Lagrange Interpolation at n nodes.

Given n distinct points x_i in R, i = 1,…,n and n values $f_i \in R$, we say that the real polynomial p interpolates the data $f_i \,_1^n$ at the nodes x_1 if

$$p(x_i) = f_i, i = 1,\ldots n \qquad (1)$$

It may happen that the data $f_i \,_1^n$ are the values of some function f at the points x_i, that is $f_i = f(x_i), i = 1,\ldots n$. In this case we say that p interpolates f at the nodes x_1. There are an infinite number of polynomials satisfying (1), however there is precisely one polynomial of degree n-1.

When the x-values are not evenly spaced, our previous method of writing an interpolating polynomial fail. The Lagrange polynomial does not require that the x-values are equispaced, as is often not the case. Suppose we have a table of data, of x and f(x) values:

x	F(x)
x_1	f_1
x_2	f_2
x_3	f_3
x_4	f_4

We do not assume uniform Δx, nor do we need the x-values arranged in a particular order, however the x-values must all be distinct. Through these four data pairs we pass a cubic. The Lagrangian form of this is

$$P_3(x) = \frac{(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)} f_1 + \frac{(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)} f_2$$
$$+ \frac{(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)} f_3 + \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)} f_4 \qquad (2)$$

There are 4 terms, each of which is a cubic in x; hence the sum is a cubic. The pattern of each term is to form the numerator as a product of linear factors of the form $(x-x_i)$, omitting one x_i in each term, the omitted value being used to form the denominator by replacing x in each of the numerator factors.

We define the polynomials l_i by

$$l_i(x) = \prod_{j=1, j\neq i}^{n} \frac{(x-x_j)}{(x_i-x_j)} \quad , \text{i= 1,…,n.} \qquad (3)$$

It is not hard to see that $l_i \in P_{n-1}, i = 1,\ldots,n$ and

$$l_i(x_j) = \delta_{i,j}, i = 1,\ldots n;\ j = 1,\ldots n. \qquad (4)$$

We introduce the polynomial p(x) by

$p(x) = \sum_{i=1}^{n} f_i l_i(x)$. Then p(x) is of degree n-1, and by (4)

$p(x_j) = \sum_{i=1}^{n} f_i l_i(x_j) = \sum_{i=1}^{n} f_i \delta_{i,j} = f_j$. So p also satisfies (1). The polynomial constructed in the above way is called the Lagrange interpolating polynomial at the nodes $x_1, \ldots, x_n$. The polynomials $l_1, \ldots, l_n$ form a basis for P_{n-1}, (i.e. the set of all polys. of degree <= n-1), and are called the basic Lagrange polynomials for the nodes $x_1, \ldots, x_n$. They only depend on the nodes.

EXAMPLE. Interpolate for f(2.3) from the following values of x and f(x).
x = (1.1, 1.7, 3.0) and f(x)= (10.6, 15.2, 20.3).

With only 3 pairs of data , a quadratic is the highest degree poly possible. It is

$$P_2(x) = \frac{(x-1.7)(x-3.0)}{(1.1-1.7)(1.1-3.0)}(10.6) + \frac{(x-1.1)(x-3.0)}{(1.7-1.1)(1.7-3.0)}(15.2) + \frac{(x-1.1)(x-1.7)}{(3.0-1.1)(3.0-1.7)}(20.3)$$

At x = 2.3, we get $P_2(x)$ =18.38.
Exercise. Write the above polynomial in the form $P_2(x) = ax^2 + bx + c$.

Error of Interpolation.

Predicting the values of the function at nontabulated points is subject to error. We now find an expression for the error of $P_n(x)$, an n-th degree interpolating polynomial. We write the error function in a form which has the known property that it is zero at the n+1 points , from x_0 through x_n. We call this function E(x). We assume that f(x) is **differentiable n+1 times.**

$E(x) = f(x) - P_n(x) = (x-x_0)(x-x_1)\ldots(x-x_n)g(x)$.
So g(x) accounts for the error at points other than the nodes x_i. We also have

$f(x) - P_n(x) - (x-x_0)(x-x_1)\ldots(x-x_n)g(x) = 0$.
We introduce an auxiliary function W(t) and define it as:

$W(t) = f(t) - P_n(t) - (t-x_0)(t-x_1)\ldots(t-x_n)g(x)$.
Now, W(t) is zero at each of the nodes $x_i, \ldots, x_n$, and also at t=x. Thus W(t) has n+2 distinct zeros. Then by Rolles theorem we see that W'(t) has at least one zero in each of the n subintervals (x_0,x_1), …,(x_m,x), (x,x_{m+1}),…(x_{n-1},x_n). Thus W'' has n zeros, and continuing we see that $W^{n+1}(t)$ has 1 zero in the interval (x_0,x_n). We call this value of t= ξ . We then have

$W^{n+1}(\xi) = 0 = f^{n+1}(\xi) - 0 - (n+1)!\,g(x)$.

Hence $g(x) = \dfrac{f^{n+1}(\xi)}{(n+1)!}$ $\xi \in (x_0, x_n)$.

We now have our error term

$$E(x) = (x - x_0)(x - x_1)...(x - x_n) f^{n+1}(\xi) / (n+1)!. \quad (5)$$

This expression gives us a means of calculating bounds on the error in interpolation, since we don't know the value of ξ.

$$E(x) \le \max((x - x_0)(x - x_1)...(x - x_n) f^{n+1}(\xi)) / (n+1)!$$
$$E(x) \ge \min((x - x_0)(x - x_1)...(x - x_n) f^{n+1}(\xi)) / (n+1)!$$

If we know the function f(x) then we can find a bound on the error.

Interpolation using MATLAB – Example with 4 data points (x,y)

```
>> x = [ 1 2 3 4];
>> y = [ 3 1 3 0];
>> p3 = polyfit(x,y,3)

p3 =

   -1.5000   11.0000  -24.5000   18.0000

>> xi = 1:0.1:4;
>> yi = polyval(p3,xi);
>> plot(xi,yi,'r',x,y,'o')
```