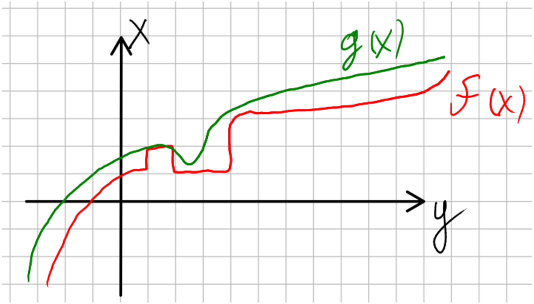
Mathematical Foundations of Computing

Serhii Denysov

February 2026

Approximation problem intuition

- Let $f(x)$ represent a process / data / simulation.
- Often $f(x)$ is "**bad**"
- We want a "**good**" function $g(x)$, such that $g(x) \approx f(x)$ in the our domain, for example $\forall x \in [a, b]$



Good versus evil again – is it a Philosophy course???

- What is a **bad** function $f(x)$?
- What is a **good** function $g(x)$?

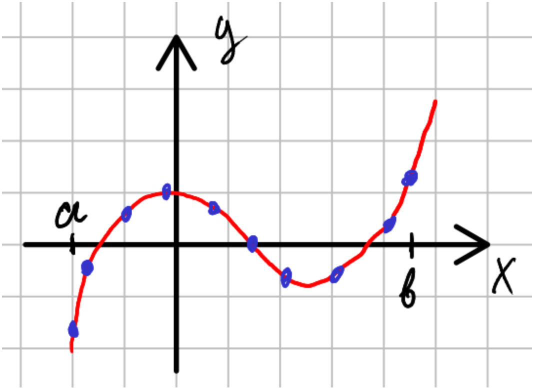
Good versus evil again – is it a Philosophy course???

- What is a **bad** function $f(x)$?
- What is a **good** function $g(x)$?
- Some ideas of "bad":
  - Very complicated – can't compute in reasonable time. Imagine salary-performance relation.
  - Not smooth. No derivatives.
  - No formula. Can not reason with mathematical toolset.
- Some ideas of "good":
  - $g(x)$ is easy to compute.
  - Smooth enough - has enough derivatives ($g'(x), g''(x)$ etc. – works for optimization, ODEs, PDEs)
  - Easy to deal with mathematically.

How to build function, which could be approximately equal?

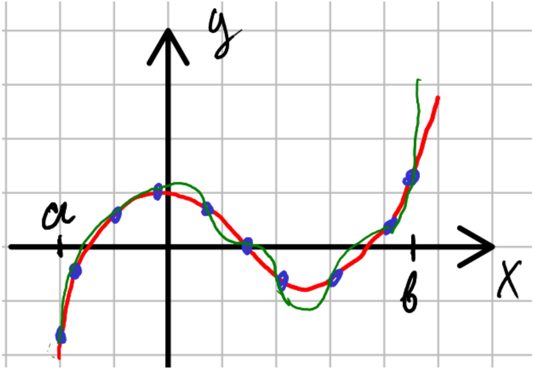
How to build function, which could be approximately equal?

Force to go through the same points – **interpolation** idea!



Interpolation approach to approximation problem

Force to go through the same points – **interpolation** idea!



Interpolation – formal

Interpolation problem (statement)

- Given some segment $[a, b]$ – where we need to approximate f
- Select a set of x points $x_i, x_i \in [a, b], x_0 = a, x_n = b, x_{i+1} > x_i$
- And calculate $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = f(x_i)$

- Now we need a "good" function $g(x)$ such that:

$$g(x_i) = y_i \text{ for all } i = 0, \ldots, n.$$

What type of the simplest function comes to you mind?

What if we have 2 or 3 points?

Polynomial interpolation

We can choose $g(x)$ as a **polynomial**!

$$p_n(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n.$$

- If we have **2 points** – what is the simplest polynomial?

Polynomial interpolation

We can choose $g(x)$ as a **polynomial**!

$$p_n(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n.$$

- If we have **2 points** – what is the simplest polynomial?
- **Degree 1** – line, $g(x) = c_0 + c_1 x$.
- If we have **3 points** – what is the simplest polynomial?

Polynomial interpolation

We can choose $g(x)$ as a **polynomial**!

$$p_n(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n.$$

- If we have **2 points** – what is the simplest polynomial?
- **Degree 1** – line, $g(x) = c_0 + c_1 x$.
- If we have **3 points** – what is the simplest polynomial?
- **Degree 2** – parabola, $g(x) = c_0 + c_1 x + c_2 x^2$.
- If we have **4 points**, it will be $g(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3$

Is it always unique? Under which condition?

Polynomial interpolation

So let $g(x)$ be a polynomial:

$$p_n(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n.$$

- Unknowns: coefficients $c_0, \ldots, c_n$.
- Conditions: $p_n(x_i) = y_i$ for $i = 0, \ldots, n$.

What we got here?

Polynomial interpolation

So let $g(x)$ be a polynomial:

$$p_n(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n.$$

- Unknowns: coefficients $c_0, \ldots, c_n$.
- Conditions: $p_n(x_i) = y_i$ for $i = 0, \ldots, n$.

What we got here?

SLE with $n + 1$ equations for $n + 1$ unknowns!

Polynomial interpolation – 3 points

Parabola through 3 points
Let $p_2(x) = c_0 + c_1 x + c_2 x^2$.

Conditions:

$$p_2(x_0) = y_0, \quad p_2(x_1) = y_1, \quad p_2(x_2) = y_2.$$

This is a **SLE (linear system)** for $c = (c_0, c_1, c_2)$.

$$\begin{cases} c_0 + c_1 x_0 + c_2 x_0^2 = y_0 \\ c_0 + c_1 x_1 + c_2 x_1^2 = y_1 \\ c_0 + c_1 x_2 + c_2 x_2^2 = y_2 \end{cases}$$

Polynomial interpolation – SLE matrix

The matrix of SLE:

$$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}.$$

General case – $n+1$ points:

$$V\mathbf{c} = \mathbf{y}, \qquad V = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix}.$$

V is called a Vandermonde matrix.

Polynomial interpolation – SLE properties

Unique solution

- If $x_0, \ldots, x_n$ are **distinct**, then V is non-singular (invertible).

Polynomial interpolation – SLE properties

Unique solution

- If $x_0, \ldots, x_n$ are **distinct**, then V is non-singular (invertible).
- So, the interpolation polynomial exists and is unique.

Polynomial interpolation – SLE properties

Unique solution

- If $x_0, \ldots, x_n$ are **distinct**, then V is non-singular (invertible).
- So, the interpolation polynomial exists and is unique.

But we can have problems finding it with algorithms

- Sadly, **invertible** does not mean **numerically stable** – remember Hilbert matrix.
- Vandermonde matrices can have huge condition numbers.
- Especially for the most intuitive setup:
  - large degree n
  - equally spaced nodes on a large interval
  - monomial basis $1, x, x^2, \ldots$

Practice – build parabola via SLE.

Polynomial interpolation – more approaches

Can we build $p_n(x)$ without solving a system?

- Construct $p_n(x)$ directly from the nodes (data).
- **Idea**: Combine from terms which catch one node behavior only.
- What properties they could have?

Polynomial interpolation – more approaches

Can we build $p_n(x)$ without solving a system?

- Construct $p_n(x)$ directly from the nodes (data).
- **Idea**: Combine from terms which catch one node behavior only.
- What properties they could have?

We need polynomials of degree n $L_0(x), \ldots, L_n(x)$ such that:

$$L_i(x_j) = \delta_{ij} = \begin{cases} 1, & j = i, \\ 0, & j \neq i. \end{cases}$$

Lagrange approach

- If we have such L_i, then we can write:

Lagrange approach

- If we have such L_i, then we can write:

$$p_n(x) = \sum_{i=0}^{n} y_i L_i(x)$$

But how to build $L_i(x)$?

Could we use our nodes (data) x_i?

Lagrange approach

- If we have such L_i, then we can write:

$$p_n(x) = \sum_{i=0}^{n} y_i L_i(x)$$

But how to build $L_i(x)$?

Could we use our nodes (data) x_i?

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

Lagrange approach

- If we have such L_i, then we can write:

$$p_n(x) = \sum_{i=0}^{n} y_i L_i(x)$$

But how to build $L_i(x)$?

Could we use our nodes (data) x_i?

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

- $L_i(x_i) = 1$ (check - all factors become 1)
- if $L_i(x_k) = 0, k \neq i$ (check - one factor becomes 0)

Interpolation polynomial in Lagrange form

$$p_n(x) = \sum_{i=0}^{n} y_i \, L_i(x), \qquad L_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}.$$

Same polynomial as from Vandermonde system, but constructed directly.

Let's practice!

Interpolation error

- With interpolation, $g(x_i) = f(x_i)$. So, it's precise at the nodes.

- But what about **error between nodes**?

If f is $(n+1)$ times differentiable, then for every x there exists $\xi = \xi(x)$ such that:

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n}(x - x_i).$$

The error depends on $f^{(n+1)}$ and on the selection of nodes.

Could we prove the error formula above?

Practice: try to obtain the error formula!

Idea of the flow:

- Define $\omega(x) = \prod_{i=0}^{n}(x - x_i)$.
- Consider $F(t) = f(t) - p_n(t) - C\omega(t)$.
- Choose C such that $F(x) = 0$ and $F(x_i) = 0$ for all nodes.
- Apply Rolle's theorem repeatedly.

Goal: show $C = \dfrac{f^{(n+1)}(\xi)}{(n+1)!}$**.**

Runge phenomenon

Classical example of interpolation failure:

$$f(x) = \frac{1}{1+25x^2}, \qquad x \in [-1, 1].$$

- Take **equally spaced** nodes $x_i = -1 + i \cdot \frac{2}{n}$.
- Build the global interpolation polynomial $p_n(x)$.

Let's check with our code!

Runge phenomenon

Why it behaves badly despite "good looking" error formula)?

Recall:

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i).$$

- For Runge function, high derivatives $f^{(n+1)}$ grow fast with n.
- For equally spaced nodes on $[-1, 1]$, the product term

$$\left| \prod_{i=0}^{n} (x - x_i) \right|$$

becomes **very large near the endpoints**.

So the bound can get worse when n grows!

Reflection on polynomial interpolation

Pros and cons of vanilla polynomial interpolation?

Reflection on polynomial interpolation

Pros and cons of vanilla polynomial interpolation?

- Clear and easy to build. *Really?? Hope so...*
- Works well for some functions.
- Fast to calculate for small number of points.

- Large n can lead to oscillations (Runge-type behaviour).
- Vandermonde can be ill-conditioned: unstable coefficients.
- High degree polynomial – not so "good" function actually!

Reflection on polynomial interpolation

Pros and cons of vanilla polynomial interpolation?

- Clear and easy to build. *Really?? Hope so...*
- Works well for some functions.
- Fast to calculate for small number of points.

- Large n can lead to oscillations (Runge-type behaviour).
- Vandermonde can be ill-conditioned: unstable coefficients.
- High degree polynomial – not so "good" function actually!

Idea: use <u>piecewise</u> polynomials!

Piecewise interpolation

- Linear piecewise interpolation.
- Cubic piecewise interpolation. Natural cubic spline.

Natural cubic spline

We seek $S(x)$ such that:

- $S(x_i) = y_i$ for all nodes
- S is piecewise cubic
- S, S', S'' are continuous on $[x_0, x_n]$
- $S''(x_0) = 0$ and $S''(x_n) = 0$

This gives a unique spline.

Natural cubic spline

How do we compute it? A standard approach: solve for second derivatives at nodes:

$$m_i = S''(x_i), \qquad i = 0, \dots, n.$$

Then $m_0 = 0$, $m_n = 0$, and interior $m_1, \dots, m_{n-1}$ satisfy a tridiagonal linear system.

Tridiagonal SLE = fast and stable computations.

Natural cubic spline

Tridiagonal system (given) Let $h_i = x_{i+1} - x_i$.
For $i = 1, \ldots, n-1$:

$$h_{i-1}m_{i-1} + 2(h_{i-1} + h_i)m_i + h_i m_{i+1} = 6\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right).$$

This is the main computational formula.

After we have m_i, how do we get $S(x)$? On each interval $[x_i, x_{i+1}]$:

$$S_i(x) = \frac{m_i(x_{i+1} - x)^3}{6h_i} + \frac{m_{i+1}(x - x_i)^3}{6h_i} + \left(y_i - \frac{m_i h_i^2}{6}\right) \frac{x_{i+1} - x}{h_i} + \left(y_{i+1} - \frac{m}{}\right.$$

Ready to evaluate $S(x)$ anywhere.

Practice: spline as "equations from conditions"

- For $n = 3$ (4 nodes), how many cubic pieces?
- How many coefficients total?
- List the conditions: interpolation $+ S', S''$ continuity $+$ boundary.