There are a few questions that can be asked about $p_n$ at this point:

**Q1** If the points $(x_j, y_j)$ come from a smooth function, what is the error between $p_n$ and the function $f$?

![Graph](image)

**Q2** What is the cost of evaluating $p_n$? If a new data point is added, $(x_m, y_m)$, what is the cost of updating $p_n$?

**Q3** In floating point arithmetic, is the evaluation of $p_n$ stable?

![Graph](image)

Note: if $y_j = f(x_j)$, then $p_n(x_j) = y_j = f(x_j)$ by construction. If $x \neq x_j$, then

**Theorem:** Let $f \in C^{\infty} [a,b]$. For $x \in (a,b)$, there exists a $s = s(x) \in (a,b)$ such that

$$f(x) - p_n(x) = \frac{f^{(n+1)}(s)}{(n+1)!} \prod_{j=0}^{n} (x-x_j)$$

**Exact pointwise error.**

Similar to Taylor's theorem, it depends highly on the choice of the interpolation points.
Moreover:

\[ |f(x) - p_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |\tilde{E}_{n+1}(x)| \]

where \( M_{n+1} = \max_{t \in [a,b]} |f^{(n+1)}(t)| \)

\[ \tilde{E}_{n+1}(x) = \prod_{j=0}^{n} (x - x_j) \]

Proof is detailed, will not go through it, see text.

Two takeaway points:

1. Only useful if \( M_{n+1} \) can be computed.
2. The interpolation error highly depends on where the nodes \( x_j \) are located.

This will be very important later on.

The cost of evaluating \( p_n \) depends on the form it is written in.

**Lagrange Form:**

\[ p_n(x) = \sum_{k=0}^{n} y_k L_k(x) \]

\[ L_k(x) = \prod_{j=0, j \neq k}^{n+1} \frac{x - x_j}{x_k - x_j} \]

\[ n \text{ adds} \]

\[ 3 \text{ flops} \]

\[ 3(n+1) \text{ flops} \]

\[ \Rightarrow (n+1)3(n-1) \text{ flops to evaluate all } L_k \text{s.} \]

\( \Rightarrow \) Overall, \( O(n^3) \) flops to evaluate \( p_n \) in Lagrange form.

\[ \text{---} \]
Compare this with Horner's Method:

If the coefficients \( a_0, \ldots, a_n \) are known in

\[ p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \]

the we can rewrite \( p_n \) as:

\[
p_n(x) = a_0 + x \left( a_1 + x (a_2 + x (a_3 + \cdots + a_n x^{n-1}) \right)
\]

\[
= a_0 + x (a_1 + x (a_2 + x (a_3 + \cdots + a_n x^{n-2}) \right)
\]

\[
= a_0 + x (a_1 + x (a_2 + x (a_3 + \cdots + a_{n-1} x^{n-2}) \right)
\]

\[
= a_0 + x (a_1 + x (a_2 + x (a_3 + \cdots + a_{n-1} + a_n x^{n-1}) \right)
\]

\[
= a_0 + x (a_1 + x (a_2 + x (a_3 + \cdots + a_{n-2} + a_{n-1} x^{n-2}) \right)
\]

\[
= a_0 + x (a_1 + x (a_2 + x (a_3 + \cdots + a_{n-2} + a_{n-1} x^{n-1}) \right)
\]

\[
= a_0 + x (a_1 + x (a_2 + x (a_3 + \cdots + a_{n-2} + a_{n-1} x^{n-1}) \right)
\]

\[
= a_0 + x (a_1 + x (a_2 + x (a_3 + \cdots + a_{n-2} + a_{n-1} x^{n-1}) \right)
\]

\[
= a_0 + x \left( a_1 + \cdots + a_{n-1} + a_n x^{n-1} \right)
\]

This means that the Lagrange Form is very inefficient.

Is there a better form?

\[ \text{Q3} \]

The numerical stability of evaluating \( p_n \) in Lagrange form:

\[ \text{Short story: The basic Lagrange form } p_n(x) = \sum_{k=0}^{n} y_k L_k(x) \]

\[ \text{can be unstable (i.e., have large condition number).} \]

\[ \text{(Ex: overflow/underflow, roundoff error, etc.)} \]

Alternative form next class.
**Barycentric Form (a) of Interpolation**

The numerical stability of evaluating an interpolating polynomial can be fixed by rearranging its terms — this does not change what the actual interpolant is.

As motivation: Examine the barycentric coordinates on a triangle.

Ex:

The barycentric coordinates of a point $P$ inside a triangle with vertices $A, B, C$ are given by:

$$P = \alpha A + \beta B + \gamma C$$

with $\alpha + \beta + \gamma = 1$, $\alpha \geq 0$, $\beta \geq 0$, $\gamma \geq 0$.

The center of mass of the triangle is given by

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix}.$$
Idea: Replace \( A, B, C \) with functions that sum to 1.

Start with the Lagrange Form:

\[
p_n(x) = \sum_{k=0}^{n} \frac{\prod_{j=0}^{n} (x-x_j)}{\prod_{j=0}^{k} (x_k-x_j)} y_k \]

\[
= \sum_{k=0}^{n} \left( \frac{\prod_{j=0}^{n} (x-x_j)}{\prod_{j=0}^{k} (x_k-x_j)} \right) \frac{1}{x-x_k} \left( \frac{1}{\prod_{j=k+1}^{n} (x_k-x_j)} \right) y_k
\]

\[
\text{does not depend on } k
\]

\[
= \frac{\prod_{j=0}^{n} (x-x_j)}{\prod_{j=0}^{k} (x_k-x_j)} \sum_{k=0}^{n} \frac{1}{x-x_k} \left( \frac{1}{\prod_{j=k+1}^{n} (x_k-x_j)} \right) y_k
\]

\[
= q(x) \sum_{k=0}^{n} \frac{w_k}{x-x_k} y_k
\]

(First Barycentric Formula)

We can even further simplify this form by "dividing by 1".

The polynomial interpolant of the function \( 1 \) at the same nodes \( x_j \) is simply:

\[
1 = q(x) \sum_{k=0}^{n} \frac{w_k}{x-x_k} y_k
\]

(since \( y_k = 1 \)).

This interpolant is mathematically equivalent to 1.

Then

\[
p_n(x) = \frac{\sum_{k=0}^{n} \frac{w_k}{x-x_k} y_k}{q(x) \sum_{k=0}^{n} \frac{w_k}{x-x_k}}
\]

(Second Barycentric Formula)
This form is stable for any reasonable choice of $x_i$ (2004, Higham).

One should always use this form to do polynomial interpolation.

**Convergence of Polynomial Interpolation**

Let's examine the question of what happens as $n \to \infty$, i.e.,

$$\lim_{n \to \infty} \max_x |f(x) - p_n(x)| = ?$$

This is the $\infty$-norm.

The pointwise error is approximately:

$$\max_{x} \left| f^{(n+1)}(s) \right| \cdot \max_{x} \frac{n!}{(n+1)!} \left| x - x_i \right|$$

It's not obvious if this increases or decreases as $n \to \infty$.

**Example**

Runge's Function $f(x) = \frac{1}{1 + (3x)^2}$

This behavior is related to the fact that the function $f(x) = \frac{1}{1 + x^2}$ has a singularity at $x = \pm i$ in the complex plane. $f(i) = \frac{1}{1 + i \cdot i} = \frac{1}{1 - 1} = \frac{1}{0} = \infty$.

This dictates the radius of convergence of its Taylor series:

$$f(x) = 1 - x^2 + x^4 - x^6 + x^8 - x^{10} + \ldots$$

(This can be fixed, we'll see later on.)