1 Interpolation

In practice one often meets a situation where a function of interest is only given as a set of tabulated discrete points $\{x_i, y_i\}$, $i = 1 \ldots n$, for example as a result of a numerical integration of a differential equation. *Interpolation* means constructing a (smooth) function, called interpolating function, which passes exactly through the given points and hopefully approximates the unknown function of interest in between the tabulated points. One can use the interpolating function for different practical needs, like estimating the unknown function between the tabulated points, differentiating, integrating etc. Interpolation is a specific case of curve fitting, in which the function must go exactly through the data points.

1.1 Polynomial interpolation

Polynomial interpolation uses a polynomial as the interpolating function. Given a table of n points, $\{x_i, y_i\}$, one can construct a polynomial of the order $n-1$ which passes exactly through the points. This polynomial can be intuitively written in the *Lagrange form*,

$$P^{(n-1)}(x) = \sum_{i=1}^{n} y_i \prod_{k \neq i}^{n} \frac{x - x_k}{x_i - x_k} \; . \qquad (1)$$

Higher order interpolating polynomials, say larger than 5, are susceptible to the *Runge phenomenon*: erratic oscillations close to the end-points of the interval, as illustrated on Figure **??**. Therefore when interpolating from a large table one usually uses only the nearest few points instead of all the points in the table.

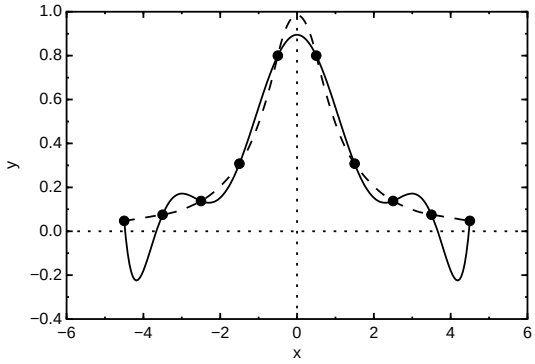

Figure 1: Lagrange interpolating polynomial (solid line) and quadratic spline (dashed line). Polynomial interpolation shows the Runge phenomenon: large oscillations at the end-points.

1.2 Spline interpolation

Spline interpolation uses a *piecewise polynomial*, called *spline*, as the interpolating function[1]: at each interval $x_i \leq x \leq x_{i+1}$ the spline $S(x)$ is represented by a polynomial $S_i^{(k)}(x)$ of a given order k,

$$S(x) = S_i^{(k)}(x), \text{ if } x_i \leq x \leq x_{i+1} \; . \qquad (2)$$

The spline of the order k can be made continuous at the tabulated points together with its $k-1$ derivatives.

If the spline polynomials are linear the spline is called *linear interpolation*,

$$S_i^{(1)}(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i) \; . \qquad (3)$$

The most practical is the cubic spline, where the polynomials $S_i^{(3)}(x)$ are of third order: the spline is a continuous function together with its first and second derivatives. The cubic spline has also a nice feature that it (sort of) minimizes the total curvature of the interpolating function. This makes the cubic splines look good.

Quadratic splines are not nearly as good as cubic splines in most respects. Particularly they might oscillate unpleasantly when a quick change in the tabulated function is followed by a period where the function is nearly a constant. The cubic spline is less susceptible to such oscillations. However quadratic spline is simpler to program.

1.2.1 Quadratic spline

At the interval $x_i \leq x \leq x_{i+1}$, $i = 1 \ldots n-1$, the spline is given by a second order polynomial,

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 \; . \qquad (4)$$

The coefficients b_i and c_i can be found from the continuity conditions for the spline itself and its first derivative,

$$\begin{aligned} S_i(x_{i+1}) &= S_{i+1}(x_{i+1}) \; , & (5)\\ S_i'(x_{i+1}) &= S_{i+1}'(x_{i+1}) \; , & (6)\\ S_{n-1}(x_n) &= y_n . & (7) \end{aligned}$$

where $i = 1 \ldots n-2$. This gives $2(n-2)+1$ equations

$$\begin{aligned} y_i + b_i h_i + c_i h_i^2 &= y_{i+1} \; , & (8)\\ b_i + 2c_i h_i &= b_{i+1} \; , & (9)\\ y_{n-1} + b_{n-1} h_{n-1} + c_{n-1} h_{n-1}^2 &= y_n \; . & (10) \end{aligned}$$

where $i = 1 \ldots n-2$, $h_i = x_{i+1} - x_i$.

[1] other classes of functions can be used as well.

Thus we have $2(n-1)$ unknown coefficients and $2(n-2)+1$ equations which leaves us with one coefficient which we can define arbitrarily. For example,

$$b_1 = \frac{y_2 - y_1}{x_2 - x_1} \ , \tag{11}$$

which via eq. (**??**) gives $c_1 = 0$. Now we can calculate all other coefficients recursively,

$$\begin{aligned} b_i &= b_{i-1} + 2c_{i-1}h_{i-1} \ , & (12)\\ c_i &= \frac{y_{i+1} - y_i - b_i h_i}{h_i^2} \ , & (13)\\ i &= 2 \ldots n-1 \ . & (14) \end{aligned}$$

Alternatively, one can choose

$$b_{n-1} = \frac{y_n - y_{n-1}}{x_n - x_{n-1}} \ , \tag{15}$$

which via eq. (**??**) gives $c_{n-1} = 0$. A recursion down gives

$$\begin{aligned} b_i &= 2\frac{y_{i+1} - y_i}{h_i} - b_{i+1} \ , & (16)\\ c_i &= \frac{b_{i+1} - b_i}{2h_i} \ , & (17)\\ i &= n-2 \ldots 1 \ . & (18) \end{aligned}$$

In practice it is better to run this iteration procedure once up and once down and than average the resulting b's and c's.

1.3 Integration and differentiation of tabulated functions

Recipe: first interpolate the tabulated data and then integrate or differentiate the interpolating function.