LECTURE 1

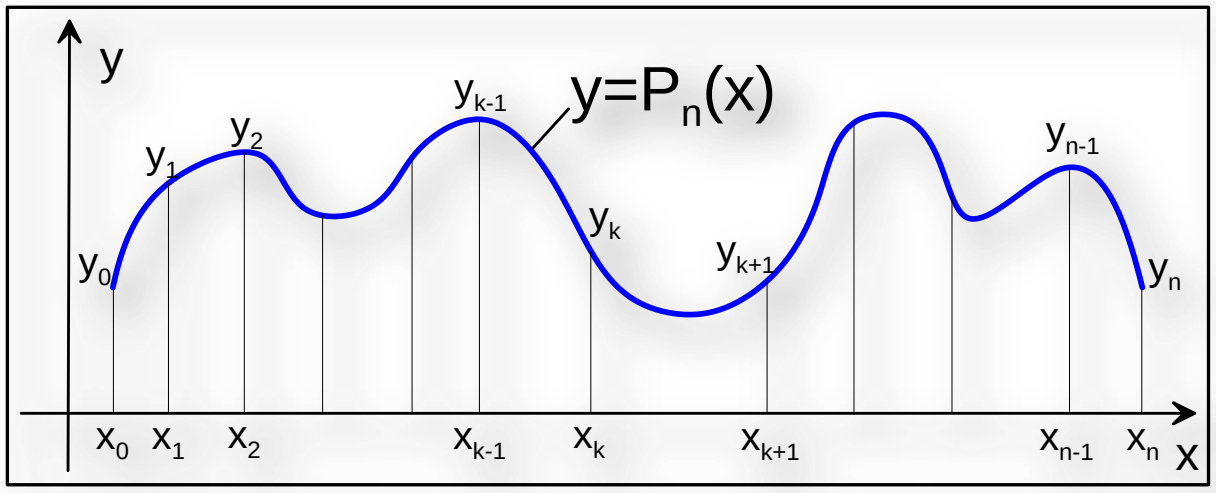
POLYNOMIAL INTERPOLATION

Formulation of the interpolation problem. The Lagrange method.

Assume the set of points $\{(x_j, y_j),\ j = 0,1,...,n\}$ is given. We will refer to these points as the **interpolation nodes**. We want to find the **interpolating polynomial**

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0 \equiv \sum_{k=0}^{n} a_k x^k$$

i.e., such that $P_n(x_j) = y_j \quad , \quad j = 0,1,...,n$



This means that the plot of the polynomial P_n should go through all the given nodes (see picture)

Three crucial questions arise:

1. Does such polynomial exist?
2. If it exists, is it unique?
3. How can we find it?

Let us first take the **„brute force"** approach. It means that we simply derive the linear system for the coefficients of the interpolating polynomial, which is

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} & x_1^n \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_j & x_j^2 & \cdots & x_j^{n-1} & x_j^n \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_j \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_j \\ \vdots \\ y_n \end{bmatrix}$$

In matrix/vector notation, we have $\boldsymbol{Wa = y}$

where $w_{j,k} = x_j^k \quad , \quad j,k = 0,1,...,n$ are the elements of the Van der Mond matrix.

In principle, this system can be solved numerically (i.e., by means of the Gauss Elimination). It can be shown that as long as all interpolation nodes are different, the Van der Mond matrix is nonsingular and the unique solution exists. Note that the order of the polynomial P_n is by one smaller than the number of the nodes. Otherwise, the interpolation problem would be either **undetermined** (infinite number of solutions exists) or **overdetermined**, i.e. there is no solution.

There exist much **smarter methods** than the "brute force" approach. **They are direct in the sense that they avoid solving a linear algebraic system**.

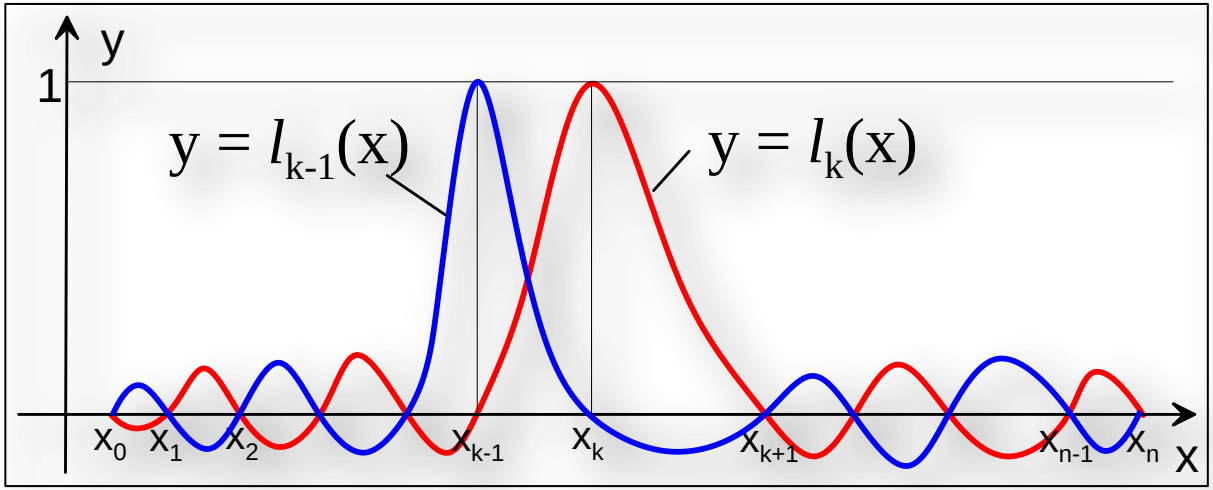
First consider the **Lagrange method.** The key idea is to introduce the special family of n^{th}-order polynomials, namely

$$l_k(x) = \frac{(x-x_0)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_0)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)} = \prod_{\substack{i=0 \\ i\neq k}}^{n} \frac{x-x_i}{x_k-x_i} \quad , \quad k=0,1,\ldots,n$$

Note that

$$l_k(x_j) = \underset{\substack{\text{Kronecker} \\ \text{symbol}}}{\delta_{jk}} = \begin{cases} 0 & \text{if} \quad j \neq k \\ 1 & \text{if} \quad j = k \end{cases}$$

The typical plots of the special Lagrange polynomials are depicted in the figure of the right.



Once these special polynomials are defined, the solution of the interpolation problem is immediate. It suffices to write

$$P_n(x) = \sum_{k=0}^{n} y_k l_k(x)$$

Indeed, we have $$P_n(x_j) = \sum_{k=0}^{n} y_k l_k(x_j) = \sum_{k=0}^{n} y_k \delta_{jk} = y_j \quad , \quad j = 0,1,...,n$$

An alternative (and theoretically useful) way to write the polynomial P_n is also

$$P_n(x) = \sum_{k=0}^{n} \frac{\omega_{n+1}(x)}{(x - x_k)\omega'_{n+1}(x_k)} y_k \qquad \text{(exercise !)}$$

where the following (n+1)-order polynomial has been used

$$\omega_{n+1}(x) = \prod_{k=0}^{n} (x - x_k) = (x - x_0)(x - x_1)...(x - x_n)$$

APPROXIMATION BY AN INTERPOLATING POLYNOMIAL

The interpolating polynomials are often used to **approximate** other functions. Assume that the interpolating nodes are defined as

$$\{(x_j, y_j),\ j = 0,1,...,n\} \qquad \text{where} \quad y_j = f(x_j) \quad , \quad j = 0,1,...,n$$

The key problem is what is accuracy of the approximation of the function f by the interpolating polynomial P_n.

We will show that the following result holds:

Theorem: Let $x_0, x_1, ..., x_n$ be distinct nodes and let x belongs to the domain of the given function $f \in C^{(n+1)}(I_x)$, where I_x is the smallest interval containing $x_0, x_1, ..., x_n$ and x. Then there exists such point $\xi \in I_x$ that the interpolation error at the point x is given by

$$E_n(x) \equiv f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)$$

Proof: Fix x and consider the function

$$g(t) = E_n(t) - \frac{E_n(x)}{\omega_{n+1}(x)}\omega_{n+1}(t) \quad , \quad x \neq x_k \quad , \quad k = 0,1,\ldots,n$$

The function g has exactly n+2 roots (zeros). Indeed, we have …

$$g(x_k) = \underbrace{E_n(x_k)}_{0} - \frac{E_n(x)}{\omega_{n+1}(x)}\underbrace{\omega_{n+1}(x_k)}_{0} = 0 \quad , \quad k = 0,1,\ldots,n$$

$$g(x) = E_n(x) - \frac{E_n(x)}{\cancel{\omega_{n+1}(x)}}\cancel{\omega_{n+1}(x)} = 0$$

Thus, the derivative $g^{(n+1)}$ has one root inside I_x, say $g^{(n+1)}(\xi) = 0 \quad , \quad \xi \in I_x$.

On the other hand, we have

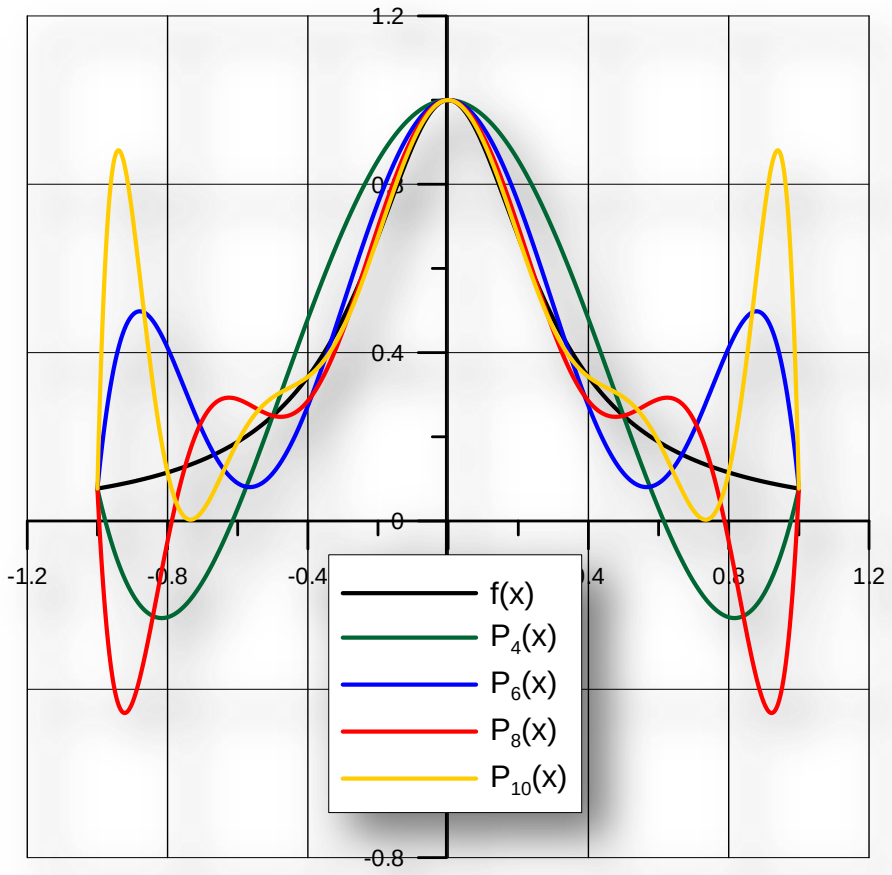
$$0 = g^{(n+1)}(\xi) = E_n^{(n+1)}(\xi) - \frac{E_n(x)}{\omega_{n+1}(x)}\omega_{n+1}^{(n+1)}(\xi) = f^{(n+1)}(\xi) - \underbrace{P_n^{(n+1)}(\xi)}_{\substack{= 0 \text{ since } P_n \text{ is} \\ \text{of the order } n!}} - \frac{E_n(x)}{\omega_{n+1}(x)}\underbrace{(n+1)!}_{\substack{\text{the coefficient} \\ \text{at } x^{n+1} \text{ in } \omega_{n+1}(x) \\ \text{is equal } 1}}$$

and the above formula for the approximation error follows immediately. **This ends the proof**.

For the equidistant nodes $x_k = x_0 + kh \;,\; k = 0,1,\ldots,n \;,\; h = \dfrac{x_n - x_0}{n}$

the following estimate of the approximation error can be obtained

$$\left|\omega_{n+1}(x)\right| = \prod_{k=0}^{n}\left|x - x_k\right| \le \tfrac{1}{4} h^{(n+1)} n! \quad \Rightarrow \quad \left|f(x) - P_n(x)\right| \le \frac{h^{n+1}}{4(n+1)} \max_{\xi \in [x_0, x_n]} \left| f^{(n+1)}(\xi) \right|$$



Increase of the polynomial's order does not necessarily improve the approximation!

Consider the polynomial approximations of increasing order of the rational function

$$f(x) = \frac{1}{1 + 10x^2}$$

The interpolation interval is [-1,1] and the interpolation nodes are equidistant. The result of the interpolation is depicted in the following figure.

The characteristic “lobes” appear on the plots of the interpolating polynomials near the ends of the interpolation interval. The amplitude of these unwanted “oscillations” become larger when the order of interpolating polynomial increases. This is a demonstration of the **Runge effect**.

The **remedy** for this problem is to use **nonequidistant nodes**. Even better, there exists the optimal choice of such nodes. For the interval [-1,1] these nodes are given as

$$x_k^T = \cos\left(\frac{2k+1}{n+1}\frac{\pi}{2}\right) \quad , \quad k = 0,1,...,n$$

i.e., they are roots of the (n+1)-order Chebyshev polynomial

$$T_{n+1}(x_k^T) = 0 \quad , \quad k = 0,1,...,n$$

The Chebyshev polynomials are the very important family of polynomials defined by the following recurrence relation

$$T_0(x) \equiv 1 \ , \ T_1(x) = 1$$
$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x) \ , \ k = 1,2,...$$

They are also related to **trigonometric functions**, since $\cos kx = T_k(\cos x)$

The crucial property of the Chebyshev polynomials is $\exists\, x \in [-1,1] \;\; |T_k(x)| \le 1 \quad , \quad k = 0,1,...$

Hence, one gets the estimate

$$T_{n+1}(x) = 2^n \prod_{k=0}^{n} (x - x_k^T) \;\; \Rightarrow \;\; \left| \prod_{k=0}^{n} (x - x_k^T) \right| \le \frac{1}{2^n}$$

Moreover, it can be shown that with any other choice of the interpolating nodes inside [-1,1]

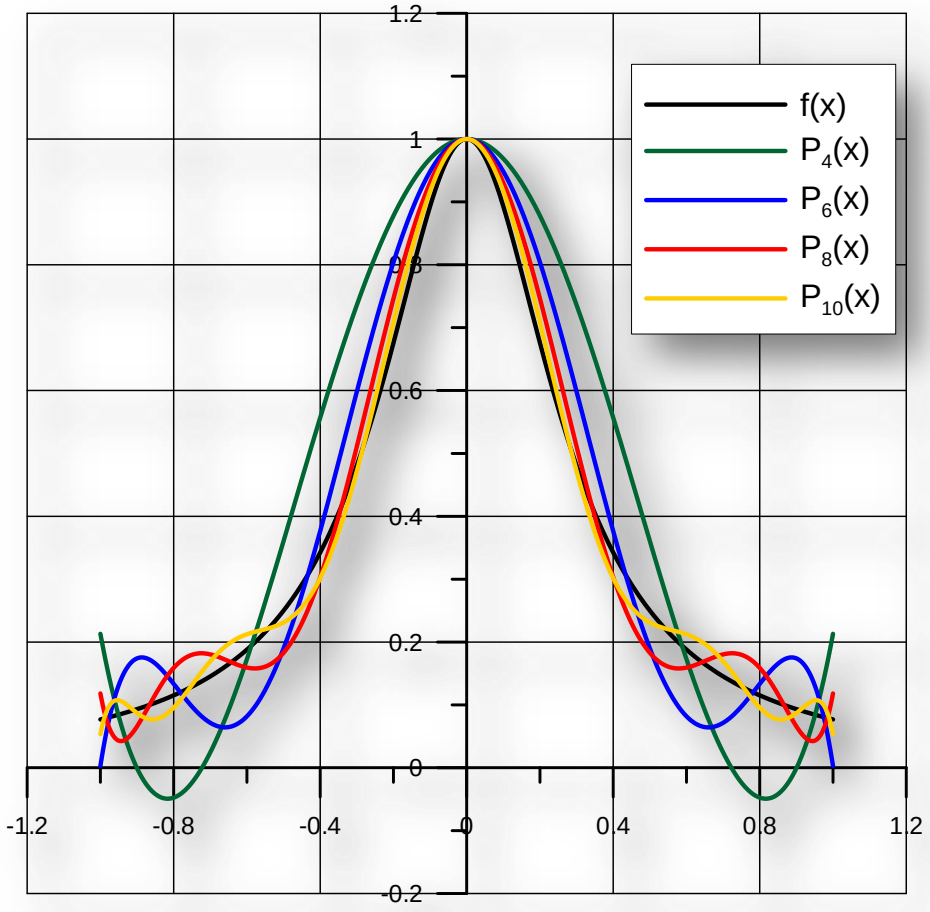
$$\max_{t \in [-1,1]} \left| \prod_{k=0}^{n} (t - z_k) \right| \ge \frac{1}{2^n} \quad , \quad z_k \in [-1,1] \;\; , k = 0,1,...,n$$

i.e., the choice of the Chebyshev nodes is **optimal** in this sense that it minimizes the absolute value of the polynomial $\omega_{n+1}(x)$ within the interval [-1,1].

As a result, we get the following estimate of the approximation error

$$|f(x) - P_n(x)| \le \frac{1}{2^n (n+1)!} \max_{\xi \in [-1,1]} |f^{(n+1)}(\xi)|$$

Note that for larger values of n the expression in front of the $(n+1)^{th}$ derivative diminishes with increasing n faster that the analogical expression for the equidistant nodes. This means that a reasonable accuracy may be achieved even for the functions with quickly growing high-order derivatives. Still, especially "pathological" functions exist such that even the Chebyshev interpolation fails to provide good results. The application of the Chebyshev nodes to our test function yields effect as in the figure.



For the general interval $[a,b]$, the Chebyshev nodes are

$$x_k^T = \frac{b-a}{2}\cos\left(\frac{2k+1}{n+1}\frac{\pi}{2}\right)+\frac{a+b}{2}$$

The corresponding error estimate is

$$\left|f(x)-P_n(x)\right| \le \frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!}\max_{\xi\in[a,b]}\left|f^{(n+1)}(\xi)\right|$$

POLYNOMIAL INTERPOLATION BY MEANS OF THE NEWTON METHOD

In the end, we will discuss shortly an alternative (and very effective) approach to polynomial interpolation which is known as the **Newton method.**

As usual, we define the interpolation nodes $\{(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)\}$. Next, we construct recursively the set of quantities called the **divided differences** – see formulae of the right.

$$
\begin{cases}
Y_k = y_k \quad , \quad k = 0,1,....,n \\
Y_{k,k+1} = \dfrac{y_{k+1} - y_k}{x_{k+1} - x_k} = \dfrac{Y_{k+1} - Y_k}{x_{k+1} - x_k} \quad , \; k = 0,1,...,n-1 \\
Y_{k,k+1,k+2} = \dfrac{Y_{k+1,k+2} - Y_{k,k+1}}{x_{k+2} - x_k} \quad , \quad k = 0,1,...,n-2 \\
\vdots \\
Y_{k,k+1,...,k+m} = \dfrac{Y_{k+1,k+2,...,k+m} - Y_{k,k+1,...,k+m-1}}{x_{k+m} - x_k} \quad , \quad k = 0,1,...,n-m \\
\vdots \\
Y_{0,1,...,n} = \dfrac{Y_{1,2,...,n} - Y_{0,1,...,n-1}}{x_n - x_0} \quad , \quad m = n
\end{cases}
$$

Along with the divided differences, the following polynomials are introduced

$$\omega_k = (x - x_0)(x - x_1)...(x - x_k) \equiv \prod_{j=0}^{k} (x - x_j) \ , \ k = 0,...,n-1$$

Finally, the interpolating polynomial P_n is constructed as follows.

$$P_n(x) = Y_0 + \sum_{k=1}^{n} Y_{0,1,..,k}\, \omega_{k-1}(x) =$$

$$= y_0 + Y_{0,1}(x - x_0) + Y_{0,1,2}(x - x_0)(x - x_2) + ... + Y_{0,1,2,...,n}(x - x_0)(x - x_2)...(x - x_{n-1})$$

The above formula is far from obvious and its proof is rather long and technical. The Reader should refer for the standard books on numerical methods for details.

For the illustration we will consider a simple case where $n = 2$.

According to the general formula, the second order interpolating polynomial can be expressed as

$$P_2(x) = Y_0 + Y_{0,1}(x - x_0) + Y_{0,1,2}(x - x_0)(x - x_1)$$

The calculations showing that this polynomial indeed interpolates through the given nodes go as follows:

$$\begin{cases} P_2(x_0) = y_0 \\ P_2(x_1) = y_0 + Y_{0,1}(x_1 - x_0) = y_0 + \dfrac{y_1 - y_0}{x_1 - x_0}(x_1 - x_0) = y_1 \\ P_2(x_2) = y_0 + Y_{0,1}(x_2 - x_0) + Y_{0,1,2}(x_2 - x_0)(x_2 - x_1) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0) + \\ + \dfrac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}(x_2 - x_0)(x_2 - x_1) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0) + y_2 - y_1 - \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_1) = \\ = y_0 + y_2 - y_1 + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0 + x_1 - x_2) = y_2 \end{cases}$$

The numerically efficient method to calculate the interpolating polynomial in the Newton's form is the **Horner algorithm**. The pseudo code for this algorithm can be written as

```
% Horner summation of the Newton polynomial
% Vector w stores necessary divided differences
%  w(k) = Y_{0,1,...,k}  ,  k = 0,1,...,n
s ← W(n)
FOR k = n-1:0:-1   !loop runs backwards!
   s ← s·(x - x_k) + W(k)
END
RETURN  s
```