Approximating splines and the representation of scattered and not-so-scattered data

Alternative Hydraulics Paper 7

John D. Fenton

St-Ulrichs-Platz 2/4, 1070 Vienna, Austria

http://johndfenton.com/Alternative-Hydraulics.html
mailto:JohnDFenton@gmail.com

Saturday 12th September, 2015

This document describes a computer program, its background theory, and its use to approximate more-or-less scattered data, or to smooth, differentiate, interpolate, or calculate an envelope to that data. Piecewise local polynomials, namely quadratic or cubic splines, are used, which overcome some characteristic problems of global approximation. A number of practical considerations are described, then the results of several applications of the program are shown. It seems to be quite flexible and powerful. Finally, in an appendix, the files necessary and instructions how to use the program are described.

This report is: Fenton, J. D. (2015) Approximating splines and the representation of scattered and not-so-scattered data, Alternative Hydraulics Paper 7, http://johndfenton.com/Approximating-splines/Approximating-splines.pdf

The program and supporting files may be accessed here:
http://johndfenton.com/Approximating-splines/

Contents

1 Introduction

Consider a problem where we have a number of data points and we want to obtain a function which represents those points, and possibly even derivatives of such a function. There are two main approaches that we can take. One is *interpolation*, finding a function that passes through all data points. The other is *approximation*, obtaining a function that passes through the assembly of the data points, approximating them in a least-squares sense such that the parameters of that function, for example the coefficients in a polynomial, minimise the sum of the squares of the differences between the function and the data points. Figure 1 shows that, in general, approximation of point data is to be preferred to interpolation. Later in this work we will develop a method which can almost do both operations, depending on the number of degrees of freedom of the approximating function. In the limit as that number approaches the number of data points, we call it quasi-interpolation.

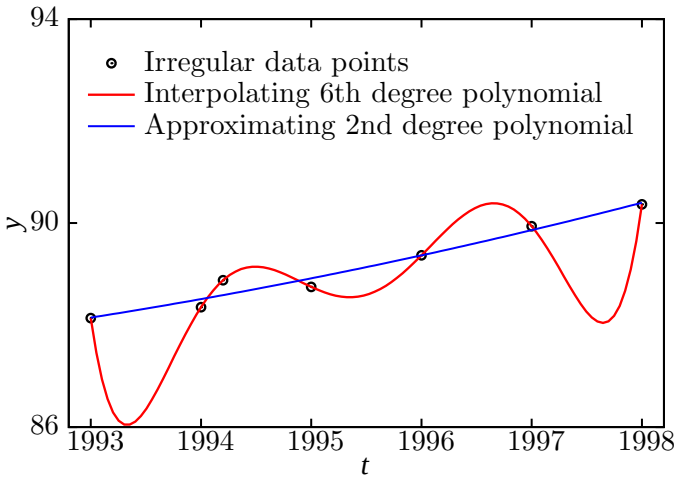


Figure 1: The difference between interpolation and approximation for data with any irregularity

In this document below we first consider some of the problems of global representation and how they can be overcome by using piecewise-continuous polynomials such as quadratic or cubic splines. Then we present a theory of approximating splines, not to interpolate as is usual, but to approximate data which might be scattered. A number of practical considerations are described, then the results of several applications of the program are shown. Finally, in an appendix, the files necessary to use the program are described.

In conventional global approximation, the end result might be a polynomial which could be written as a single equation. The output of the present program, however, includes the boundary points of a series of intervals, say 3-7, or even 20 in one application described, with 3 or 4 spline coefficients for each. The results here are not expected to be encapsulated in a single formula, but are output to a file, some of which could be used as a data file for other software, while others are the approximating values and derivatives at a number of points which could be used for plotting.

1.1 Problems of global representation

Often a single *global* function is used for either interpolation and approximation, such as a polynomial composed of a number of monomials x^m, which is valid over the whole interval $[x_{\min}, x_{\max}]$, required to represent the data points (x_i, y_i) for $i = 1, 2, \ldots, N$, say:

$$f(x) = \sum_{m=0}^{M} a_m x^m = a_0 + a_1 x + a_2 x^2 + \ldots + a_M x^M. \tag{1}$$

This global formulation has some problems which can destroy the accuracy of interpolation or approximation. Two aspects are:

Apparent similarity of basis functions leading to poor conditioning of results: Figure 2 shows the behaviour of three monomials of first, second, and third degree for two different intervals in x. In part (a) of the figure, on the interval $[1000, 1200]$, the monomials apparently show a similar behaviour to each other – they look almost like straight lines. This means, to approximate a function or data with curvature, the coefficients a_m in equation (1) have to be large, such that there is poor convergence in the degree M of the polynomial, and high accuracy is necessary in specifying and using the coefficients. On the other hand, part (b) of the figure shows that over the interval $[-1, 1]$ the monomials x, x^2, x^3 show diverse behaviour and are better able, with smaller coefficients, to interpolate or approximate a function that varies arbitrarily. Hence, in global representation it is always a good idea to scale the independent variable to $[-1, +1]$ or possibly $[0, 1]$ and to use that in calculations. Not doing this can have severe consequences, as shown by Fenton (1994), especially in civil engineering problems, where the numerical values of x might be huge, corresponding to distances along a road, railway or river, or as the author has seen, where a river height is specified in centimetres. Even better for higher degrees of approximation, when the monomials on $[-1, +1]$ also begin to look alike, would be to use Chebyshev polynomials, which have a strong orthogonal nature, each having different properties from the others.

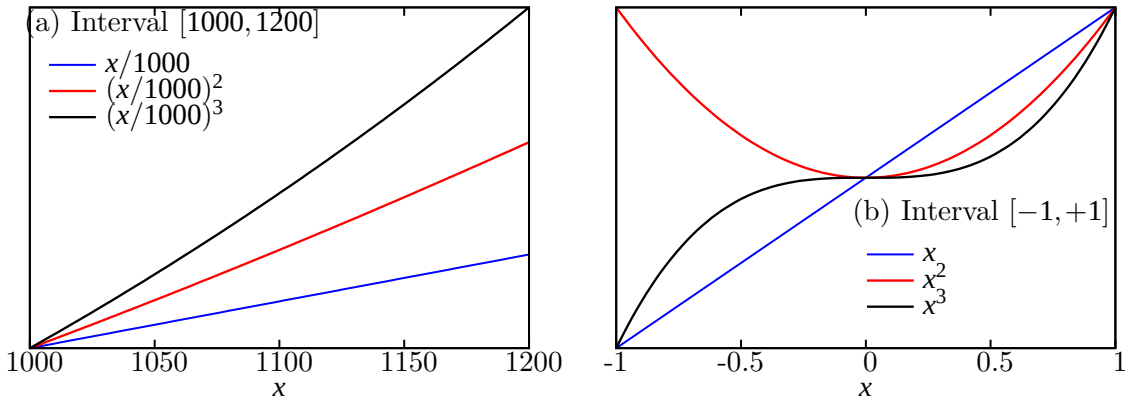
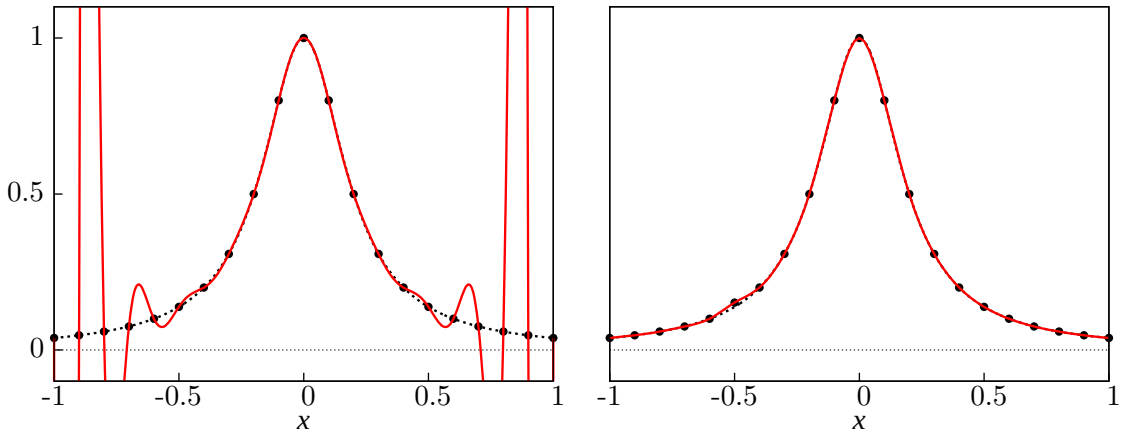


Figure 2: Comparison of variation of mononomials on the interval $[1000, 1200]$ and on $[-1, 1]$, showing the greater diversity of behaviour in the latter case, with a better ability to represent data or functions

It is worth mentioning here that this problem of poor conditioning of a polynomial formulation can be almost accidentally invoked. There is a facility in the spreadsheet *Excel* by which "trendlines" can be very simply added to sets of experimental points, which are none other than approximations to data using a relatively low-degree function. Generally the program is very robust, and the plotted curves are accurate. Various functions are available, in terms of only two or three coefficients, but for polynomial approximation as many as 6 coefficients are possible, which is more useful. Often one needs the actual function which *Excel* has obtained, and that can be displayed on the figure as well. However, that facility has a flaw, in that however robustly the program internally calculates the approximating function, quite possibly using a scaled interval as mentioned above, it displays the formula in expanded form such as equation (1), with the problems that entails, so that round-off errors might be large. One can change the number of digits shown, but the problem remains.

Region(s) of rapid variation: Another problem for global approximation by a function such as a polynomial is that if the data to be interpolated or approximated has a region of rapid variation, then because the global function has to approximate that region, and elsewhere, the interpolation or approximation can be poor. This is known as *Runge's phenomenon*, and the consequences can be very serious and surprising, such that increasing the degree of approximation can simply make the problem worse. Figure 3(a) shows this dramatically for the global polynomial approximation of a function $1/(1+25x^2)$ (on the previously recommended interval of $[-1, +1]$!) devised by Runge to show that the region of rapid



(a) Polynomial interpolation (b) Spline interpolation

Figure 3: Interpolation of Runge's function $1/(1+25x^2)$ with a sharp crest using 21 data points. For the spline case, a data point at $x = -0.5$ was raised slightly – the effects are quite localised.

variation near the crest has destroyed the accuracy in the slowly-varying region away from the crest. A series of Chebyshev polynomials would be no solution to the problem, because they would give exactly the same results. We now consider the method whose much better results are shown in part (b) of the figure.

1.2 Piecewise-continuous interpolation – cubic splines

It is well-known that for interpolation, piecewise-continuous splines can be used to overcome both problems described in the previous section. They are a sequence of low-degree polynomials, with a high degree of continuity of function and derivatives at data points, but not complete continuity. Using them it is no longer necessary to scale the independent variable, and the effects of rapid variation are more limited to the region in which it occurs. Figure 3(b) shows the results for Runge's problem using cubic spline interpolation. The results are excellent, even in the region of high curvature near the crest. To further demonstrate the power of the method, a data point at $x = -0.5$ was raised by 10% to make the data slightly irregular. It can be seen that the effects are highly localised, reflecting the largely local nature of the spline interpolation.

The method is described in many books (for example, Conte and de Boor, 1980, and de Boor, 2001) and included in software packages. The physical interpretation and the name of cubic splines is familiar to civil engineers, for it comes from a draughtsperson's flexible strip or "spline" which can be used to fair smooth curves between points. If the strip is held in position at various points by pins, then between any two of those pins there are no lateral forces acting so that the shear force in the strip is constant, the bending moment varies at most linearly, and hence by beam theory (for sufficiently small deflections) the strip takes on a cubic variation between the two points. As the variation of moment is different between other points, other cubics will apply there. However, because the shear force and bending moment are continuous across each pin, then the first and second derivatives are continuous across the pins, or interpolation points. With four unknown coefficients for the cubic in between each pair of points, and the requirement that each of the two cubics, to left and right of each interpolation point, must interpolate at that point and must have the same first and second derivatives, almost enough equations are obtained for all the four coefficients of each cubic.

It is necessary, however, to specify two more conditions. This may be by specifying the slope at the two end points, as in Conte and de Boor (1980, Section 6.7), or by arbitrarily specifying that the second

derivative at both the end points be zero. This ”moment-free” end condition gives the so-called ”natural spline” approximation, which is the method traditionally adopted. In general, however, there is no particular reason at all why the first or the second derivative of the interpolating spline should be zero at the ends, and in almost all presentations and software, with the exception of de Boor (2001), the method suffers from this disadvantage and is not as accurate at the ends as it might be.

A better way to obtain the two extra conditions is to use the ”not-a-knot” condition at the first and last *interior* points (de Boor 2001), where it is required, in addition to the first and second derivatives agreeing to left and right of the interpolation point, that also the third derivatives agree. The physical significance of this is simply that a single cubic interpolates over the first two intervals and another over the last two intervals. No arbitrary assumptions have been introduced, and it can be shown that the error is much less than for ”natural” splines. The manner in which the fluctuations of the interpolating polynomial are held down has made cubic splines popular as a means of interpolating and obtaining derivatives numerically.

2 Piecewise-continuous approximation – quadratic and cubic splines

2.1 Introduction

Here we extend the use of splines from interpolation to approximation in a least-squares sense with a finite number of specifiable knot/node points, across which the function and some of its derivatives will be continuous, but where the functions are primarily determined by least-squares approximation. The node points can be spaced more closely where the data varies more rapidly. It is expected that not many such intervals will be necessary. We will call this method that of *Approximating Splines*, using low-degree spline functions and least squares approximation.

The method is a quite obvious one. However, the author has spent some effort searching for presentations and applications. It is mentioned enigmatically and briefly in obvious places on the Internet. There are a number of papers available, but the author has not found any with a presentation of the theory and method and an exploration of practical aspects of applying them. Many papers use the term *Regression Splines*, following statistical terminology. These all seem to accept that the method is obvious and then to pursue abstract and arcane areas of theoretical statistics, with no presentation of method or results for practical problems. Of papers that use the word ”approximation” rather than ”regression”, there are some with source code versions available in C, Fortran and MatLab, for example http://people.sc.fsu.edu/~jburkardt/m_src/spline/spline.html. But nowhere is the method explained or its performance described.

There is one variant of piecewise polynomial approximation which is described elsewhere, and that is *Smoothing Splines* (de Boor 2001, pp207–214). That approach is where no two values of the independent variable x_i are the same, one fits a spline over every interval $[x_i, x_{i+1}]$, $i = 1 \ldots N$, and calculates the error e as the sum of squares of errors at every data point, but adds a ”roughness” term to the quantity e to be minimised, which is the integral of the curvature, the second derivative of the splines, over the whole interval. For many data approximation problems the requirement that no two x_i be the same is an important limitation, and we will not consider it further.

When the author began this work, he also included a roughness term, minimising total curvature, but found that it was not necessary, and the Approximating Splines described here seem to have a natural tendency to minimise total integrated curvature.

If one pursues some statistical papers, one encounters much preoccupation with homoscedasticity, or the requirement that the variance of the points about the fitted curve be constant over the whole domain. Otherwise, it seems that the Gauss-Markov theorem shows that least squares as we use it will result

in bias of the results. We are going to going to ignore such considerations. Amongst others, we will consider a problem where there is more than a thousand-fold variation of the dependent variable, and presumably variance changes by a similar ratio. Excellent results are obtained for that problem and for all others we have considered. We will simply approximate in a least-squares sense.

2.2 Method

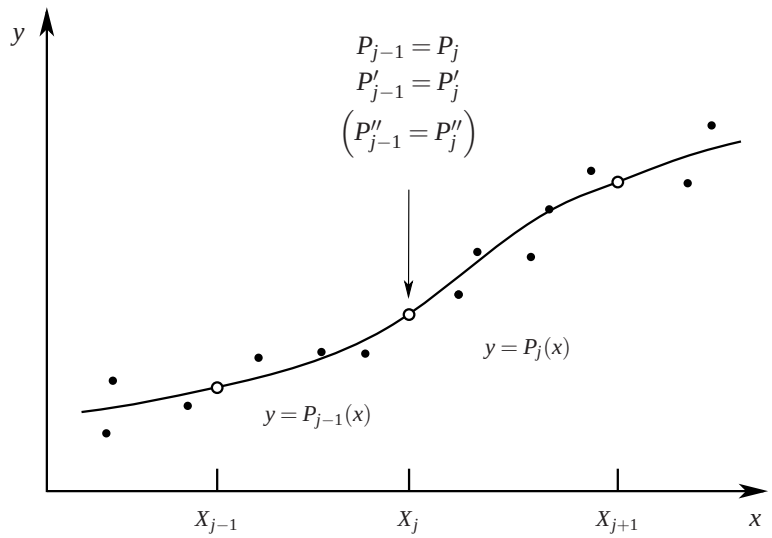


Figure 4: Spline approximation of scattered points – the vertical position of the knots, shown by open circles, is calculated by the program

Consider the scattered data such as shown in Figure 4 comprising a number of data pairs (x_i, y_i) with $i = 1, 2, \ldots$. Let the data interval $[x_{\min}, x_{\max}]$ be subdivided into J intervals by $J+1$ points, whose co-ordinates are $x = X_j$ for $j = 1 \ldots J+1$. We expect that these will be placed roughly in accordance with the rapidity of variation of the data. The X_j, however, are separate and ordered, such that $X_{j+1} > X_j$ for all the j. The points are to be approximated by a number of low-degree polynomials such as quadratics or cubics, $P_1(x)$, $P_2(x)$, …, $P_J(x)$, where the number indicates the interval over which the polynomial $P_j(x)$ is valid, between knot or node points at $x = X_j$ and $x = X_{j+1}$. Over each interval we have a polynomial of degree M, such that at over the interval j, $X_j \leqslant x \leqslant X_{j+1}$:

$$P_j(x) = c_{j,0} + c_{j,1}(x - X_j) + c_{j,2}(x - X_j)^2 + \ldots = \sum_{m=0}^{M} c_{j,m}(x - X_j)^m, \tag{2}$$

where in this case, M is expected to be only 2 or 3, giving quadratic or cubic functions. These are the same as the functions used in spline interpolation. It is not necessary to scale the x as we recommended in §1 as it only appears as the local shifted value $x - X_j$ and the degree of the polynomial is low.

The spline nature of our approximation now requires us to satisfy across each interior node the continuity of function value plus all derivatives up to $M-1$. Hence for all $j = 1 \ldots J-1$ we have

$$P_{j-1}(X_j) = P_j(X_j), \tag{3a}$$
$$P'_{j-1}(X_j) = P'_j(X_j), \quad \text{and if } M = 3, \tag{3b}$$
$$P''_{j-1}(X_j) = P''_j(X_j). \tag{3c}$$

From equation (2) at left and right, and using $\delta_j = X_{j+1} - X_j$ for the interval length, the continuity

conditions become

$$c_{j+1,0} = \sum_{m=0}^{M} c_{j,m}\delta_j^m = c_{j,0} + c_{j,1}\delta_j + \ldots + c_{j,M}\delta_j^M, \tag{4a}$$

$$c_{j+1,1} = \sum_{m=1}^{M} m c_{j,m}\delta_j^{m-1} = c_{j,1} + \ldots + M c_{j,M}\delta_j^{M-1}, \text{ and if } M = 3, \tag{4b}$$

$$2c_{j+1,2} = \sum_{m=2}^{3} m(m-1)\, c_{j,m}\delta_j^{m-2} = 2c_{j,2} + 6c_{j,3}\delta_j. \tag{4c}$$

Now, unlike Interpolating or Smoothing Splines, we do not use the conditions that the knot points are data points, $X_j = x_j$. Neither do we require, like Interpolating Splines, that each spline passes through the corresponding data point $P_j(X_j) = y_j$. Instead, we seek to approximate the data points such that the sum of the squares of the errors over all the points is minimised. We write this as a sum over all the intervals of all the contributory points in each interval, and where each data point is assigned a weight of w_i:

$$e = \sum_{j=1}^{J} \sum_{i \in I_j} w_i \left(P_j(x_i) - y_i\right)^2, \tag{5}$$

where we have used the mathematical notation $i \in I_j$ where I_j is the set of points which are in interval j, $I_j = \{i : X_j \leqslant x_i < X_{j+1}\}$ which simply means taking all the points i which are in interval j. This means that the contribution to the total error e of a data point is only given by the spline function on the interval in which it falls. Nevertheless it will affect the overall result by the continuity conditions at the ends of that interval.

We write e using the polynomial as given in equation (2) as

$$e = \sum_{j=1}^{J} \sum_{i \in I_j} w_i \left(\sum_{m=0}^{M} c_{j,m}(x_i - X_j)^m - y_i \right)^2. \tag{6}$$

Convenient aspects of this formulation are:

- The x_i can be in any order
- There can be multiple points with the same x_i, and
- The weights w_i can be assigned arbitrarily so as to attach less or more importance to a point or, with a large weight, to force the spline to go through or near that point.

We calculate how many unknowns we have. For each interval $j = 1 \ldots J$ we have $M+1$ values of the $c_{j,m}$ giving $J(M+1)$ unknowns or degrees of freedom. However, equations (4) enable us to eliminate M unknowns at each interior knot point $j = 1 \ldots J-1$ giving a total of $M(J-1)$ such unknowns so that the net number of unknowns is $J+M$. To be specific, these are all $M+1$ coefficients in the first interval: $c_{1,0}, \ldots c_{1,M}$ plus the Mth degree coefficients at each of the $J-1$ internal points $c_{2,M}, \ldots c_{J,M}$.

The problem is now to find all the $c_{j,m}$ such that e is minimised. We have two ways of proceeding. We could set up the *Normal Equations* by differentiating equation (6) with respect to each unknown $c_{j,m}$ and setting equal to zero for an extremum, which gives the same number of equations as unknowns. For global approximation such as a polynomial, the equations are famously poorly conditioned, and numerical solution can be quite difficult and not so accurate. While the spline formulation would lead to a diagonally-dominant matrix form, which would be more robust, here we adopt a rather more modern method and avoid the Normal Equations altogether, where we just use optimisation software to minimise e and determine the values of the coefficients. Such software is widely available, including the Solver module in spreadsheets. This method is rather simpler and more easily implemented.

3 Practical considerations and computer program

We have written computer programs to implement the Spline Approximation method, one in MAPLE, and one in C, giving an executable file *Approximating-splines.exe*. The detailed use of that program, available at http://johndfenton.com/Approximating-splines/ is described in Appendix A.

3.1 Modes of operation, number of intervals J, and placement of knots

There are different applications of the program, however they form something of a continuum, whether the number of intervals is very much less than the number of points, $J \ll N$, whether there might be not such a big difference between the two, or in the limit as $J \to N$, which we call *quasi-interpolation*:

Approximation of scattered data, $J \ll N$: one has to have enough intervals so as to be able to describe the local variation adequately, but in general the fewer the number of intervals the better so that there are more data points in each interval to better define the local quadratic or cubic.

The program does a very good job of approximating whatever it is given, but sometimes this is too good, for example when in one interval there might be as few as 2 or 3 data points, the local quadratic/cubic obligingly tries to interpolate them, passing through or close to each point. This is not necessarily what one wants with scattered data, so that it is desirable to have a number of data points in each interval.

It is recommended for the first application of Approximating Splines to a problem that the default option be chosen, where the program places the knots automatically such that there are equal numbers of data points in each interval. Then, visual examination of the results might suggest the clustering of knots in regions of rapid variation, from which a **.knots* file could be prepared with values of the X_j modified by hand. In typical applications described below, values of $J = 3,4,5,6,7,\ldots$ intervals have been found to be usually satisfactory.

Near the ends of the data set, where there might be more-or-less isolated points, the program also tends to agree closely with the data, which in this case is a pleasant property, especially for stream-gauging data, where there may only be one or two points at the upper end.

Smoothed interpolation and differentiation: here we consider an application of the program where one might have a number of discrete data points lying almost on a continuous curve, such as if they had been scanned and digitised, or if we wanted to differentiate the data, when numerical differentiation would give irregular results. We approximate the points to give a smooth continuous curve that can be used for plotting, calculation of function values, or even differentiation. The process is not quite interpolation, it is not quite approximation, but a combination of the two, for which we have given it the name ”Smoothed interpolation”. Of course, it is not really different from the approximation problem, just that the data is not as scattered. In this case it is probably necessary to have a larger number of intervals so that approximation is more precise.

Quasi-interpolation, $J \leqslant N-3$: The question arises, what happens if we let every data point be a knot such that $J = N-1$? Can we use the program to interpolate a set of data points? The answer is ”no”, for now there are too many degrees of freedom. It would be analogous to using a conventional spline interpolation program but without the two extra equations required, as described above. In fact, the present program almost works in that under-specified manner, except over the last two or three intervals. It could be modified to include extra continuity conditions. However, as we are mainly concerned with approximation we will not do this. Instead, if we simply use two fewer knots than data points, that is, we choose $J = N-3$, or slightly fewer than that, the program seems to work well. Figure 10 below shows an example – *almost* interpolating every data point, including the two we have chosen not to be knots.

3.2 Performance

The program has worked remarkably well in all the examples the author has considered. It has never failed badly. Sometimes adjustment of the knot positions is necessary to describe regions of high curvature or to avoid having too few points in an interval.

3.3 Quadratic or cubic

It does not seem to matter very much whether quadratic or cubic splines are used. The method is quite robust, with a minimal tendency for oscillations to develop. One may as well use quadratic splines, as there are fewer unknowns and computation time is shorter.

3.4 Computation time

On a personal computer built in 2011, using the Windows operating system, approximating the Nikuradse data in §4.1 with $N = 232$ data points, using quadratic splines and $J = 7$ intervals, with results shown in Figure 5, computation time was 0.1s, and for $J = 20$, 0.9s (the results for the latter were not as good as for $J = 7$, as with fewer data points in an interval, as described previously, the splines do a faithful job of approximating them to the point of almost interpolating them, and this gave irregularities in the result). These times were typical for other problems too.

The author often uses cubic spline interpolation to process the results of time-stepping solutions of partial differential equations, so as to be able to plot results at every few steps and possibly obtain derivatives. For such applications, with hundreds of interpolations necessary, the present approximation method would be too slow. Fortunately in those problems with smooth data, conventional spline interpolation works well. For the stand-alone problems we study in this document, computer time was never a problem.

3.5 Imposing conditions

There is a simple way of requiring the approximating spline to pass through (or near) a particular point, by adding that point to the data file and giving it a large weight, such as 10. Conversely, of course, the importance of any data points can be minimised by reducing the weight of that point or giving it a zero weight. If weights are not specified, they are assumed to be 1. There seems to be no simple way of using the present program to include information about first or second derivatives at any point.

4 Examples of applications

4.1 Nikuradse's results for resistance in pipes with uniform boundary roughness

Figure 5 shows Nikuradse's 1933 results (English translation: Nikuradse 1950) for resistance in pipes with uniform boundary roughness. The present author has scanned and digitised the data from figure 11 in that work, as there are errors in that table 7. For the abcissa the logarithm of the grain Reynolds number $\log_{10} R_*$ is used, where $R_* = \log(u_* d/\nu)$, u_* is the shear velocity $u_* = \sqrt{\tau/\rho}$, in which τ is the boundary shear stress and ρ is the fluid density, d is the grain diameter and ν is the kinematic coefficient of viscosity. The ordinate is the quantity introduced by Colebrook and White, $F = 2.0\log_{10}(3.7/\varepsilon) - 1/\sqrt{\lambda}$, where $\varepsilon = d/D$ is the relative roughness, in which D is pipe diameter, and λ is the Darcy-Weisbach

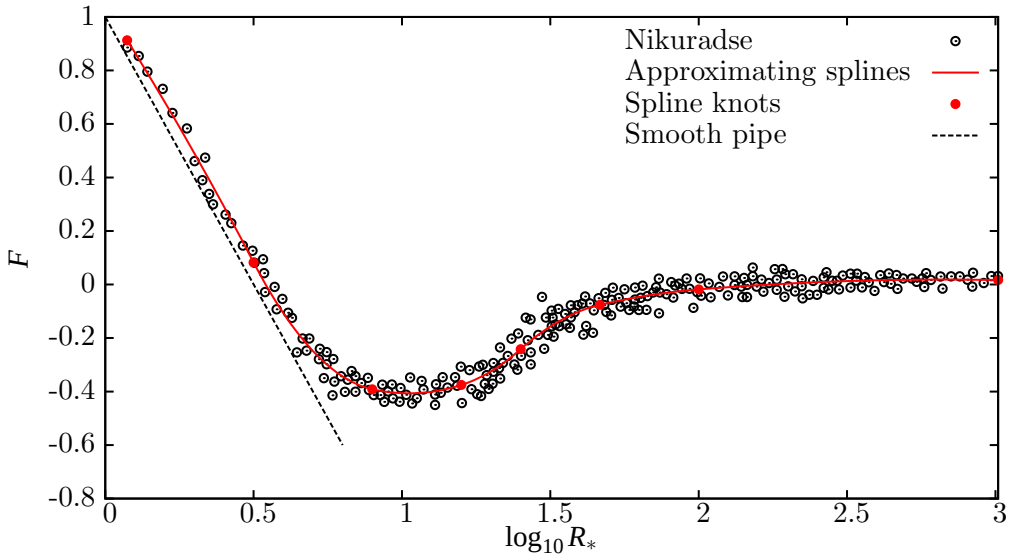


Figure 5: Nikuradse's results for F as a function of the logarithm of the grain Reynolds number

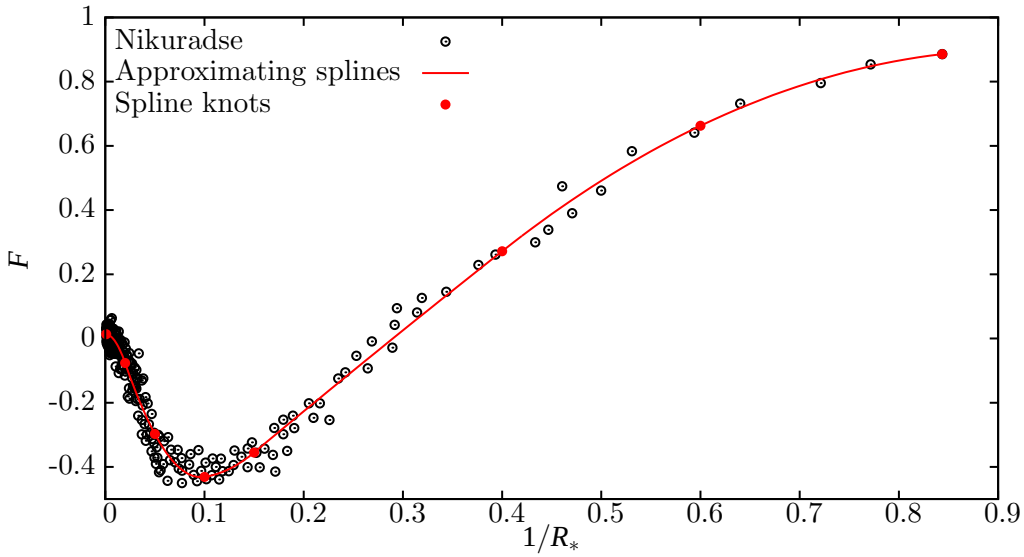


Figure 6: The results re-plotted in terms of the inverse of the grain Reynolds number

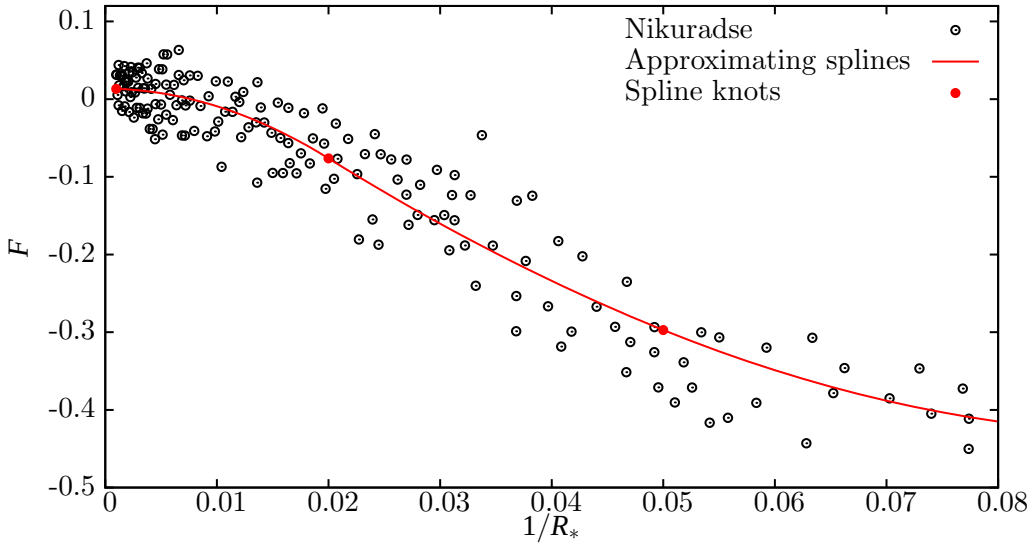


Figure 7: Enlargement of part of figure 6 in the large Reynolds number limit, $1/R_*$ small, for civil engineering flows

resistance coefficient. For fully-rough conditions $R_* \to \infty$, Nikuradse concluded that a quantity like F approached a constant value. Colebrook and White introduced the definition here such that in the fully rough limit $F \approx 0$, as suggested by the figure. The figure here shows that the spline approximation using $J = 6$ intervals and quadratic splines $M = 2$ works well.

To further examine that limit we try to do something different from conventional practice, where even using $\log_{10} R_*$ is difficult in the limit of infinite Reynolds number, and the use of the logarithm has no real physical justification. It makes sense to use another abcissa, and an obvious choice is $1/R_*$, as this goes to zero in the large Reynolds number limit, which is of rather more interest for hydraulic engineering. Possibly of greater importance, it has a physical significance, in that it is a dimensionless viscosity number, directly representing the importance of viscosity, as $R_*^{-1} = \nu/(u_* d)$. It would have been better had the Reynolds number itself originally been defined in this way.

The results are shown on Figure 6, and we get a very different appreciation – the region of small viscosity, typical of civil engineering flows, is really very small. It can be seen that the approximating quadratic splines seem to work well, with 7 intervals specified by a separate knots file as described above. Initially we tried using automatic allocation of knots, placing knots such that each interval had the same number of data points, but the results were not quite as good. At the sharp dip in the curve, the approximation is not completely satisfactory. We tried including more knot points there, but that just led to a small oscillation in the approximation as there were fewer data points in each interval.

To examine the limit of large grain Reynolds number further, we plot an enlarged version of the left of the figure, for small $1/R_*$, in Figure 7. The results are now more revealing, for it can be seen that F does not go to 0 but seems to go to about 0.01 in the fully-rough limit, as suggested in Figure 5, although that only went as far as $\log R_* \approx 3$.

There is quite a scatter, of about ± 0.03 on Figure 7, however this large Reynolds number limit is sufficiently important that we present the formula obtained by the program for the first interval. Here, however, we are going to be a little bit naughty, and in spite of warnings above against extrapolating and against expanding polynomials with shifted origins, we expand and assume that we can extrapolate to $1/R_* = 0$ beyond the minimum experimental value of $1/R_* = 0.00098$ (it is a very small amount) so that we write

$$\text{For } 1/R_* < 0.02, \text{ that is, } R_* > 50, \qquad F = 2.0 \log_{10}\left(\frac{3.7}{\varepsilon}\right) - \frac{1}{\sqrt{\lambda}} \approx 0.0135 - \frac{0.0453}{R_*} - \frac{222}{R_*^2}. \tag{7}$$

This raises a question, however, as to whether we should have imposed a zero derivative on the approximation in the limit as $1/R_* \to 0$. That is, in the limit of infinite Reynolds number, should the function be independent of Reynolds number? Our program cannot actually do that, in any case. However the second term on the right of equation (7) is so small in the range $0 < 1/R_* < 0.02$ that it suggests we can omit it and write

$$\text{For } 1/R_* < 0.02, \text{ that is, } R_* > 50, \qquad F = 2.0 \log_{10}\left(\frac{3.7}{\varepsilon}\right) - \frac{1}{\sqrt{\lambda}} \approx 0.0135 - \frac{222}{R_*^2}. \tag{8}$$

4.2 Rating curves in river hydraulics

The original stimulus for the development of Approximating Splines came from an important problem in river hydraulics, of approximating rating data for a gauging station. Relatively infrequently, hydrographers measure the flow velocities across a river and integrate them to give the discharge Q, while the water surface elevation h is also measured. With the results from many such stream-gaugings, one would like to establish a relationship between Q and h. From this, daily, hourly, or even more frequent automatic measurements of the surface height are then used to give the discharge at that station corresponding to each reading. Current practice in obtaining the relationship shows a surprising prevalence of arbitrary and laborious hand/screen methods. It is hoped that the present approach might give a means of automating the task.

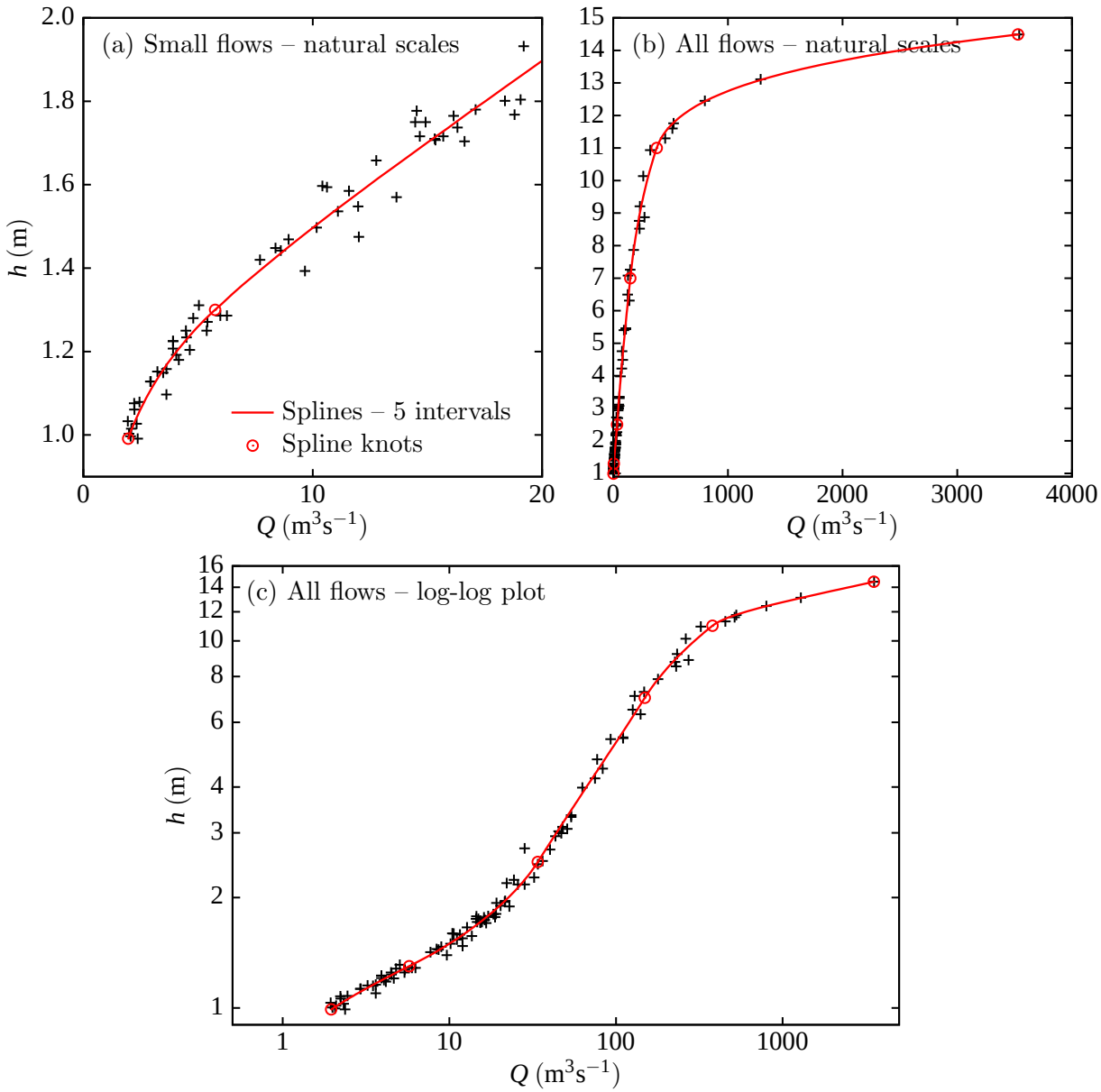


Figure 8: Results for the 1970s for Gauging Station 02448500 on the Noxubee River near Geiger, Alabama, USA

As an example, Figure 8 shows all data, roughly 100 points, from the 10 years of the 1970s for United States Geological Survey Station 02448500 on the Noxubee River near Geiger, Alabama. The approximating spline method was applied to give Q as a function of h, not by approximating data pairs (h_i, Q_i) but instead taking the square root of the discharges, $\left(h_i, \sqrt{Q_i}\right)$ and approximating. As the relationship between Q and h for small flows often looks approximately like $Q \sim (h - h_{\min})^2$ this makes the description of small flows more accurate. In all three parts of Figure 8 the results are shown, characteristically for all such rating curves, with the independent variable h or $\log h$ plotted vertically, and the dependent variable Q or $\log Q$ plotted horizontally. Figure 8(a) shows the results for low flows using natural scales; part (b) shows all the results on natural scales; while part (c) shows the results on log-log axes, commonly used in practice. It can be seen that the spline approximation is highly satisfactory, and is capable of approximating over the whole range of flows from $2\text{m}^3\text{s}^{-1}$ to $3500\text{m}^3\text{s}^{-1}$, something that is required in practice. Only 5 intervals were necessary, despite the more than thousand-fold variation in the dependent variable, the discharge. It was found that using the logarithms of h and Q to actually perform the approximation gave results which were no better. The method works surprisingly well. The author has a paper in preparation describing in detail the application of Approximating Splines to such rating data.

4.3 A different application – Smoothed interpolation

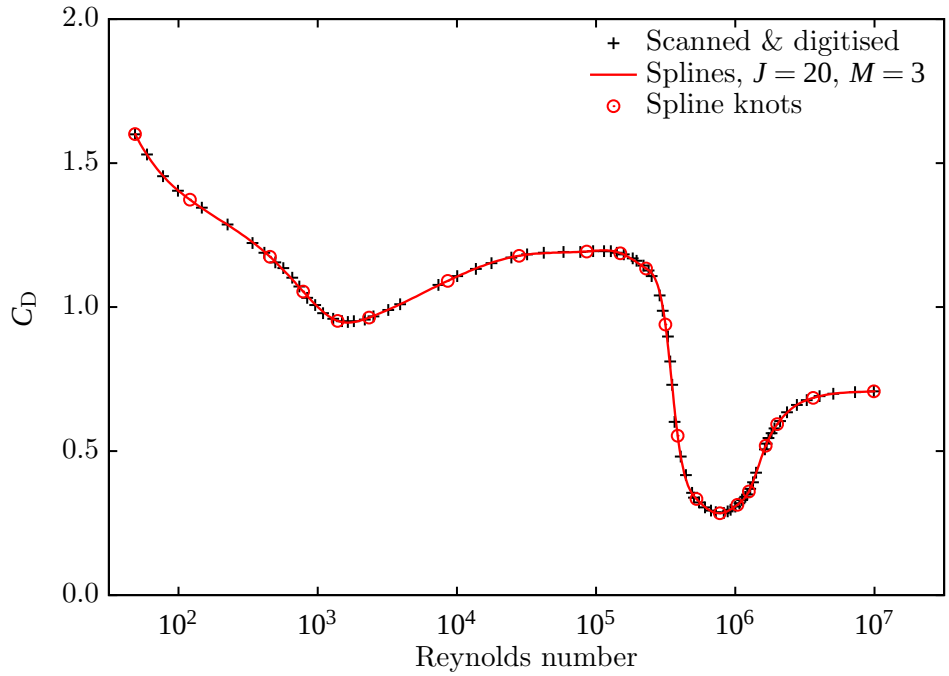


Figure 9: Example of the use of approximating splines to interpolate and smooth data – using scanned and digitised results from Batchelor (2000, p341) for the variation of drag coefficient with Reynolds number of a smooth circular cylinder

The situation here is where one has a set of discrete points that might have been digitised or measured by hand, which might have small deviations from a smooth curve for plotting because of the imperfections of the digitising process or using straight lines between data points.

As an example we consider the data set obtained by the author when he scanned and digitised figure 5.11.6 on page 341 of Batchelor (2000), for the variation of the drag coefficient C_D of a smooth circular cylinder as a function of Reynolds number. This includes the well-known sudden drop in C_D as the boundary layer goes from laminar to turbulent, making interpolation/approximation by global methods very difficult, and which was the reason the author chose this problem as a demanding test.

Here the author tried to approximate the problem as simply as possible, without giving an extra knots file, the default option being simply to allow the program to space knots such that there are the same number of data points in each interval. Results were highly satisfactory, as shown in Figure 9, with the smooth continuous line of the output of the program and the points of the author's digitisation. The results shown are for $J = 20$ intervals with cubic functions, $M = 3$, using values of the logarithm of the Reynolds number for the x_i, *i.e.* using the co-ordinates in which the figure is plotted.

4.4 A quasi-interpolation example

We consider the limit when J is equal to $N - 3$ or just less than that, as has been mentioned above, giving a near-interpolation of the data. Figure 10 shows a sine wave defined by 21 points and the near- or quasi-interpolation using 19 knots or 18 intervals. The missing two points can be identified on closer examination. The results are good. This is a way of using splines for quasi-, or almost-interpolation using the program described in the Appendix. It can be simply implemented by taking the data file, for example *Filename.dat*, copying to file *Filename.knots* and deleting any two or more lines (the second column of numbers in the knots file is not read and could stay there).

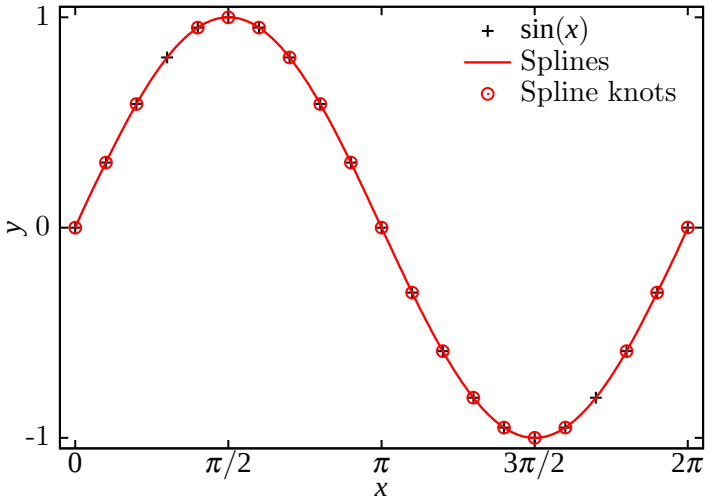


Figure 10: Example of ”quasi-interpolation” where there are two fewer knot points than data points

4.5 Numerical differentiation

It is well-known that numerical differentiation is an operation which may give quite poor results, whether from natural variability in the data, or where the data has been truncated to too few significant figures. Using the present program is one way to obtain smoother and possibly more-reliable results. The derivatives are easily obtained from equation (2), and have been used in equations (3) and (4). The output of the program includes for a large number of points output, the default value being 20 over each spline interval, the function value at the point, first and second derivatives there and third derivatives if cubic splines $M = 3$ are used.

4.6 Numerical integration

This is, of course, relatively simple and robust anyway, even with noisy data, nevertheless we present the expressions from Approximating Splines, trivially obtained. Within a single interval, the integral is

$$\int_{X_j}^{x} P_j(x)\,\mathrm{d}x = c_{j,0}\,(x-X_j) + \tfrac{1}{2}c_{j,1}\,(x-X_j)^2 + \tfrac{1}{3}\,c_{j,2}\,(x-X_j)^3 + \tfrac{1}{4}\,c_{j,3}\,(x-X_j)^4\,,$$

where for quadratic splines the last term is dropped. The integral of the spline over the whole interval of approximation, from the first knot point X_1 to the last X_{J+1}, is

$$\sum_{j=1}^{J}\int_{X_j}^{X_{j+1}} P_j(x)\,\mathrm{d}x = \sum_{j=1}^{J}\left(c_{j,0}\delta_j + \tfrac{1}{2}c_{j,1}\delta_j^2 + \tfrac{1}{3}\,c_{j,2}\delta_j^3 + \tfrac{1}{4}\,c_{j,3}\delta_j^4\right),$$

where δ_j is the length of interval j, as defined previously.

4.7 Calculation of envelopes to data points

Another application of the present approximation method is to the calculation of an *envelope* above or below data points. The approach suggested is first to calculate the approximating spline to all the points, and to delete those points which lie above or those below the approximation approximate the remaining points, and repeat as many times as necessary. The procedure is simple using the accompanying program

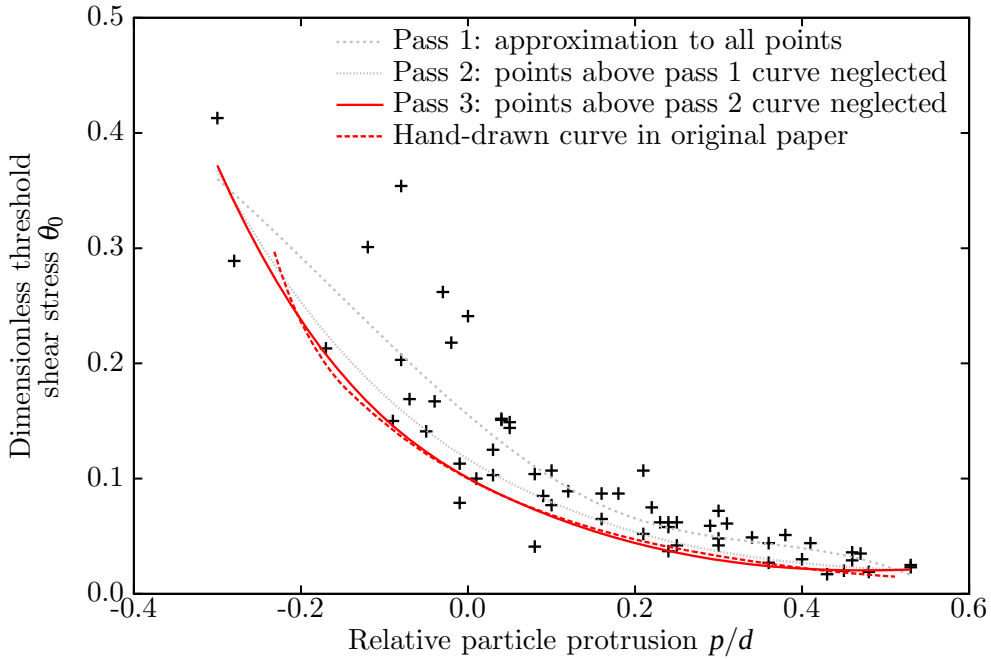


Figure 11: Calculation of envelope by systematic deletion of points above or below approximation

and is described in the Appendix. Figure 11 shows results from table 3 of Fenton & Abbott (1977) and plotted in their figure 5. The ordinate is the dimensionless threshold shear stress $\theta_0 = \tau_0/(\rho g(G-1)d)$, where τ_0 is the bed shear stress at which particle movement occurred, ρ is fluid density, g is gravitational acceleration, G is specific gravity of the bed grain, and d is grain diameter. The protrusion p is the amount by which the particle protruded into the flow above the otherwise co-planar bed grains. The figure shows how the threshold bed shear stress on a bed particle, that which entrains the particle into the flow, depends on how much that particle protrudes into the flow. Because of the experimental arrangement with a finite time of exposing each particle at a particular elevation, it was the *minimum* shear stress which was important, and so it was desired to plot the envelope below the points.

The process of successive approximations, neglecting all points above the curve each time, gave a good result for the envelope to the points. As approximately half the data points are lost with each pass, the number of passes is limited. In the figure shown, and the standard in the accompanying program, the two data points with the smallest and largest values of independent variable p/d, were always retained to ensure that the final envelope would extend from the smallest to largest values of p/d. Cubic splines with $M = 3$ were used – but with only $J = 2$ intervals. Another pass of the program gave poor results, where the envelope passed near to the outlying points below the others, and with oscillations. There is still a certain amount of arbitrary judgement necessary in applying the method.

Also plotted is the original envelope curve sketched by Fenton & Abbott (scanned and smoothed with the present program). It can be seen that the calculation of the envelope by the present program agrees surprisingly closely. At the left of the figure, one might say that the present program has shown greater courage and judgement than the original authors, who truncated their envelope curve as shown.

The result of this figure was of some interest, for it confirmed Bagnold's conjecture that the Shields diagram for incipient motion of bed particles is misleading. For large grain Reynolds number R_* the dimensionless threshold stress for incipient motion was believed to be $\theta_0 \approx 0.03$ to 0.06, almost certainly obtained from experiments on large bed material which was artificially levelled in laboratories such that $p \approx 0$. In a natural bed, however, particles can project above their neighbours with a finite value of p/D and, according to the figure, a correspondingly smaller value of $\theta_0 \approx 0.01$ for large grain Reynolds numbers.

References

Batchelor, G. K. (2000), *An Introduction to Fluid Dynamics*, Cambridge.

Conte, S. D. & de Boor, C. (1980), *Elementary Numerical Analysis*, third edn, McGaw-Hill Kogakusha, Tokyo.

de Boor, C. (2001), *A Practical Guide to Splines*, revised edn, Springer, New York.

Fenton, J. D. (1994), Interpolation and numerical differentiation in civil engineering problems, *Civ. Engng Trans, Inst. Engnrs Austral.* **CE36**, 331–337. http://johndfenton.com/Papers/Fenton94b-Interpolation-and-numerical-differentiation-in-civil-engineering-problems.pdf

Fenton, J. D. & Abbott, J. E. (1977), Initial movement of grains on a stream bed: the effect of relative protrusion, *Proc. Roy. Soc. London Ser. A* **352**, 523–537.

Nikuradse, J. (1950), Laws of flow in rough pipes (English translation from the original German), Technical Memorandum 1292, National Advisory Committee for Aeronautics, Washington D.C. http://digital.library.unt.edu/ark:/67531/metadc63009/m2/1/high_res_d/19930093938.pdf

Appendix

A Program package

A.1 Approximating splines program files

The program files are available from http://johndfenton.com/Approximating-splines/.

The following looks complicated, but use of the package is really quite simple: one needs the program file *Approximating-splines.exe*, a file *Control.dat* in the same directory, whose first line contains information where to find a third file *Path/Filename.dat* containing the data, and which columns of that file to use. A fourth file *Path/Filename.knots* may be later necessary to adjust the spline knot points.

Approximating-splines.exe – the program file.

Control.dat – in the same directory as the program, directs it to the working directory where the data is stored, which is where a results file of the same name, but with **.res* extender will be created. This file has all the computational parameters on the same line. The program only reads the first line, so that the file can contain many lines of information, each for a different problem, which can be brought to the first line when required.

Data directory path/ *Filename* m_x m_y m_w m_{columns} M J *Comment*/*Remark*

The entries are:

- The path where the data is (it can be anywhere on the computer – *e.g. C:/Hydraulics/Pipelines/* – and can be specified relative to where the executable file is *e.g. ../Thesis/Chapter_1/*). It is also where output will be. It must terminate with /.
- The second entry is the name of the example or data set file, *without extender*, which is the common name of one or two data files (*Filename.dat* and possibly *Filename.knots*) which will be used to produce a results file *Filename.res*.
- m_x: column number in the data file *Filename.dat* for the x_i
- m_y: column number for y_i
- m_w: column number for weights w_i. If there is no column for the w_i, this number is 0, and the program sets $w_i = 1$ for all i
- m_{columns}: Total number of data columns in file, however annotations can follow on any data line
- M: the degree of the splines, 2 or 3.
- J: the number of spline intervals. If $J \leqslant 0$, a data file *Filename.knots* must be provided.
- After this a comment is possible

Example

./Examples/	Scattered	1	2	0	2	2	3	Experiment with no. of intervals: 3, 10
./Examples/	Batchelor341	2	3	0	3	3	20	Scanned figure from Batchelor
./Examples/	Cubic	1	2	0	2	3	6	Trivial example of a cubic to test
C:/Pipelines/	Nikuradse	4	5	0	5	2	0	No weights, needs knots file

Computational-parameters.dat – this file may never have to be examined or modified. After a header line it contains five numbers, one on each line possibly followed by a comment:

- The number of equally-spaced results points to be printed out for each spline interval. If your resulting figure is not sufficiently continuous, the default value of 20 could be increased.
- The number of significant figures for output, default 5
- The convergence criterion for the optimisation, default 1.e-16
- Is an envelope curve to be calculated? If 0 no, if an integer < 0 (> 0), the envelope below (above) the data will be calculated after repeated runs, described below.
- The maximum number of data pairs that the program allows, default 2000. This is included such that with data that is not precisely as expected by the program, it does not go into an infinite loop. If you have more data pairs than this, the program will print out an instruction for you to increase the number in this file and will then exit.

In the directory *C:/Data directory path/* specified in *Control.dat* – one or two data files are to be placed, each with the leading name given, for example, **Filename**.

Filename.dat – containing the actual data. There is a header line, then possibly a blank line, then any number of lines with m_{columns} numbers, plus if necessary, any annotation of the data. The x_i are in column m_x, the y_i in m_y, and if m_w is not equal to 0, the weights w_i to be used for that particular point are in column m_w. Any other columns of data not relevant to the approximation can be included in the file. The m_x, m_y, m_w and m_{columns} specify only the data that is to be used. The lines can be placed *in any order*.

Example, for which $m_x = 1$, $m_y = 2$, $m_w = 3$, $m_{\text{columns}} = 3$ would be used in *Control.dat*.

Sample data set

0	1	10	Weight of 10 to force the approximation to honour this point
1	0	1	
0.8571	0.0570	1	
0.7143	0.1392	1	
...	...	...	...

Filename.knots – if the number of spline intervals J has been set to zero or a negative number, this file must be provided. It contains a header line, possibly a blank line and then any number of single values of the spline node points X_j, ideally beginning with the minimum of all the x_i and ending with their maximum value. The X_j must be in ascending order. On each line any annotation of the data is possible. If the first X_1 is less than the minimum of all the x_i and/or the last X_{J+1} is greater than the maximum data value, then the splines would be extrapolating, and this is not recommended.

Sample knots file

0	This is the minimum
0.03	The first interior point
0.25	
...	

A.2 Files output by the program

Two files are usually generated, three if an envelope curve is required.

1. The main result file is output to the data directory *C:/Data directory path/* and is called **Filename.res**. It has three blocks of data:

Data points (sorted): The input data, extracted as described above, possibly from different columns of a data file, is sorted according to the x_i and copied here with x_i and y_i in columns 1 and 2. The sorting is necessary if the program automatically allocates knot points. We thought it a possibly-useful extra in any case. The precision of the original numbers is retained.

Spline data – knot points and coefficients for each interval: This block could be copied to another file where it could be used in simple stand-alone machine readable form by other software or inserted into a spreadsheet. The first line contains, after a Gnuplot comment symbol, the values of J and M. The next $J+1$ lines contain, for $j = 1 \ldots J+1$, values of X_j, and $c_{j,m}$ for $m = 0, \ldots, M$, the whole spline solution. Each number is in scientific format with the number of significant figures given in *Computational-parameters.dat*.

Computed splines - values and derivatives: This is useful for plotting. At the knot points and also at a number of points intermediate between them, that number specified in *Computational-parameters.dat*, with 20 as default, values of x are printed followed by the value of the spline there $P_j(x)$, and its M derivatives $P_j'(x)$, $P_j''(x)$, and if $M = 3$, $P_j'''(x)$. The numbers are output in floating-point format with the number of significant figures given in *Computational-parameters.dat*.

The file as created with these three blocks separated by at least two blank lines is suitable for reading by the graph plotting software Gnuplot (http://www.gnuplot.info/).

2. A small file **Control.plt** is created by the program in the directory where *Approximating-splines.exe* and *Control.dat* are placed. There is already a file *Plot.plt* there which can be used for immediate plotting with Gnuplot to see the results of running the program. It reads *Control.plt* and plots the results in a window.

3. If in *Computational-parameters.dat* the fourth data line contains an integer $\neq 0$, this enables the calculation of an envelope using repeated passes of the program. An extra new data file is created with a "-1" appended to the filename, *Filename-1.dat*. If the integer in the fourth data line is positive/negative, only those data points falling above/below the splines curve are output. Then in the first line of *Control.dat*, *Filename* should be amended by the user to *Filename-1*, the program run, fitting a curve to half of the original points and generating a file *Filename-1-1.dat* containing a quarter of the original points. Modifying the filename in the first line in *Control.dat* to *Filename-1-1*, running the program again, gives a curve which has 1/8 of the original points outside it, and can probably be considered to be the envelope, in the spirit of approximation. The extra file *Filename-1-1-1.dat* will now have only those points and is probably not useful. The process can probably be terminated without running again. At each stage the program retains the first and last points. They could be edited out if required.

A.3 Additional files included in the package

- *Plot.plt* is a Gnuplot file which can be used for immediate plotting of results from the program – see below.
- Directory *Examples* with data files *Scattered.dat*, *Sine.dat*, *Sine.knots*, *Batchelor341.dat*, *Cubic.dat* plus result files from the authors computations.