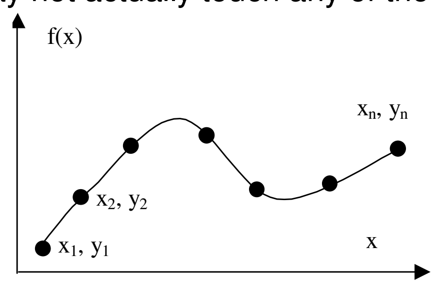
Interpolation with Lagrange Polynomials

Interpolation means estimating a value between actual data points (extrapolation means estimating a value outside the range of actual data points). In this section we discuss the Lagrange method of interpolation that uses a series of terms that are powers of x (polynomial functions) to interpolate and is simple to code. Another, widely used, method is Cubic Splines that limits the highest power to x^3 and produces smoother functions over a large range of data, but is more complicated to code.
In interpolation we accept each data point as having no error and want the interpolation function to go precisely through each data point (unlike least squares where the fit may not actually touch any of the points).



A feature of the Lagrange method described here is that the single constructed function extends across all the data points.

The method generates a series of polynomials of powers of x with the highest power of x one less than the number of points. So if there are 4 points, the polynomials will be of the form $ax^3 + bx^2 + cx + d$ where a, b, c and d are constants. It is based on the facts that a straight line (y = ax+b) is the simplest function that goes through two points, a parabola $y = ax^2 + bx + c$) is the simplest function that goes through three points, a cubic ($y = ax^3 + bx^2 + cx + d$) is the simplest function that goes through four point and so on.
It is given explicitly by Lagrange's classic formula,

$$f(x) = \frac{(x-x_2)(x-x_3)\ldots(x-x_N)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_N)}y_1 + \frac{(x-x_1)(x-x_3)\ldots(x-x_N)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_N)}y_2 +$$

$$\ldots\ldots\ldots\ldots + \frac{(x-x_1)(x-x_2)\ldots(x-x_{N-1})}{(x_N-x_1)(x_N-x_2)\ldots(x_N-x_{N-1})}y_N \qquad 1.$$

There are N terms each a polynomial of degree N - 1 and each is constructed to be zero at all of the x_i except one, at which it is constructed to be y_i. The j^{th} term in the formula is:

$$P_j(x) = y_j \prod_{k=1 k \neq j}^{n} \frac{x - x_k}{x_j - x_k}$$

which is a helpful way of thinking of it when writing code ;-).

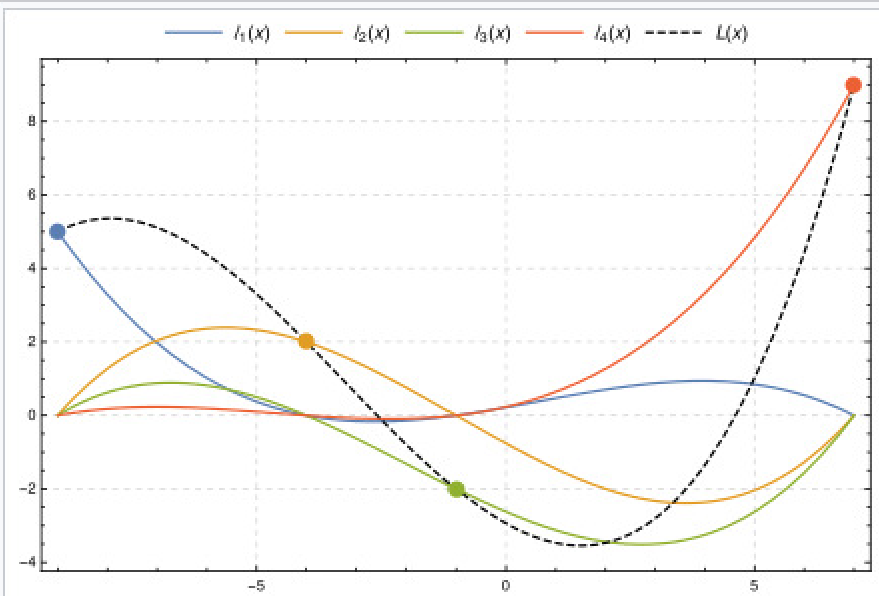
So, for two points it's a straight line (highest power x), for three points it's a parabola (highest power x^2), for three points it's a cubic (highest power x^3), and so on. Note that there is no requirement that the points be evenly spaced.

Since the data points control the shape of the curve, they are otherwise known as ***control points***.
We can write the function in the form:

$$f(x) = y_1 l_1(x) + y_2 l_2(x) + \ldots + y_N l_N(x)$$

where the l(x) functions are called the **Lagrange basis functions** or **fundamental polynomials**.

Here's an example showing the basis functions four control points. Note that they do not need to be equally spaced in x.



This image shows, for four points ((−9, 5), (−4, 2), (−1, −2), (7, 9)), the (cubic) interpolation polynomial $L(x)$ (dashed, black), which is the sum of the *scaled* basis polynomials $y_0\ell_0(x)$, $y_1\ell_1(x)$, $y_2\ell_2(x)$ and $y_3\ell_3(x)$. The interpolation polynomial passes through all four control points, and each *scaled* basis polynomial passes through its respective control point and is 0 where x corresponds to the other three control points.

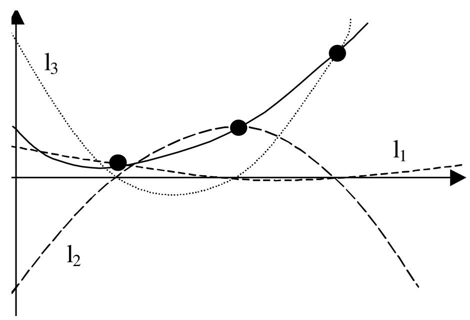
Note how the number of oscillations increases with the number of data points – in general a straight line has one root (crosses the x-axis once), a quadratic two roots (crosses the x-axis twice), a cubic three roots (crosses the x-axis three times) and so on. For a large number of points, the terms become high order polynomials and may have undesirable high frequency oscillations that result in the loss of precision as shown below.

As an example, if there are three points we get a series of quadratics:

$$f(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} y_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} y_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} y_3$$

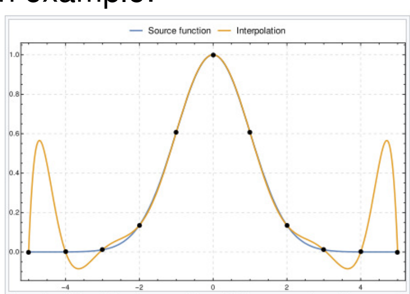
At $x = x_1$ the first term evaluates to y_1 and the others evaluate to 0, leaving, correctly: $f(x_1) = y_1$. Similarly, at $x = x_2$ and $x = x_3$, the function evaluates to y_2 and y_3 respectively.

In this case since the highest order power in each basis function is one less than the number of data points, each basis function will be a parabola (highest order x^2). The following graph shows an example of three data points interpolated in this way:



Note that the basis functions (parabolas) must be zero at data points they do not pass through, and each parabola passes through only one data point.

Lagrange interpolation is a computationally simple method that works well for data that changes slowly and does not have large fluctuations. Large fluctuations in one part of the data will introduce the high order oscillations in the interpolation function in other parts of the data that shouldn't have them – this is a problem with this method. Here's an example:



Example of interpolation divergence for a set of Lagrange polynomials.

There are other methods of interpolation – e.g. Cubic Splines – that do not have these problems.

Example 1

Interpolate the following data at x = 2.0 using Lagrange polynomials:

x	y
1	0
4	1.386294
6	1.791760

The Lagrange polynomial is:

$$f(x) = \frac{(x-4)(x-6)}{(1-4)(1-6)}0 + \frac{(x-1)(x-6)}{(4-1)(4-6)}1.386294 + \frac{(x-1)(x-4)}{(6-1)(6-4)}1.791760$$

giving the basis functions:

$l_1(x) = (x-4)(x-6)/(1-4)(1-6) = 1/15(x^2 - 10x + 24)$
$l_2(x) = (x-1)(x-6)/(4-1)(4-6) = -1/6(x^2 - 7x + 6)$
$l_3(x) = (x-1)(x-4)/(6-1)(6-4) = 1/10(x^2 - 5x + 4)$

Putting them all together the Lagrange polynomial is:

$f(x) = (-1.386294/6)\,(x^2 - 7x + 6) + (1.791760/10)\,(x^2 - 5x + 4)$

$= -0.051873x^2 + 0.721463x - 0.66959$

Then to interpolate at x = 2

f(2) = -0.207492 + 1.442926 – 0.66959 = 0.565844

Alternatively we can calculate the interpolation directly from the basis functions by substitution before multiplying out the factors (*– this is the best way to do it in code since the basis function can be implemented as a function call)*:

$$f(2.0) = \frac{(2-4)(2-6)}{(1-4)(1-6)}0 + \frac{(2-1)(2-6)}{(4-1)(4-6)}1.386294 + \frac{(2-1)(2-4)}{(6-1)(6-4)}1.791760 = 0.5658444$$

The function is actually ln(x) and therefore ln(2) = 0.69315, so the fractional (relative) error in the interpolation is (0.69315 – 0.5658444)/ 0.69315 = 0.1837 giving a percentage fractional error = 18.4%

Example 2
Interpolate the following data at x = 2.0 using Lagrange polynomials:

x	y
1.0	1.0
4.0	6.0
6.0	4.0

The Lagrange polynomial is:

y = $f(x) = \frac{(x-4)(x-6)}{(1-4)(1-6)}1.0 + \frac{(x-1)(x-6)}{(4-1)(4-6)}6.0 + \frac{(x-1)(x-4)}{(6-1)(6-4)}4.0$

at x = 2

y = $f(x) = \frac{(2-4)(2-6)}{(1-4)(1-6)}1.0 + \frac{(2-1)(2-6)}{(4-1)(4-6)}6.0 + \frac{(2-1)(2-4)}{(6-1)(6-4)}4.0$

= 8/15 + 4 – 0.8 = 3.733