Linear Interpolation

In many (most?) numerical calculations, the desired results are obtained not as continuous functions, but instead at only a finite set of data points.

For example, suppose we wish to compute numerically the trajectory of a particle moving under the influence of a uniform gravitational field. The *analytic* solution is well known: the x and y coordinates of the particle at time t are given by

$$\begin{aligned} x &= x_0 + v_{x0}\, t \\ y &= y_0 + v_{y0}\, t - \tfrac{1}{2}\, g\, t^2 \, , \end{aligned}$$

where x_0 and y_0 are the initial coordinates (at time $t = 0$), v_{x0} and v_{y0} are the corresponding initial velocities, and g is the (constant) acceleration due to gravity. However, the *numerical* solution is typically computed at some set of discrete times t_i, so the numerical data set consists of triples of the form $(t_i,\, x_i,\, y_i)$. In principle, without some additional physical insight into the problem, we know nothing of the solution at times other than the t_i.

Often we need to say something about the behavior of the solution at other times. For example, in computing the range of the particle, we want to know the value of $x - x_0$ when y is equal to y_0 (for time $t > 0$), and this instant in general does not correspond to one of the t_i. How are we to proceed?

Mathematically, it is always possible to fit a smooth curve through any given set of data points (e.g. N points define a unique polynomial of degree $N - 1$). This curve can then be used to *define* the intermediate values of the numerical solution. Bear in mind, though, that the intermediate values are only mathematical fits—"best guesses" of the true solution based on the assumption that it is smooth.

The simplest fitting approach, and the one which we will employ throughout this course, simply connects successive points by straight lines. This approximation is known as *linear interpolation*. Let's suppose that we want to use linear interpolation to connect the values of two quantities—p and q, say—along the trajectory (p and q can be any variables—t, x, y, etc.). We want to write p as a function of q. The equation of the straight line through the points $(q_{i-1},\, p_{i-1})$ and $(q_i,\, p_i)$ is easily shown to be

$$p(q) = p_{i-1} + (q - q_{i-1}) \left(\frac{p_i - p_{i-1}}{q_i - q_{i-1}} \right) .$$

It is clearly linear, and you can verify for yourself that $p(q_{i-1}) = p_{i-1}$ and $p(q_i) = p_i$. The *piecewise linear* function $p(q)$ defined by this equation is our approximation to the true solution. We could do better by fitting higher-order curves to the numerical data, but the linear approximation is convenient and accurate enough for our purposes.

The uses of linear approximation are most easily illustrated by a few examples. Suppose that the result of a simulation of two-dimensional projectile motion is a set of triples $(t_i,\, x_i,\, y_i)$. Interpolating one variable as a function of another is just a matter of choosing p and q in the above expression.

1. To determine the position of the particle at some intermediate time t, we must interpolate both x and y to that time, setting $q = t$ and $p = x$, then $p = y$, in the equation for $p(q)$) given above.

2. To compute the time of flight and range of the particle, we must determine the values of t and $x - x_0$ corresponding to $y = y_0$ for the second time. In this case, we must interpolate t and x to their values at $y = y_0$. If i is such that $y_{i-1} < y_0 <= y_i$, we can write ($q = y$, $p = t$):

   $$t(y) = t_{i-1} + (y - y_{i-1}) \left(\frac{t_i - t_{i-1}}{y_i - y_{i-1}} \right) ,$$

and similarly for $x(y)$ ($q = y$, $p = x$). Substituting $y = y_0$ leads to the interpolated values of t and x. Note that this procedure gives the same result as interpolating

$$y(t) = y_{i-1} + (t - t_{i-1}) \left(\frac{y_i - y_{i-1}}{t_i - t_{i-1}} \right)$$

and then solving for $y(t) = y0$.

3. To determine the height $(y - y_0)$ of the particle at some given horizontal position $(x - x_0)$, we use linear interpolation in y to the specified x ($q = x$, $p = y$):

$$y(x) = y_{i-1} + (x - x_{i-1}) \left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}} \right)$$

Note that we may regard *any* of the variables t, x, y,... as the independent variable when interpolating the values of the others.

A C/C++ program fragment to perform the first interpolation above might look as follows:

```
while (t <= t_int) {

    t_prev = t;
    x_prev = x;
    y_prev = y;

    t = t + dt;
    x = x0 + vx0*t;
    y = y0 + vy0*t - 0.5*g*t*t;
}

x_int = x_prev + (t_int - t_prev) * (x - x_prev) / (t - t_prev);
```

Note how "previous" values are stored at each iteration; `prev` values correspond to index $i - 1$. Examples of the other interpolations are found in Homework #2.