NAG Spline Fitting Routines on a Graphics Workstation – the Story so far

Tim Hopkins
Computing Laboratory
University of Kent
Canterbury, CT2 7NF
Kent, U.K.

March 29, 2001

1 Introduction

One question which seems to recur with alarming regularity at our numerical advisory desk is *"How do I fit a nice smooth curve through this set of data points?"* Unfortunately most users seem to think that computer software for curve fitting should know exactly what the user expects the fitted curve to look like and provide it – *By the best fit I mean exactly what I want – neither more nor less!*

Borrowing some phraseology from *1066 and All That*, most of our users have learned from somewhere that

- Spline fits are a good thing,
- Interpolation is a bad thing,
- The NAG library is a good thing.

So most are reasonably happy when pointed to the relevant spline fitting routines within the NAG E02 chapter. Prior to Mark 13 the library contained four routines concerned with B-spline approximations. E02BAF computes, given sets of data points and knot positions, a weighted, least squares, cubic B-spline approximation. This approximation may then be evaluated using E02BBF; if derivative values are also required E02BCF may be used. Finally E02BDF computes the integral of the spline approximation.

The major problem facing the user (once she had fathomed the mysteries surrounding NCAP7) is almost invariably the choice of knot positions. This has been partially alleviated at Mark 13 by the addition of the routine E02BEF which attempts to place the knots automatically although this is by no means the answer to all users' prayers. The chapter introduction provides some help, although it is often the case that the advice does not always produce the desired result.

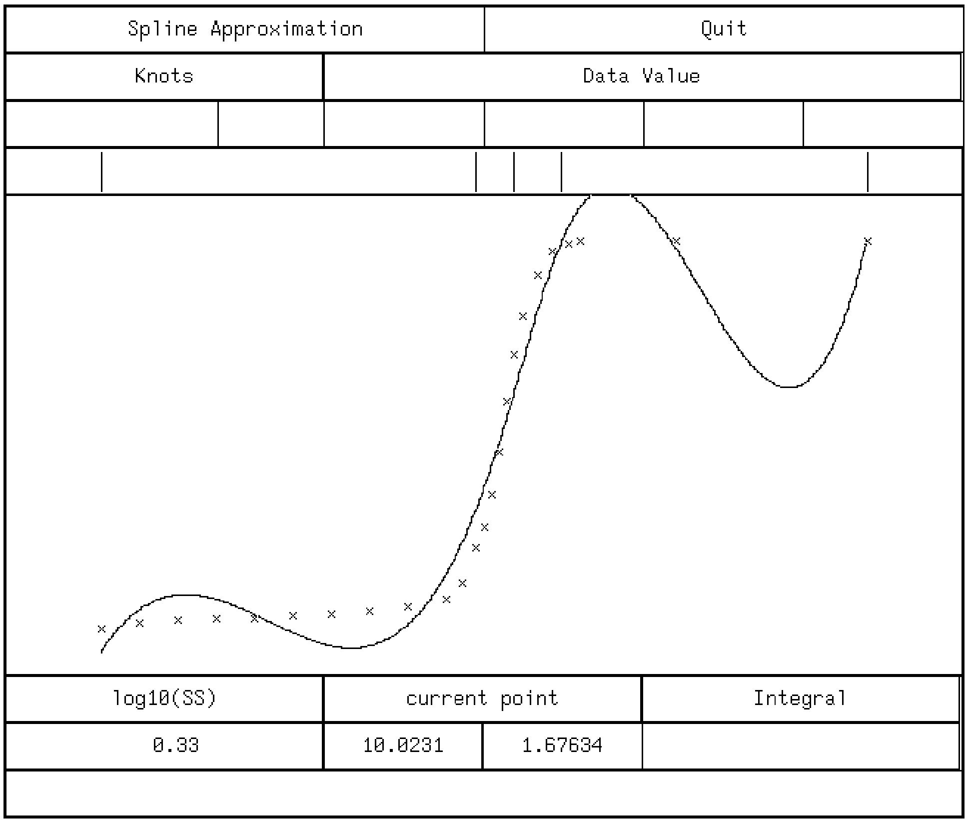

Figure 1: Knots chosen as in the E02BAF example program

For example, consider the data and spline fit shown in Figure 1; this shows the effect of choosing the knot positions in a similar way to that used in the example program for E02BAF. The resultant curve is obviously far too 'wiggly'. Following the advice in the introduction, more knots are

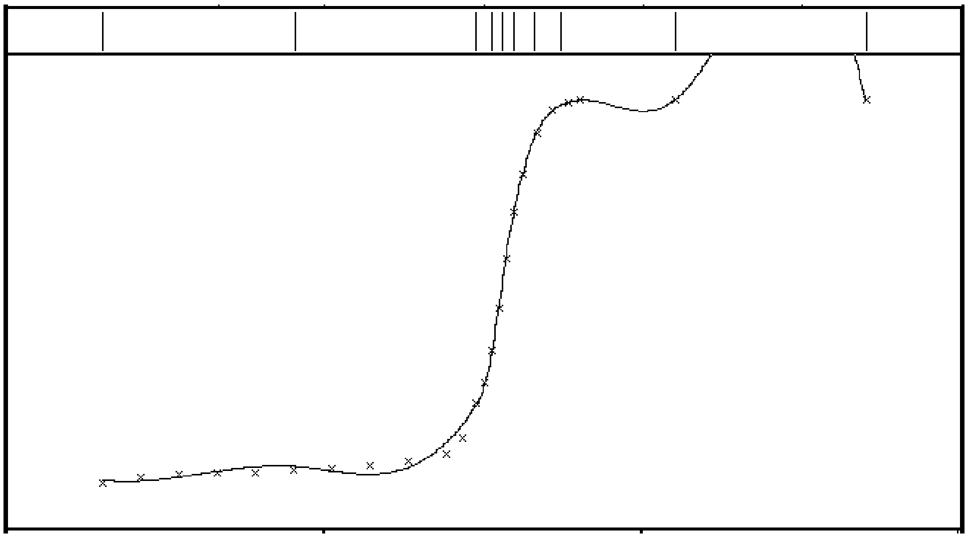

Figure 2: Knots added in region where data changes rapidly

placed where the data changes rapidly; the effect is illustrated in Figure 2. This certainly smooths out the curve at the left-hand end at the cost of making things worse at the right. Placing an additional knot in the area of poor fit (again as advised in the introduction) makes things even worse (see Figure 3). Adding more knots in this region would cause the fitting routine to exit with `IFAIL` set to 5 – the dreaded Schoenberg-Whitney condition fails to hold. As a desperation measure, Figure 4 shows the effect of choosing equally spaced points.

Even though the data set chosen here may be described as pathological, the figures illustrate a major problem with this type of fitting procedure. The user often has a clear idea of what she wants the final approximation to look like. In many of the cases we have seen there are additional constraints on the fitted function which cannot be specified directly to the routine; for example, the final fit may need to be a monotone increasing function. Such constraints may often be accommodated by a judicious choice of knots; the main cause of frustration to the user is often inadequate feedback. She has been placed in a loop of the form

while (not what I want) do {
1. Edit the number and/or position of the knots
2. Rerun the program
3. Display the resulting approximation

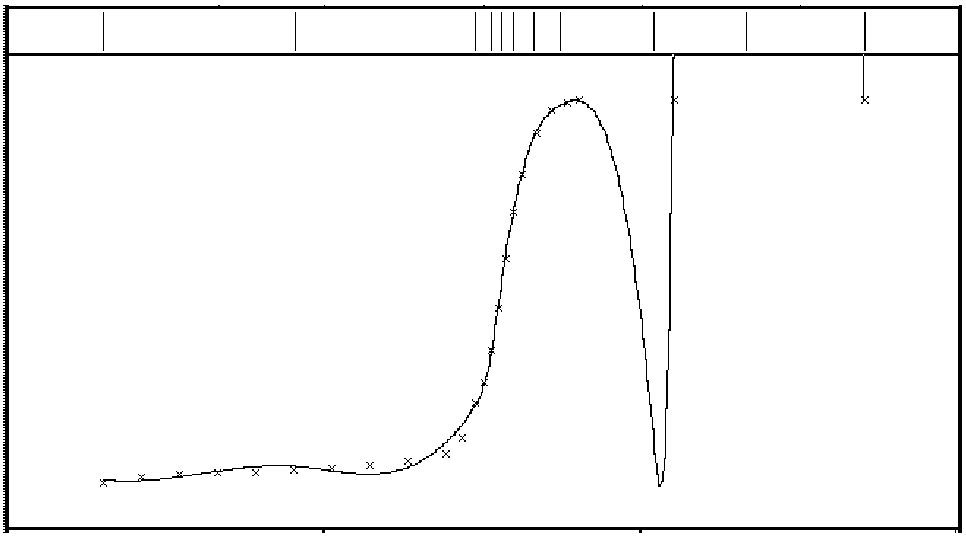

Figure 3: A further knot added where the fit is poor

4. Compare with the previous result

}

where, on commonly available plotting devices, the comparison at step 4 is very difficult to interpret. Certainly the effect on the approximating curve of moving, inserting or deleting a knot is not intuitive to relative experts let alone end users. This problem led to the development of a prototype system to investigate how a more interactive approach to knot placement could be achieved.

2 The Prototype System

The first attempt at such a system was made in 1985 using an ICL Perq workstation. This was part of the work undertaken by the Kent Software Tools Centre, a research group funded by the SERC as part of their Common Base Policy to look at the possible benefits to the SERC research community offered by the then emerging workstations. At this stage the project amounted to no more than a feasibility study. The first system offered the following facilities

1. It could read in, from a file, a set of data points and knot values and display the resulting approximation,

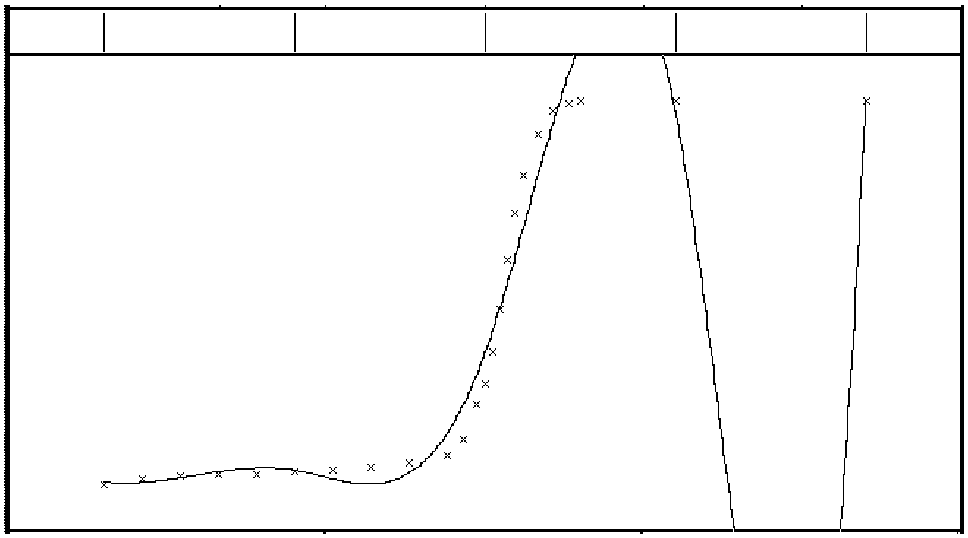

Figure 4: Equispaced knots

2. It would allow the user to select a knot using the mouse and drag it; while the knot was being dragged the program would recompute and redisplay the approximation. It was hoped to be able to recompute and redisplay in real time,

3. It would allow the user to insert, delete and, for the more ambitious user, change the multiplicity of knots.

With this basic set of facilities it was hoped that a non-expert could very quickly position knots so as to obtain a final approximation with the desired properties.

The system was written in C using the *ww* library (a set of windowing routines supplied by the SERC as part of the Common Base) to implement the windowing, graphics and user interaction via the mouse. The NAG routines were called directly from C.

The major problem was that the program ran extremely slowly; even so, the ability to have the approximation updated as a knot was moved was found to be very useful in determining just how much effect that knot had on the currently displayed fit. As there were no profiling capabilities available on the Perq it was very difficult to ascertain just where the computational bottleneck was. One attempted improvement was to recode the NAG routines in C; this produced a slight increase in speed but the effect of moving a knot still produced a very jerky display.

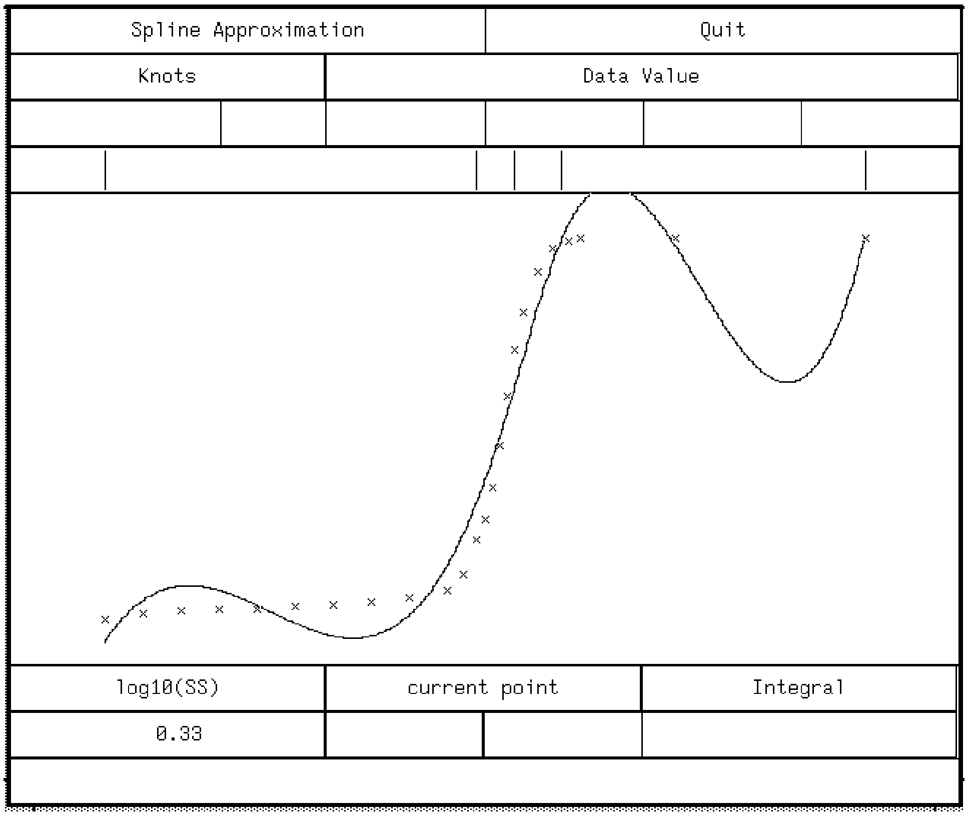

Figure 5: Basic screen layout

At this point the project was put into mothballs.

3 The Second Phase

During 1987 I came across the following quote from Pat Gaffney [1],

> In my opinion we are at a fairly critical stage in the development of numerical software and the acceptance of this software by a broader community of users. We must begin to produce software that includes graphical capabilities, we must experiment with how best to use these techniques in the intelligent workstation environment ... in order to communicate good numerical techniques to users.

He then goes on to suggest that the way forward is to relax the insistance on portability and to produce software tailored to particular systems. Reading this article coincided with the arrival of a Sun 3 at Kent with a 68881 maths coprocessor and, since the *ww* library was also available, it was possible to run the same software on the new hardware (graphics software and portability!).

There was a vast improvement; redrawing on dragging a knot was very much faster and it was possible for the program to keep up with a reasonably slow, but steady, knot movement. It was now easy to experiment with knot positioning and the time taken for users to generate their final fits was reduced dramatically. Furthermore it was found that the system acted as an effective teaching aid; after experimenting with just a few data sets, users found that they were able to select good initial knot placements.

The system was extended at this stage to provide a number of extra facilities. The coordinate of the cursor when placed within the plotting area was displayed along with the log of the sum of squares of the residuals for the current approximation. The routines E02BCF and E02BDF were also incorporated for evaluating the fitted curve and its first three derivatives at a user selected point along with the integral of the approximation. An option was also provided to allow the input data to be interpolated.

Figures 5, 6 and 7 show three views of the system. The first illustrates the layout of the screen and shows the user data marked by crosses and the current spline fit. The second shows the pop-up menu used to change the state of the knots, and the third the result of interpolating the data and evaluating the curve at the currently selected point along with the value of the integral.

The arrival of a DECStation 3100, with the promise of even faster floating-point computation, produced the first portability problem for the newly named PONS (Pictures of NAG Splinefits or, if you prefer, Go Away NCAP Seven) system. This machine ran X and did not come with the *ww* interface library. However a locally produced windowing library, *wn*, was available and this had been implemented for Sunview, X10 and X11; rewriting the window and mouse handling code within PONS allowed portability between Sun and Dec equipment. The X server on the DECStation produced a good deal of flicker when redrawing the fitted curve during knot dragging and to rectify this it was necessary to rewrite the plotting routines to perform much smoother updating.

The increased power of the 3100 (and, more recently, SparcStations) allows knots to be dragged quite quickly producing a cartoon effect in the resulting approximations.

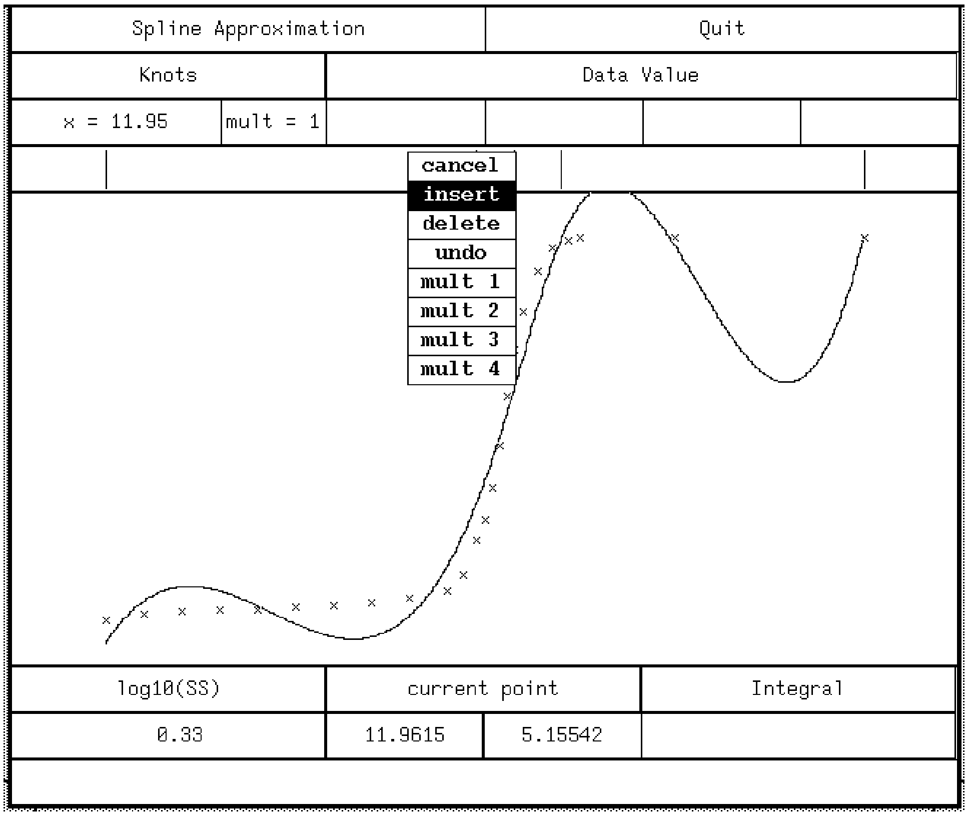

Figure 6: Menu for knot changes

4 The Future

A design for a new version of PONS is already underway; this is a collaborative effort between the University of Kent, NAG and the IBM Bergen Scientific Centre (Pat Gaffney and Debbie Lloyd).

The new interface will be based on Motif (a widget set running on top of X) and will provide all the facilities available in the prototypes plus much more. The sort of extensions currently being considered include automatic knot placement (for example, using the E02BEF routine introduced at Mark 13), the interactive addition and deletion of data points and the generation of high quality hard copy (possibly by generating a Fortran program which calls NAG Graphics routines). Of necessity this will require some rewriting or new coding of the approximation routines in order to economise the linear algebra involved when changing knots. This research is underway.

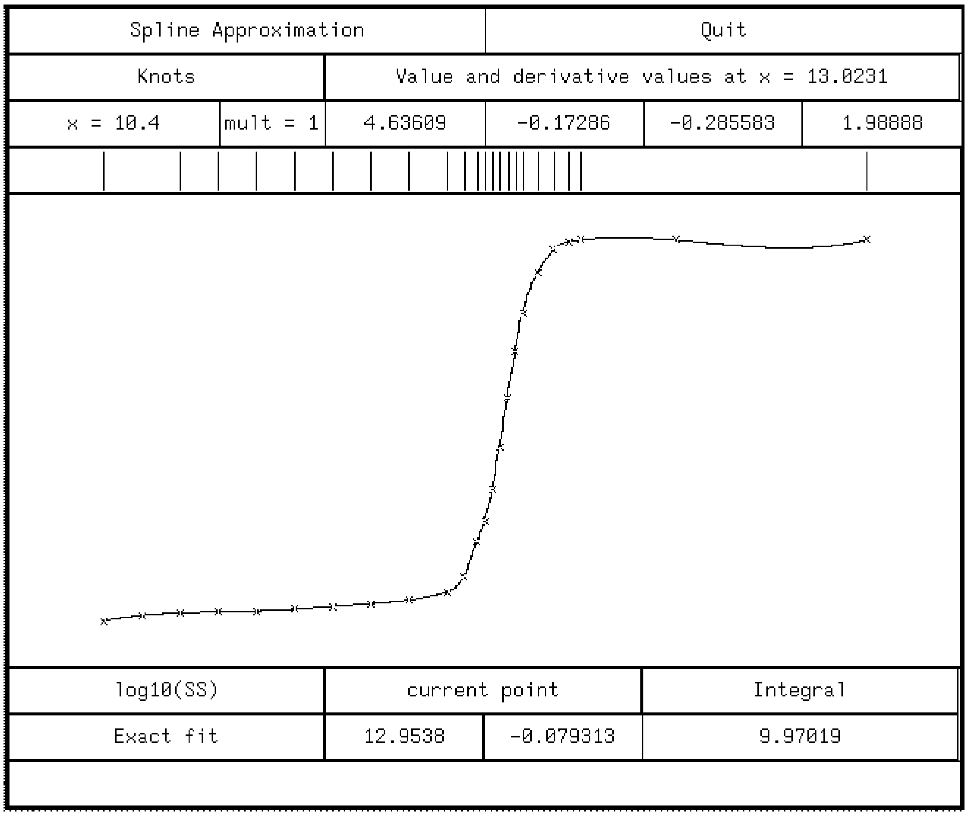

Figure 7: An interpolating curve

The recent rapid take-up of X as a de facto standard has really drawn the sting from the portability problem. This, along with the computational power of the current generation of workstations, opens the door to a new generation of interactive numerical software.

References

1. Gaffney, P. W., *When things go wrong*, IBM Bergen Scientific Centre, Bergen, Norway, Report No. 87/1, August 1987.