Approximation of experimental data to functions is common in physics. The approximations come in two general forms. In *interpolation* an approximating polynomial is found that exactly matches the function at a set of specified points (the data, for example). *Curve fitting* finds a set of parameters that *best* fits the data to a pre-specified function (usually derived from theoretical considerations). It will not, in general, go through all the data points. We begin with interpolation.

1 Cubic Splines

We will study only one method of interpolation, *cubic splines*. This method is very general and the most used method for interpolating data. In the process, we will need to introduce methods for solving *systems* of linear equations.

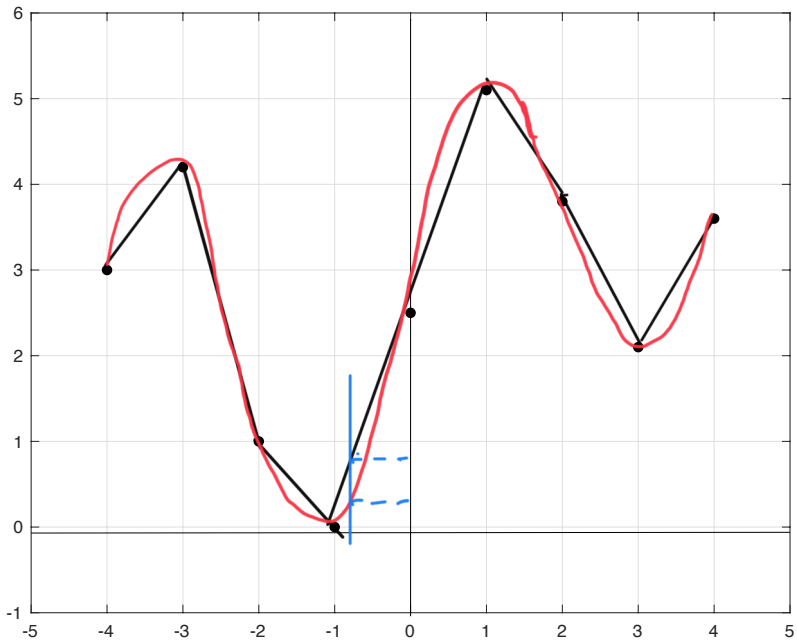

Figure 1: Data points.

(1) Consider figure 1, which shows a set of discrete points. For example, these

could be experimental data points, say the intensity of light, measured at discrete times. We would like to be able to approximate the function value at all other times. On the figure, connect the points with straight lines. Using these lines approximate the function at non-integer points, say -1.5, .5, etc. Discuss at your table what might be some problems with this approximation.

- Derivatives typically exists
- Can't pass a curve through every point
- No curves

(2) Notice that linear approximation between points is very spiky. Now try 'to approximate the function by using quadratics. The general form of a quadratic is $f(x) = ax^2 + bx + c$ so you need three points. Taking the function three points at a time, sketch a quadratic at these points. Is the approximation 'better' than the linear approximation? What issues still arise.

- Still have areas where data may be off

(3) Now taking four points at a time, fit a cubic to the data. You need four points because a cubic has the form, $f(x) = dx^3 + cx^2 + bx + d$. Is the approximation 'better'? What issues still arise.

- Still have discontinuities - Not smooth at all points

There is a better way, using *cubic splines*. The idea is to fit a cubic between *any* pair of points such that the function, first, and second derivatives match at the end points. This seems to be challenging, but in fact, turns out relatively easy to compute numerically. Lets begin.

(4) The general form of a cubic (one that does not necessarily pass through the origin) is

$$p(x) = a_j(x - x_j)^3 + b_j(x - x_j)^2 + c_j(x - x_j) + d_j. \tag{1}$$

The j's here represent the j data points to which we are trying to fit a cubic. In order for us to find the interpolating cubic, we must solve

for the constants a_j, b_j, c_j and d_j. To begin to do this, first note that requiring that the approximation be exact at the end point $x = x_j$ will yield d_j as we showed in the lecture. Now let's make things a little cleaner. Letting $h_j = x_{j+1} - x_j$ and $p_j = p(x_j)$, find p at $x = x_{j+1}$.

$p'(x) = 3a_j(x - x_j)^2 + 2b_j(x - x_j) + c_j$

$p''(x) = 6a_j(x - x_j) + 2b_j$

$p''_j = 2b_j \Rightarrow b_j = p''_j/2$

$p_{j+1} = a_j h_j^3 + b_j h_j^2 + c_j h_j + p_j$

(5) Use the same procedure we just used in the lecture when $x = x_{j+1}$ to solve for the constant a_j. Again you will solve for one of the constants of the cubic in terms of the second derivative.

$p''_{j+1} = 6a_j h_j + 2b_j \qquad a_j = \frac{1}{6}\frac{p''_{j+1} - p''_j}{h_j}$

The result from the lecture is unwieldily but contains only known values and unknown second derivatives. Specifically we have that a function can be approximated at any point by

$$p(x) = p_j + \left[\frac{p_{j+1} - p_j}{h_j} - \frac{h_j p''_{j+1}}{6} - \frac{h_j p''_j}{3}\right](x - x_j) + \frac{p''_j}{2}(x - x_j)^2 + \frac{p''_{j+1} - p''_j}{6h_j}(x - x_j)^3, \quad x_j \le x \le x_{j+1} \quad (2)$$

with first derivative

$$p'(x) = \frac{p_{j+1} - p_j}{h_j} - \frac{h_j p''_{j+1}}{6} - \frac{h_j p''_j}{3} + p''_j(x - x_j) + \frac{p''_{j+1} - p''_j}{2h_j}(x - x_j)^2, \quad x_j \le x \le x_{j+1}. \quad (3)$$

There are a total of N unknowns and N equations. All these equations can

$c_j = \frac{p_{j+1} - p_j}{h_j} - \frac{h_j p''_{j+1} + 2h_j p''_j}{6}$

h is known

p'' is the only thing unknown

be written in matrix form as

$$\begin{bmatrix} 2h_1 & h_1 & & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & & \\ & & & \ddots & & & \\ & & & & h_{N-2} & 2(h_{N-2}+h_{N-1}) & h_{N-1} \\ & & & & & h_{N-1} & 2h_{N-1} \end{bmatrix} \begin{bmatrix} p''_1 \\ p''_2 \\ p''_3 \\ \vdots \\ p''_{N-1} \\ p''_N \end{bmatrix}$$

$$= \begin{bmatrix} 6\frac{p_2-p_1}{h_1} - 6p'_1 \\ 6\frac{p_3-p_2}{h_2} - 6\frac{p_2-p_1}{h_1} \\ 6\frac{p_4-p_3}{h_3} - 6\frac{p_3-p_2}{h_2} \\ \vdots \\ 6\frac{p_n-p_{N-1}}{h_{N-1}} - 6\frac{p_{N-1}-p_{N-2}}{h_{N-2}} \\ -6\frac{p_N-p_{N-1}}{h_{N-1}} + 6p'_N \end{bmatrix} \quad (4)$$

Eq.(4) gives us the unknown second derivatives that we need for Eq.(2).

(6) There are two pieces of information still needed in Eq.(4), what is that information.

First Derivative at the endpoints

Don't know the first Derivatives (p')

(7) In the *natural spline*, the second derivatives at the end points $x = x_1$ and $x = x_N$ are set to zero. Do so and write down the resulting matrices.

first and last entries are zero on RHS

(8) You have now worked out all the steps to interpolate data with a cubic spline. Just to make sure we've got it, list all the steps necessary to

interpolate using a cubic spline. Refer to specific equations (by number) as needed

Tridiagonal Linear Systems. Notice that our method of interpolation using cubic splines has been reduced to an *algebra* problem (Eq. 4). The particular form of Eq.(4) is called *tridiagonal* due to the fact that all but the first and last rows have three non-zero entries that lie along the main and adjacent diagonals of the matrix.

Systems of linear equations such as Eq.(4) occur quite often in physics and can be solved in many ways. Perhaps the most common method is with *Gaussian elimination*. It turns out that Gaussian elimination is especially useful and easy for tridiagonal systems. Because most students have seen some form of Gaussian elimination, here we present the main formulae to solve for the unknowns.

First we write the general tridiagonal system as

$$\begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & & \ddots & & \\ & & & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & & a_N & b_N \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-1} \\ x_N \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_{N-1} \\ r_N \end{pmatrix} \tag{5}$$

Gaussian elimination essentially involves solving for one variable at a time by eliminating that variable from a row. For example, if one multiplies the first row by a_2/b_1, subtracts it from the second, a new equation results. This new equation is then inserted in place of the original second row. The first two equations now are

$$\begin{aligned} b_1 x_1 \quad & + c_1 x_2 && = r_1 && \text{original first row} \\ & \left(b_2 - \tfrac{a_2}{b_1} c_1\right) x_2 + c_2 x_3 && = r_2 - \tfrac{a_2}{b_1} r_1 && \text{modified second row} \end{aligned}$$

The process continues $N-1$ times after which the system has the form

$$\begin{array}{rrrrrrrrr} \beta_1 x_1 & +c_1 x_2 & & & & & = & \rho_1 \\ & \beta_2 x_2 & +c_2 x_3 & & & & = & \rho_2 \\ & & \beta_3 x_3 & +c_3 x_4 & & & = & \rho_3 \\ & & & & \ddots & & \vdots & \\ & & & & \beta_{N-1} x_{N-1} & +c_{N-1} x_N = & \rho_{N-1} \\ & & & & & \beta_N x_N = & \rho_N \end{array} \tag{6}$$

where

$$\beta_1 = b_1, \quad \beta_j = b_j - \frac{a_j}{\beta_{j-1}} c_{j-1} \quad j = 2, \ldots, N \tag{7}$$

and

$$\rho_1 = r_1, \quad \rho_j = r_j - \frac{a_j}{\beta_{j-1}} \rho_{j-1} \quad j = 2, \ldots, N. \tag{8}$$

From Eqs (6) - (8) the process for solving tridiagonal systems of linear equations is now clear. Using Eqs. (7) and (8) one solves for the β's and $\rho's$. Then using the last line in Eq. (6), x_N is solved, and back substituted to find x_{N-1} and so forth. We can write the general form for the solution of x_j as,

$$x_{N-j} = \frac{(\rho_{N-j} - c_{N-j} x_{N-j+1})}{\beta_{N-j}}, \quad j = 1, \ldots, N. \tag{9}$$

That's it! You now have everything you need to interpolate using cubic splines. You use Eqs (6) -(9) to find the second derivatives, which are then substituted into Eq. (2) to find the values of the spline.

(9) In left column of the table below, write a step by step procedure for interpolating using a cubic spline. In the right column, write snippets of `MatLab` code that you might use to carry out each step.

Step	Code

Homework 3, due Monday, Sept. 24

(1) Do problem **3.5** from the text

(2) Do problem **3.6** from the text. Note, the first derivative of J_1 is given by

$$\frac{dJ_1(x)}{dx} = \frac{1}{2}\left(J_o(x) - J_2(x)\right).$$

(3) (*Graduate students*) Do problem **3.7** from the text.

(4) Do problem **3.8** from the text.