of the spline can be altered by the –k option, see Sec. 4. Bear in mind that a higher degree does not necessarily lead to a better or more flexible curve, since spline2 uses the number of knots and the knot distribution to control the flexibility of the spline, which are very powerful for this purpose.

Line (6) reports that the **statistical “goodness-of-fit” test** applied is the Durbin-Watson test [2]. This is contrary to the χ^2-test that is normally applied when a theoretical function with a *fixed* number of parameters is fitted to data. The reason is that a spline can be given enormous (or even too much) flexibility by increasing the number of knots, and the Durbin-Watson test excels in

restricting this flexibility, especially when the errors in the data are not precisely known. The test is explained in Sec. 6.

Autocorrelation. Lines (7)-(9) show how spline2 attempts to detect autocorrelation effects in the data (see Sec. 2). Other ways than the automatic search shown here (invoked by the –s option) are discussed in Sec. 4. The average datapoint spacing on line (9),

$$\langle \Delta x \rangle \equiv \frac{x_N - x_1}{N - 1}, \tag{15}$$

is given only for user convenience. It is a relevant quantity to compare with, for example, the autocorrelation length ξ.

Line (10) indicates that splines with **non-optimized breakpoint positions** are allowed. This is the default mode. Only in very specific cases, which are often recognized by strange fit-results, the user can use the –e option to forbid these splines. This matter has not yet been studied thouroughly.

The **ksi-sweep** line (11) is generated in real time, while spline2 performs its calculations. Often spline2 is so fast that one does not see the string of plusses evolve. Each plus denotes a trial value of the autocorrelation length ξ.

The data on the **results** line (12) are the most important:

rms=0.108386
The quantity rms is the root mean square value of the noise amplitude in the data, as determined by spline2. The value is given relative to the user estimate, Eq. (14). To be precise,

$$\text{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\Delta_i}{s_i}\right)^2}\,. \tag{16}$$

In the example given here all values s_i were set equal to 1, and rms therefore expresses the actual noise amplitude in the data. However, if the values s_i *would* have been specified (in a 3-column file or by using the –a or –r options), *and* their values are correct, the value of rms would come out as 1. If the *s*-values would be too small (on average) by a factor of 2, rms would come out as 2, etc.

dws=1.98444
The quantity dws is the generalized Durbin-Watson statistic for the fitted spline (see Sec. 6). A value in the range 1.9-2.2 usually indicates a good fit. Larger values are suspect, since they may indicate that some of the noise is fitted. Smaller values, which are very rare, definitely point to a systematic misfit. (*Side note: dws is also a very useful statistic for measuring the quality of other types of fits to data, such as for example prescribed theoretical or model curves*.)

l=3 (eqi)
The parameter *l* is the number of intervals of the fitted spline, see Sec. 1a. The number of internal knots is one less than this. Unless your data are extremely complicated or sparse, *l* should only be a fraction of the number of datapoints. The abbreviation “(eqi)” stands for “equal information”, and it denotes that the fitted spline has its knots distributed so that each interval contains the same number of datapoints (or as close to this as possible). This is synonymous with “non-optimized breakpoints”. The other possiblility is “(opt)”, which stands for “optimized breakpoints”.

ksi=0.8
The autocorrelation length ξ reported for the spline fit (see Sec. 2) is expressed as a number measured on the x-axis. A value of zero or much smaller than the average data spacing Δx indicates that the data are essentially uncorrelated. A value much larger than Δx is suspect, because such large autocorrelation effects should normally be interpreted as part of the "trend" between x and y.

acffit=0.168573
The quantity acffit measures how closely the autocorrelation function of the fit-residuals matches the assumed autocorrelation function with autocorrelation length ξ (the previous number). When the automatic search option is used (–s), the value reported for ξ is the one that – out of all trial-values of ξ – leads to the smallest acffit, see Sec. 6. A value smaller than 0.2 is good, and a value in the range 0.2-0.3 is reasonable. Larger values of acffit indicate that the assumed autocorrelation model is probably incorrect.

File *splres* contains the point-to-point fit results. Its columns list the following data (top to bottom: i = 1, 2, .. N):

column 1	column 2	column 3	column 4	column 5	column 6	column 7
abscissa	ordinate	spline value	weighted fit-residual	weight in fit (user estimate)	spline 1st derivative	spline 2nd derivative
x	y	$S(x)$	d	$1/s^2$	dS/dx	d^2S/dx^2

The weighted residual d_i in point i (column 4) is given by

$$d_i \equiv \frac{r_i}{s_i} = \frac{y_i - S(x_i)}{s_i}. \qquad (17)$$

See Sec. 5 for how to compute the values of the spline and its derivatives in other points than the datapoints. This is often needed for generating good plots.

Other output files are *splacf*, *splksi*, *splstat*, and *splcall*. They are not very useful for a casual user. Their format is explained on-screen when spline2 is run without the –q option (except *splcall*'s format). Section 6 is essential for understanding their contents.

4. Spline2 options

For fine-tuning the behavior of spline2, one can run the program with many options other than (or in addition to) –s and –q, although this is not frequently needed. A brief overview of the available options can be obtained by typing just the name of the program:

```
spline2
```

Here we will discuss the options in some detail.

-s
Performs an automatic search for autocorrelation. Using –s is usually the best choice. Sometimes, however, one is pretty sure that the data are uncorrelated or that they have a known correlated behavior (or one wants to test this specifically). In these cases use the –n option or –K option (or

both) instead of –s. A second occasion for not using the –s option is when one is certain that the user-supplied estimates s_i of the uncertainties σ_i are correct (note: their *absolute* values should be correct, not just their relative values). In this case one should use –a or –r instead of –s. The –s option is new to version 6 of spline2. *The behavior of version 5 of spline2, see Ref. [3], can be simulated by NOT using the –s, –i, –K, and –e options.*

-q
The program runs in "quiet" mode. This is the preferred way. If –q is omitted, the user is given the option of selecting a spline other than the one calculated by the program. *This is almost always an unwise thing to do*. A better way is to change some of the tuning options below and repeat the fit, so that at least one knows what one is doing. Note that for historical reasons the –q option generates output on stdout. This output is hardly interesting for humans, because it is meant to be input for the program evalsp, see Sec. 5.

-i index
Specifies the assumed autocorrelation function (see Eq. (13)):

index = 1	Exponential	$\exp(-\lvert x/\xi \rvert)$	(default)
index = 2	Gaussian	$\exp(-\frac{1}{2}(x/\xi)^2)$	
index = 3	Linear	$\max(1-\frac{1}{2}\lvert x/\xi \rvert, 0)$	
index = 4	Sinc	$\sin(2\pi x/\xi)/(2\pi x/\xi)$	

All four functions have a value of roughly 0.5 if $\lvert x \rvert = \xi$, and die off to zero at approximately $\lvert x \rvert = 3\xi$. The sinc function is the only one that allows negative autocorrelation. The –i option is new to version 6 of spline2.

-x xbeg xend
Limits the fit to the data in the range [xbeg...xend]. This is sometimes convenient for performing fits to part of a dataset – one does not have to extract a new datafile.

-X xbeg xend
Excludes the data in the range [xbeg...xend] from the fit. This option can be used e.g. to let spline2 construct a "baseline" for a peak, by excluding the datapoints in the peak range from the fit. The –x and –X options can be used simultaneously.

-K ksi
Imposes a fixed value of ξ on the assumed autocorrelation function during the autocorrelation tests (instead of letting the program scan over a range of values, which is what –s does). The –s option cannot be used together with –K. The –K option is new to version 6 of spline2.

-n spacing
Imposes a fixed value of the data index spacing n (see Eq. (13)) during the autocorrelation tests (instead of a range of values, as is explained in Sec. 6). The default value is $n = 1$. The –n option implies a fixed value for ξ (If –K is not used to set this value, ξ defaults to 0). The effect of the –n option is that the autocorrelation is incompletely tested. The –s option cannot be used together with –n.

-a value
-r value
The –a and –r options tell spline2 that it should consider the user-supplied estimates s_i of the uncertainties σ_i as being correct (their *absolute* values). These options have the effect that the χ^2-test is used to find the best spline, and not the Durbin-Watson test. The –a and –r options have different meanings in what they tell about the uncertainties:

-a value	s_i = value (should be > 0)	(one value for all datapoints)
-a 0	s_i = value in 3rd column of the datafile	
-a -number	number of significant digits	(example: -a –4)
-r value	s_i = value · $\|y_i\|$	(value should be > 0)
-r -value	s_i = value · $\|y_i\|$	

The –a and –r options can not be used together with –s, –n, or –K, since autocorrelation effects are not tested when the χ^2-test is in use.

The following options are not often necessary or desired:

-e
Rejects splines with non-optimized breakpoints. The –e option is new to version 6 of spline2.

-k order
Splines of order other than 4 (cubic) are used. Note that the order of a spline is one more than its degree.

-o lopt
Forces knot optimization to start from an "equal information" spline with *lopt* knots, instead of leaving it up to spline2 to decide from which number of knots to start (see Sec. 6, algorithm steps I.1-I.3). This can be used to prevent the outcome of a too simple spline, which the program sometimes appears to produce for data with a rather structureless trend.

-L rejlev
Specifies a statistical rejection level other than 0.05, which is the default value (i.e., 5%). The rejection level is the fraction of the splines that – although representing the *correct* trend – are rejected, because of the unusual statistical properties of the residuals. A higher rejection level leads to stricter testing.

4a. Spline2 in action: various example datafiles

Whereas spline2 handles the datafile example3 extremely well in automatic mode (-s), it is instructive to see what happens if one would assume that the datapoints are uncorrelated. Running:

```
spline2 example3 –K 0 –q > /dev/null
```

leads to the following result:

```
Spline-fit: rms=0.102989, dws=1.97787, l=5 (eqi) ksi=0 acffit=0.407016
```

This shows that acffit is considerably larger than in the example of Sec. 3, indicating that the assumed autocorrelation function (here with $\phi = 0$) is probably incorrect. The spline needs five intervals instead of three, and therefore – not unexpectedly – runs somewhat closer to the datapoints (this can be concluded from the smaller rms-value). However, a visual inspection of this spline (the green line in Fig.2) shows that the spline looks very acceptable. Some might even consider it better than the "automatic" spline of Sec. 3 (red line in Fig. 2), yet on the basis of full statistical consistency spline2 prefers the red line.

If, again, one would assume that the datapoints were uncorrelated, but this time one would check this by *only* looking at the correlation of *immediately* neighboring points:

```
spline2 example3 –K 0 –n 1 –q > /dev/null
```

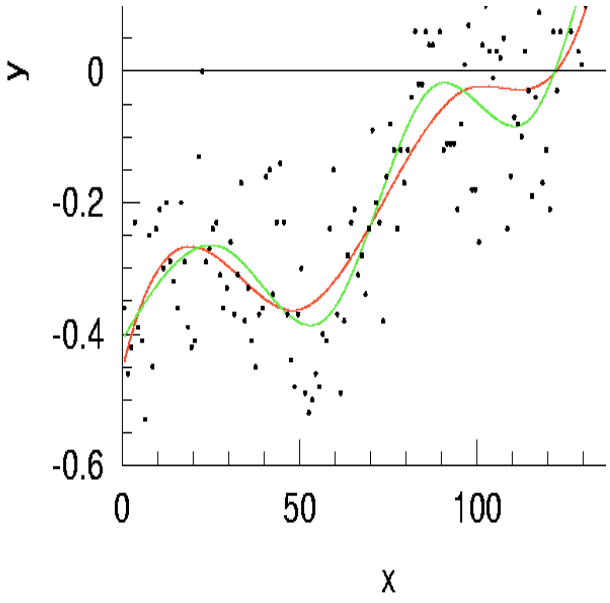

Fig. 2. Datafile *example3* and result of *spline2 –K 0 –q* (green line).

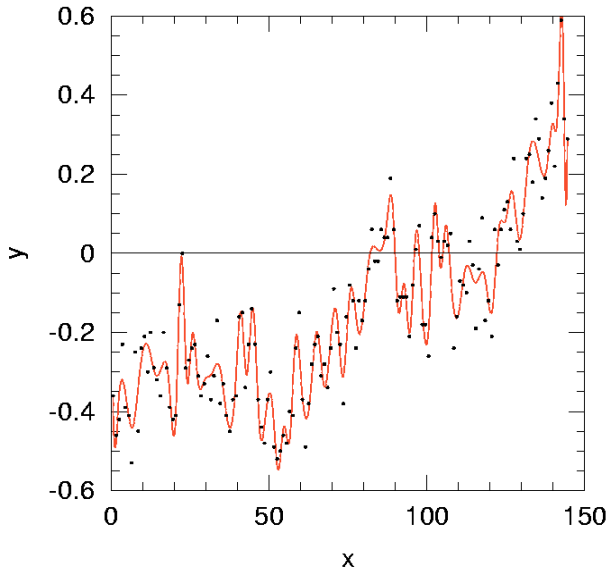

Fig. 3. Datafile *example3* and result of *spline2 –q*.

(or: spline2 example3 –q > /dev/null)

the result one gets is:

Spline-fit: rms=0.0741955, dws=3.0843, l=68 (opt) ksi=0 acffit=1.09183

Now it becomes clear why autocorrelation effects should not be taken lightly. This spline needs no fewer than 68 intervals, and it is very oscillatory (Fig. 3). It is clearly an incorrect approximation of the trend underlying the data.

Shifting our attention to the data in file example1, the result of applying spline2 –s –q to these data is shown in Fig. 4. The numerical outcome is:

Spline-fit: rms=0.974533, dws=2.03407, l=34 (opt) ksi=0.631888 acffit=0.049209

The spline needs 34 intervals and the fit shows that the data are practically uncorrelated (= 0.63, but |*x*| = 3.16). The acffit value is very good. Also, rms is nearly equal to 1. All of this makes sense, because these data are experimental pulse count data, which were uncorrelated, and of which the (Poisson) uncertainty estimates were precisely known (*example1* is a 3-column file). This is confirmed by a fit in which is held fixed at 0 and *n* at 1, because spline2 –q yields:

Spline-fit: rms=0.961404, dws=2.02985, l=40 (opt) ksi=0 acffit=0.0317353

The resulting spline, not shown, is not exactly the same as the one in Fig. 4 (more intervals are needed), but the difference is small. Finally, forcing the uncertainty estimates to be taken as correct leads to the same spline as the one resulting from the “automatic” mode. Spline2 –a 0 produces:

Spline-fit: rms=0.974533, dws=1.94891, l=34 (opt) ksi=0 acffit=0.048454

This is all quite satisfactory.

Fig. 5 shows the fit results for datafile example2. These data are those of example1 after processing by a filter having a width of approximately two datapoints. Fig. 5 shows the results of spline2 -s –i 2 –q, i.e. of a fit for which a gaussian autocorrelation function was assumed. The numerical results are:

Spline-fit: rms=51.2616, dws=2.07562, l=29 (opt) ksi=4.38149 acffit=0.31239

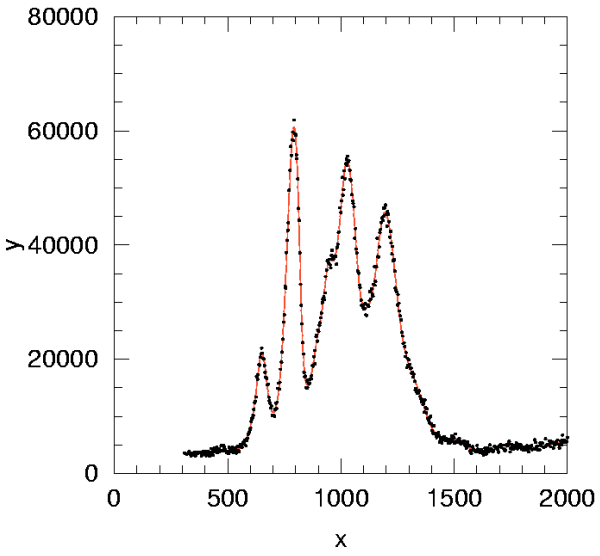

Fig. 4. Datafile *example1* and result of *spline2 –s –q*.

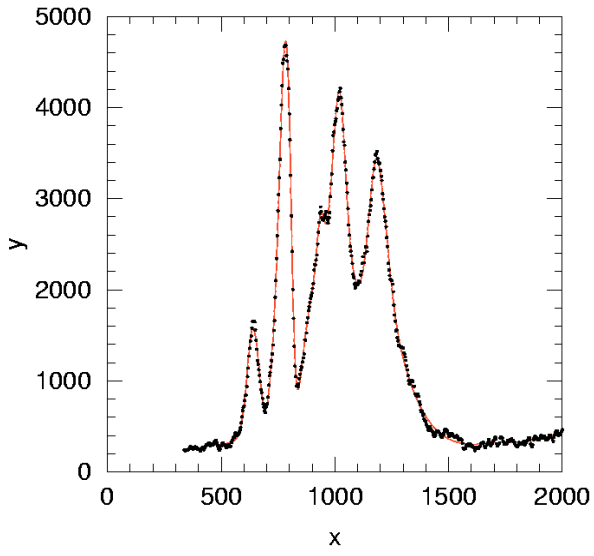

Fig. 5. Datafile *example2* and result of *spline2 –s –i 2 –q*.

which are quite reasonable. The autocorrelation length is found to be about 1.4□*x*□ roughly equal to the filter width, and also the number of intervals (29) is quite reasonable. If an exponential autocorrelation function is assumed (spline2 –s –q), the result is:

Spline-fit: rms=54.4925, dws=2.11049, l=25 (opt) ksi=5.63335 acffit=0.467772

which is only slightly less reliable (acffit is somewhat larger). However, if correlation is totally ignored and spline2 –q is used, one finds

Spline-fit: rms=16.9967, dws=2.707, l=176 (opt) ksi=0 acffit=0.708541

Again, ignorance of correlation leads to a very bad fit. All the signs of noise-fitting are there: a very large dws, an extremely large number of intervals, and a large value of acffit.

The famous [1] Titanium Heat data are shown in Fig 6. For these data □*x*□= 10, which makes the data rather sparse. Running spline in automatic mode (spline2 –s –q) leads to very strange results (red line). The outcome is a single(!) cubic polynomial for the whole data range, which is clearly totally wrong (Fig. 6). The numerical data show a complete misfit with the autocorrelation function:

Spline-fit: rms=0.319708, dws=2.9141, l=1 (eqi) ksi=30 acffit=0.830519

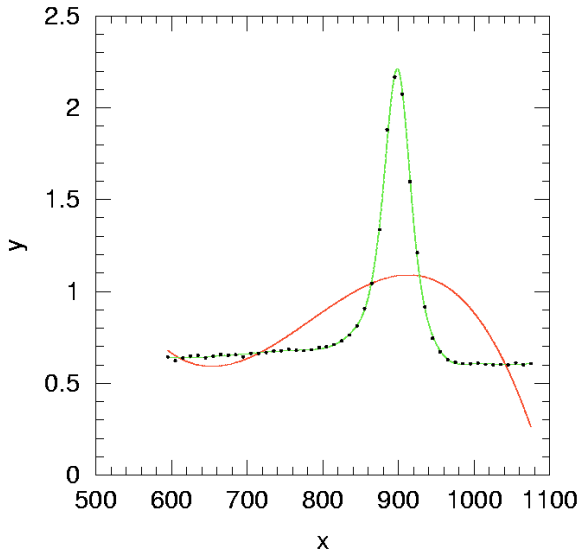

Fig. 6. Datafile *tiheat* and results of *spline2 –s –q* (red line) and *spline2 –q* (green line).

Table I. Results of *spline –s –q* fits to N=100 model datapoints $f(x) = \sin(15x) - (30/6\ \ (2\square))\ \exp(-(x/6)^2/2)$ + white gaussian noise with amplitude □. The datarange is x = [1...100]. The results are given as *mean* ± *standard deviation* derived from 100 replica datasets.

	□ = 0.1	□ = 0.2	□ = 0.4
□ estimated from spline fit (rms)	0.109 ± 0.013	0.216 ± 0.019	0.438 ± 0.035
q(□) (dws)	2.03 ± 0.09	1.98 ± 0.08	1.94 ± 0.09
number of spline intervals (l)	8.2 ± 1.0	7.2 ± 1.0	5.4 ± 1.1
□ estimated from spline fit (ksi)	0.22 ± 0.29	0.20 ± 0.30	0.23 ± 0.32
acffit	0.18 ± 0.08	0.16 ± 0.07	0.14 ± 0.06
rms deviation from true trend $[S(x) - f(x)]_{rms}$	0.055 ± 0.014	0.106 ± 0.021	0.207 ± 0.043

If –e is added, in order to forbid "eqi" splines (spline2 –s –e –q), the program starts struggling with the data and comes up with a spline (not shown) that it does not find acceptable:

Spline-fit: rms=0.00606332, dws=2.39357, l=20 (opt) ksi=12 acffit=1.00512 (no good fit) (DW indecisive)

It turns out that this spline runs extremely close to the datapoints (small rms), and that it has an even larger acffit than the previous one.The solution to finding a good fit is to ignore autocorrelation totally and apply just spline2 –q. Although we warned against this in the foregoing examples, *these* data appear to need it (presumably a rare exception). The result, shown in Fig. 6 as the green line, is characterized as:

Spline-fit: rms=0.0162182, dws=2.41782, l=8 (opt) ksi=0 acffit=0.429195

Finally, the data in example4 are an exercise for you to try. These data are model data, based on a known analytical function $f(x)$ to which white gaussian noise with amplitude □ = 0.1 was added. The fit found by spline2 confirms this. You should get:

Spline-fit: rms=0.107104, dws=2.06853, l=10 (eqi) ksi=0 acffit=0.105021

For a **quantitative assessment of spline2 performance**, model data are very useful. As an example, we have generated three series of 100 replicas of the data of example4, for different values of the noise amplitude □, each replica having a different set of random errors. Table I summarizes the results of applying spline2 –s –q to these data. What we see is that □ is well estimated by spline2 (apparently somewhat on the large side) and that q(□) is close to 2 in all cases. The average number of spline intervals systematically decreases with increasing data noise. This is not surprising, since spline2 has less and less information available to figure out the details of the shape of the underlying trend. The estimates of the autocorrelation lengths □ are all higher than the true value (□ = 0), although the true value does lie within one standard deviation of the mean. (From other tests it appears that there is no such bias of the □ estimate when a *nonzero* true value is used. Clearly, □ = 0 is a special case in that smaller values can not emerge as spline2 estimates, only larger values.) The □ estimates are independent of the noise level, which is reassuring. The values of acffit all fall within the range 0.17 ± 0.09 (slightly decreasing with increasing noise), which in this case apparently is the range that should be associated with a good fit. In the previous examples we have seen much lower values, however. Finally, the last line in Table I shows how well the fitted splines approximate the true trend $f(x)$. Obviously such results are only possible for model data. What we find from the examples is that the fitted splines are on average a factor 2 closer to the true trend than the noise amplitude □. As can be easily checked for other model data, especially when correlation is introduced, this factor 2 is certainly not a general result.

5. Using evalsp

The fitted spline calculated by spline2 can be manipulated by the utility program evalsp. One way to do this is to run spline2 with the –q option and send its output through a pipe to evalsp:

spline2 datafile [options] –q | evalsp [options]

The second way is to use spline2 in stand-alone mode first, as we have did in the previous examples, and afterwards run evalsp using the file *splrep* which supersp creates:

spline2 datafile [options] –q > /dev/null

evalsp splrep [options]

When evalsp runs without options, it creates a file named *evlres* which contains the spline and its derivatives in 2001 equidistant points, ranging from $x = x_1$ (the first datapoint) up to and including $x = x_N$ (the last datapoint). The columns of the file are as follows:

column 1	column 2	column 3	column 4
abscissa	spline value	spline 1st derivative	spline 2nd derivative
x	$S(x)$	dS/dx	d^2S/dx^2

Options for evalsp are:

-x xbeg xend xstep
Evaluates the spline and its derivatives in the range from x = xbeg up to and including x = xend, in increments of xstep (instead of using the default x-range). Example: -x 10 30 0.1. Keep in mind that extrapolation outside the data range is not possible with spline2. Results go to file evlres.

-f xfilename
Evaluates the spline and its derivatives in the x-values listed in the specified file (instead of using the default x-range). Results go to file evlres.

-F sbeg send sstep
Calculates the complex Fourier transform $F(s)$ of the spline in the range from s = sbeg up to and including s = send, in increments of sstep. The Fourier transform is defined as

$$F(s) = \int_{x_1}^{x_N} S(x)\exp(-2\pi i s x)\,dx .$$

The result goes to a file named *evlfour*, which has the following layout:

column 1	column 2	column 3
abscissa	real part of Fourier transform	imaginary part of Fourier transform
s	Re $[F(s)]$	Im $[F(s)]$

Additional options for on-screen display of certain quantities (any number of the following):

-e
Shows the minima and maxima of $S(x)$.

-i
Shows the inflection points of $S(x)$.

-v value
Inverse interpolation. Shows for which x the spline assumes the given value: $S(x)$ = value.

-a
Shows the value of $\int_{x_1}^{x_N} S(x)\,dx$.

-h [-b value]
Shows for which x the spline assumes a value half of the peak height, assuming the data have a peak-shaped trend. Evalsp first attempts to estimate a horizontal baseline and subtracts its value from the data. Optionally one can add the –b option to force the baseline at a certain value.

6. Numerical methods

In this section we discuss the mathematical details of the Durbin-Watson test as implemented into spline2. We also briefly explain the steps by which spline2 finds the spline that it considers best.

The original Durbin-Watson test [2] is used on a statistic that we call $Q(1,0)$. It is defined as

$$Q(1,0) \equiv \frac{N-1}{N}\frac{\left\langle (d_{i+1}-d_i)^2 \right\rangle'}{\left\langle d_i^2 \right\rangle}, \tag{18}$$

where d_i is the weighted fit-residual in point i (given in Eq. (17)). The following notation for avarages is used:

$$\langle ... \rangle \equiv \frac{1}{N}\sum_{i=1}^{N}(...), \tag{19}$$

$$\langle ... \rangle' \equiv \frac{1}{N-n}\sum_{i=1}^{N-n}(...), \tag{20}$$

where in this case (Eq. (18)) n is equal to 1. In essence, $Q(1,0)$ is a ratio of two estimates of the residual variance, one based on the magnitudes of the point residuals, and one based on the serial correlation between them. The important thing to note is that since it is a *ratio*, $Q(1,0)$ is totally insensitive to any common factor by which the weighted residuals may be incorrect. Eq. (17) shows what this implies. It makes clear why only the relative values in the estimates of the data uncertainties s_i matter (Eq. (13)), not the absolute values. This is *the* great advantage of the Durbin-Watson test, since the outcome is much more insensitive to user misjudgements of the noise magnitude than if the χ^2-test would be used. Durbin and Watson have calculated the statistical properties of $Q(1,0)$, and it is on these calculations that the statistical tests in spline2 are based. This is less straightforward than it seems, because *spline2 uses a more generalized version of the statistic than Q(1,0)*, namely one that takes into account autocorrelation effects. To see how autocorrelation comes in, we first rewrite Eq. (18) as

$$Q(1,0) \equiv \frac{N-1}{N}\left[\frac{\left\langle d_{i+1}^2 \right\rangle' + \left\langle d_i^2 \right\rangle'}{\left\langle d_i^2 \right\rangle} - 2C(1,0)\right], \tag{21}$$

with

$$C(1,0) \equiv \frac{\langle d_{i+1} d_i \rangle}{\langle d_i^2 \rangle}. \tag{22}$$

For uncorrelated data the expectation value of $C(1,0)$ is zero, which is a statement similar to Eq. (4). In fact, for *any* n the expectation value of $C(n,0)$, defined as

$$C(n,0) \equiv \frac{\langle d_{i+n} d_i \rangle}{\langle d_i^2 \rangle}, \tag{23}$$

should be zero for truly white noise. Next, this quantity is generalized for the case of correlated data, in the spirit of Eq. (13), and we introduce

$$C(n,\Delta) \equiv \frac{\langle d_{i+n} d_i \rangle}{\langle d_i^2 \rangle} \cdot \langle \exp(-(x_{i+n} - x_i)/\Delta) \rangle, \tag{24}$$

which, again, is expected to be zero for a good fit (of course, any other assumed autocorrelation function may take the place of the exponential function in Eq. (24)). Inserting this expression into Eq. (21) we find as generalized Durbin-Watson statistic

$$Q(n,\Delta) \equiv \frac{N-1}{N} \left(\frac{\langle d_{i+n}^2 \rangle + \langle d_i^2 \rangle}{\langle d_i^2 \rangle} - 2C(n,\Delta) \right). \tag{25}$$

But we are not done yet. The quantity $Q(n,\Delta)$ refers to one n value only, that is, the actual autocorrelation function is compared with the assumed autocorrelation function on the basis of only one value for the data index spacing. To obtain a comparison over a wider range of the autocorrelation function, we take the average over several n values and arrive at our final generalized Durbin-Watson statistic $q(\Delta)$:

$$q(\Delta) \equiv \frac{1}{n_{\max}} \sum_{n=1}^{n_{\max}} Q(n,\Delta). \tag{26}$$

In the spline2 algorithms the value of $n_{\max}$ is taken large enough to let n cover the most interesting range of the autocorrelation function,

$$n_{\max} = \mathrm{int}\left(3 \frac{\Delta}{\langle \Delta x \rangle} + 3 \right), \tag{27}$$

where the average data spacing $\langle \Delta x \rangle$ is obtained from Eq. (15).

The parameter dws listed in the output of spline2 is equal to $q(\Delta)$ when the –s or –K options are used. When, instead, the –n option is used, dws is equal to $Q(n,\Delta)$ (where Δ defaults to 0 if –K is absent). When –a or –r is used, dws is equal to $Q(1,0)$.

With this background we can now explain how spline2 does its job (when the –s option is on):

I. For a range of Δ values, from $\Delta = 0$ to $\Delta = 3\sigma_x$, in increments of $0.5\sigma_x$, spline2 executes the following steps (details can be found in Ref. [3]):

1. A series of trial "equal-information" splines with gradually increasing numbers of knots is fitted to the data in the least-squares sense. For each fit $q(\Delta)$ is calculated.
2. As soon as (a tolerant version of) the Durbin-Watson test performed on $q(\Delta)$ indicates that the fit is acceptable, the spline of this fit is considered a good first approximation.
3. A second series of trial splines is fitted, starting from the first approximation just calculated, but this time with gradually *decreasing* numbers of knots *and* with a breakpoint optimization algorithm (Ref. [1]) activated.
4. Of all the splines of this series that are deemed acceptable by (a strict version of) the Durbin-Watson test, the spline with the smallest number of knots is considered optimal for the particular value of Δ under investigation. Side step: if none of the splines in this series is deemed acceptable, the spline series of step 1 is extended by one more spline and the process is repeated.
5. The parameter acffit is calculated for the optimal spline found in step 4. This parameter measures how closely the residuals of the spline fit agree with the assumed autocorrelation function. It is calculated according to

$$\text{acffit}(\Delta) = \left|\min_n C(n,\Delta)\right| + \left|\max_n C(n,\Delta)\right|, \tag{28}$$

where all n values in the range $[1...n_{\max}]$ are considered, see Eq. (27).

II. After the loop over Δ is completed, the values acffit(Δ) are examined. The smallest of these is considered to represent the best corrspondence with the assumed autocorrelation function. The Δ value that gave rise to this smallest acffit is then selected as the most probable value, and the corresponding optimal spline is finally presented to the user as the overall best.

7. References

[1] C. de Boor, *A Practical Guide to Splines* (Springer, New York, 1978).
[2] J. Durbin and G.S. Watson, Biometrika **37** (1950) 409; Biometrika **38** (1951) 159; Biometrika **58** (1971) 1.
[3] B.J. Thijsse, M.A. Hollanders, and J. Hendrikse, Computers in Physics **12** (1998) 393.

Appendix: compilation note

Compilation of spline2 and evalsp is straightforward. Assuming that cc is the name of your C-compiler, just type:

```
cc –o spline2 spline2.c –lm
cc –o evalsp evalsp.c –lm
```

No special libraries are needed. An additional code optimization flag, such as –O2, is recommended. Note: Don't be alarmed if the loader complains about a "multiple definition of the symbol _lgamma". On some Unix and Linux machines the mathematical function lgamma, which is part of the spline2 code, is *also* defined as a system function.