Geometric Modeling Lecture 05

Rhaleb Zayer Hans-Peter Seidel

Summer Term 2018

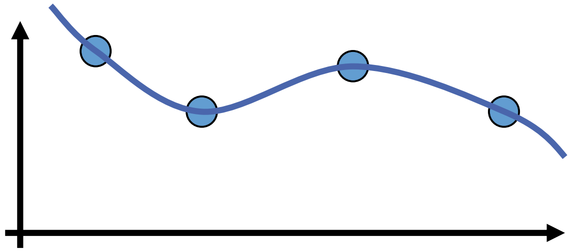
Announcements

- Programming Tutorial groups?

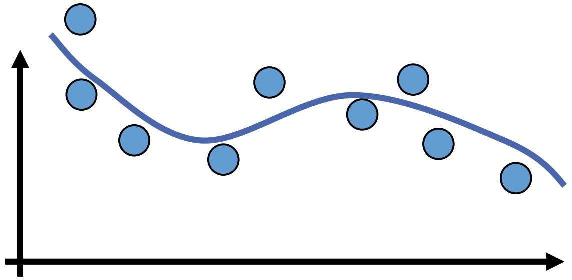
- Next week (08.05.18), the afternoon tutorial session will not take place.
  - Please, try to attend either the Monday, or the Tuesday morning session.
  - Optionally, a makeup session can be scheduled (after Wednesday's lecture).

- Monday tutorial sessions in room 23 MPII building 16:15—17:45

Interpolation & Approximation



Interpolation



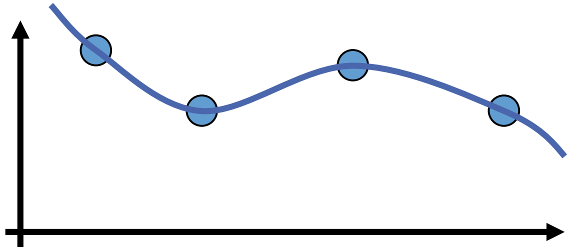
Approximation

Interpolation

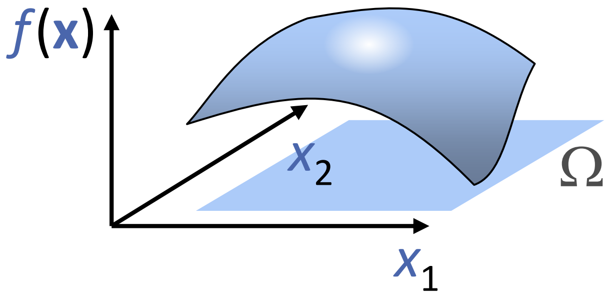
General & Polynomial Interpolation

Interpolation Problem

- Our first attempt at modeling smooth objects:
    - Given a set of points along a curve or surface
    - Choose basis functions that span a suitable function space
        - Smooth basis functions
        - Any linear combination will be smooth, too
    - Find a linear combination such that the curve/surface interpolates the given points



General Formulation



- Settings:
  - Domain $\Omega \subseteq \mathbb{R}^d$, mapping to $\mathbb{R}$.
  - Looking for a function $f : \Omega \to \mathbb{R}$.
  - Basis set: $B = \{b_1, \dots, b_n\}, b_i : \Omega \to \mathbb{R}$.
  - Represent f as linear combination of basis functions:

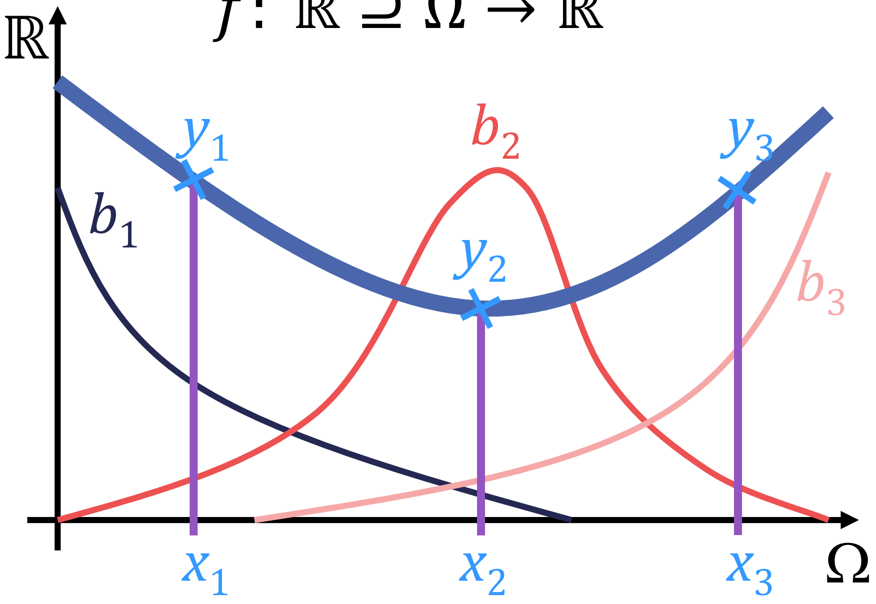
$$f_\lambda(\mathbf{x}) = \sum_{k=0}^{n} \lambda_i b_i(\mathbf{x})$$

i.e. f is just determined by $\lambda = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix}$.

  - Function values: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$, $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$
  - We want to find λ such that: $f_\lambda(\boldsymbol{x}_i) = y_i$ for all i

Illustration

$f\colon \mathbb{R} \supseteq \Omega \to \mathbb{R}$



1D Example

Solving the Interpolation Problem

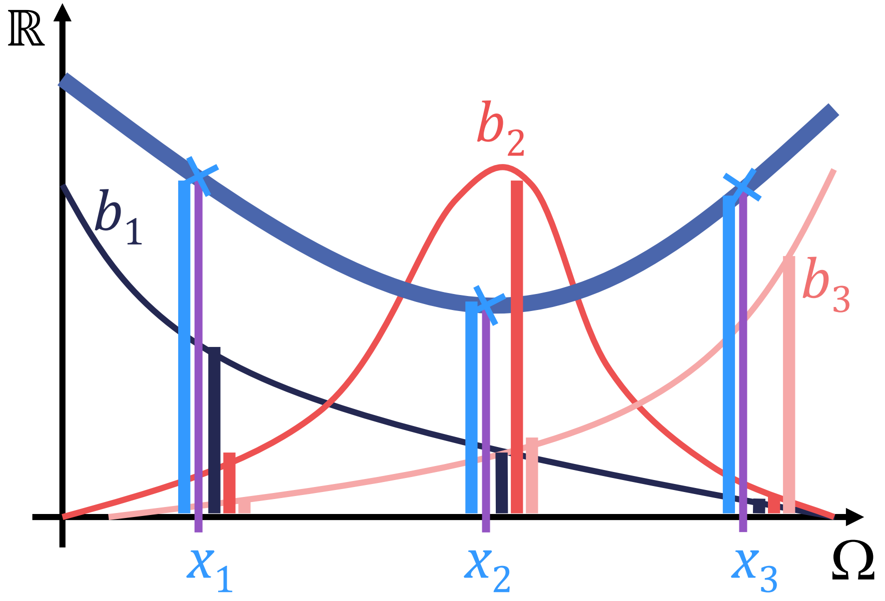
- Solution: linear system of equations
  - Evaluate basis functions at points $\mathbf{x}_\mathrm{i}$:

$$\forall i \in \{1,\ldots,n\} : \sum_{i=1}^{n} \lambda_i b_i(\mathbf{x}_i) = y_i$$

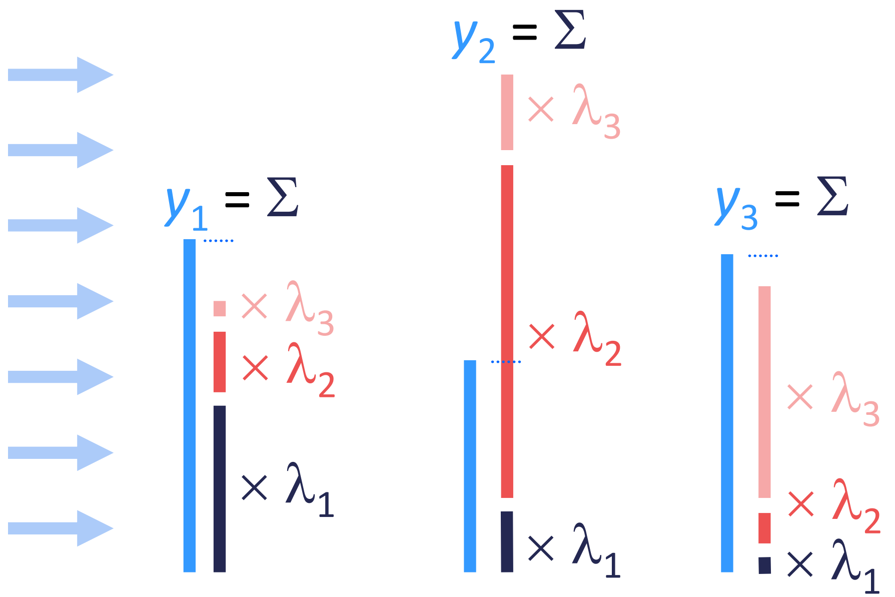
  - Matrix form:

$$\begin{pmatrix} b_1(\mathbf{x}_1) & \Lambda & b_n(\mathbf{x}_1) \\ \mathrm{M} & & \mathrm{M} \\ b_1(\mathbf{x}_n) & \Lambda & b_n(\mathbf{x}_n) \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \mathrm{M} \\ \lambda_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \mathrm{M} \\ y_n \end{pmatrix}$$

Illustration

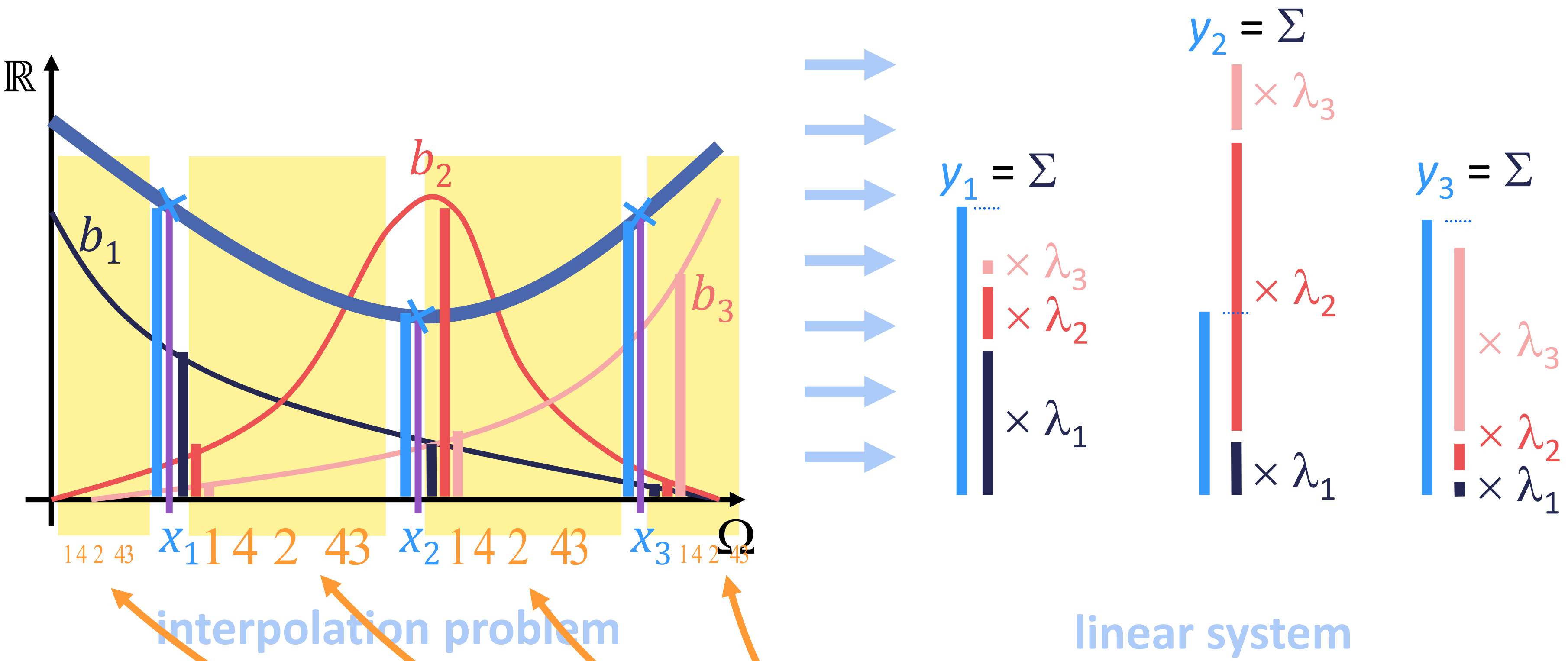


interpolation problem



linear system

Illustration



anything in between
does not matter
(determined by basis only)

Example

- Example: Polynomial Interpolation
  - Monomial basis B = {1, x, x^2, x^3, ..., $x^{n\text{-}1}$}
  - Linear system to solve:

$$\begin{pmatrix} 1 & x_1 & \Lambda & x_1^{n-1} \\ 1 & x_2 & \Lambda & x_2^{n-1} \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ 1 & x_n & \Lambda & x_n^{n-1} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \mathrm{M} \\ \lambda_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \mathrm{M} \\ y_n \end{pmatrix}$$

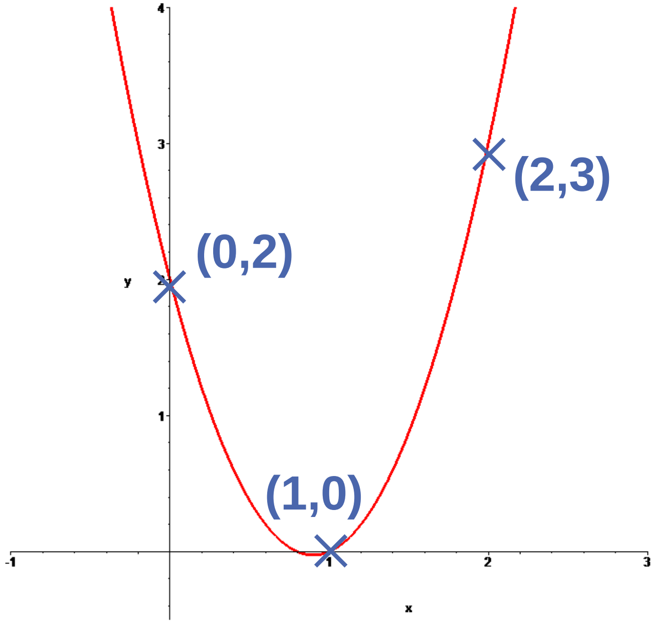
"Vandermonde Matrix"

Example with Numbers

- Example with numbers
  - Quadratic monomial basis B = {1, x, x^2}
  - Function values: {(0,2), (1,0), (2,3)}   [(x, y)]
  - Linear system to solve:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 3 \end{pmatrix}$$

  - Result: $\lambda_1 = 2$, $\lambda_2 = -9/2$, $\lambda_3 = 5/2$



Problems with interpolation

- The arising system matrix is generally dense
- Depending on the choice of the basis, the matrix can be ill-conditioned (difficult to invert/solve)

ill-conditioning example

- Consider the system
  - Clearly (1,1) is a solution

$$\begin{aligned} 1.000x_1 + 0.500x_2 &= 1.500, \\ 0.667x_1 + 0.333x_2 &= 1.000. \end{aligned}$$

- Now perturb the right hand side of the second equation by 0.001 (order 10^{-3})
  - The solution is then (0.000,3.000) (order 1)

$$\begin{aligned} 1.000x_1 + 0.500x_2 &= 1.500, \\ 0.667x_1 + 0.333x_2 &= \underset{\Uparrow}{0.999}. \end{aligned}$$

- Now consider perturbing the coefficient
  - The solution (2.000, -1.000)

$$\begin{aligned} 1.000x_1 + 0.500x_2 &= 1.500, \\ 0.667x_1 + \underset{\Uparrow}{0.334}x_2 &= 1.000. \end{aligned}$$

ill-conditioning

- Small change in the input data induces relatively large change in the output (solution)
- Thinking of equations as lines (hyperplanes), when the system is ill-conditioned the lines become almost parallel
    - Obtaining a solution (intersection) becomes difficult and imprecise

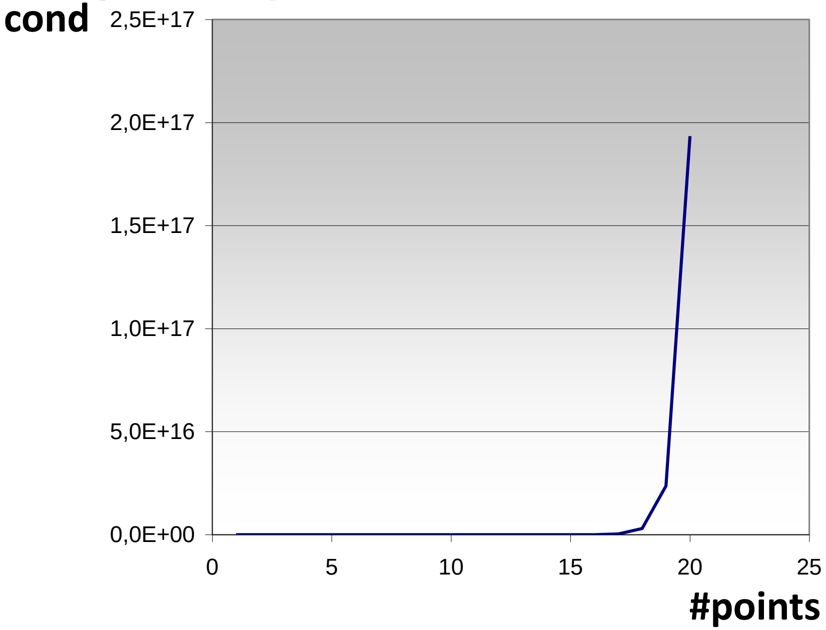
Condition number

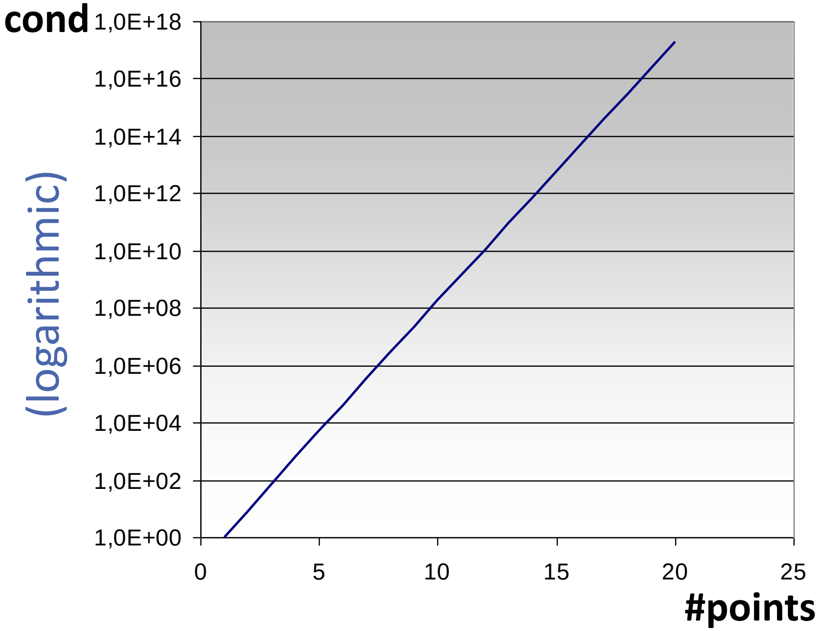
$$\kappa_2(A) = \frac{\max_{x \neq 0} \frac{\|Ax\|}{\|x\|}}{\min_{x \neq 0} \frac{\|Ax\|}{\|x\|}}$$

- Can be regarded as the ratio of highest eigenvalues / lowest eigenvalue
- When the condition number is high it reflects there is too much interdependence between the elements of the basis

Condition Number…

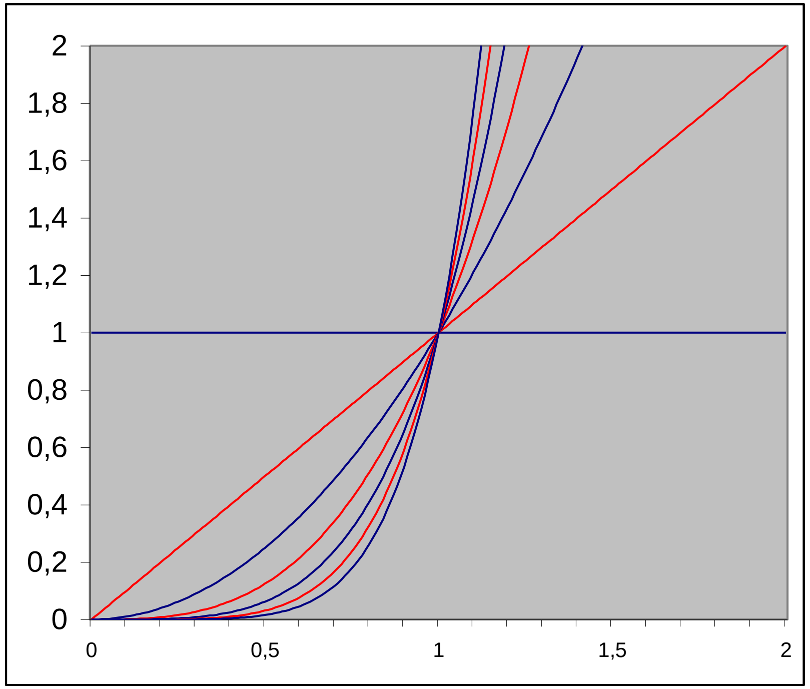
- The interpolation problem is ill conditioned:
  - For equidistant x_i, the condition number of the Vandermode matrix grows exponentially with n (maximum degree+1 = number of points to interpolate)





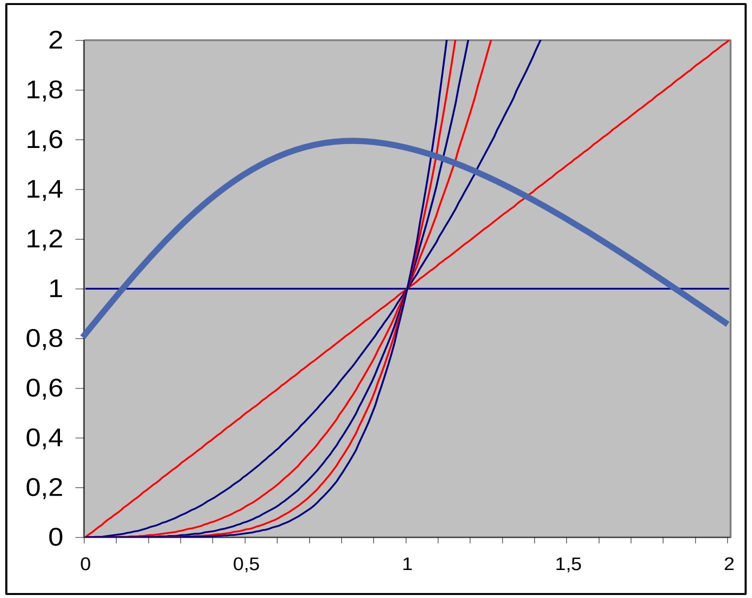
Why is that?

- Monomial Basis:
  - Functions become increasingly indistinguishable with degree
  - Only differ in growing rate (x^i growth faster than x^{i-1})



Monomial basis

Cancellation



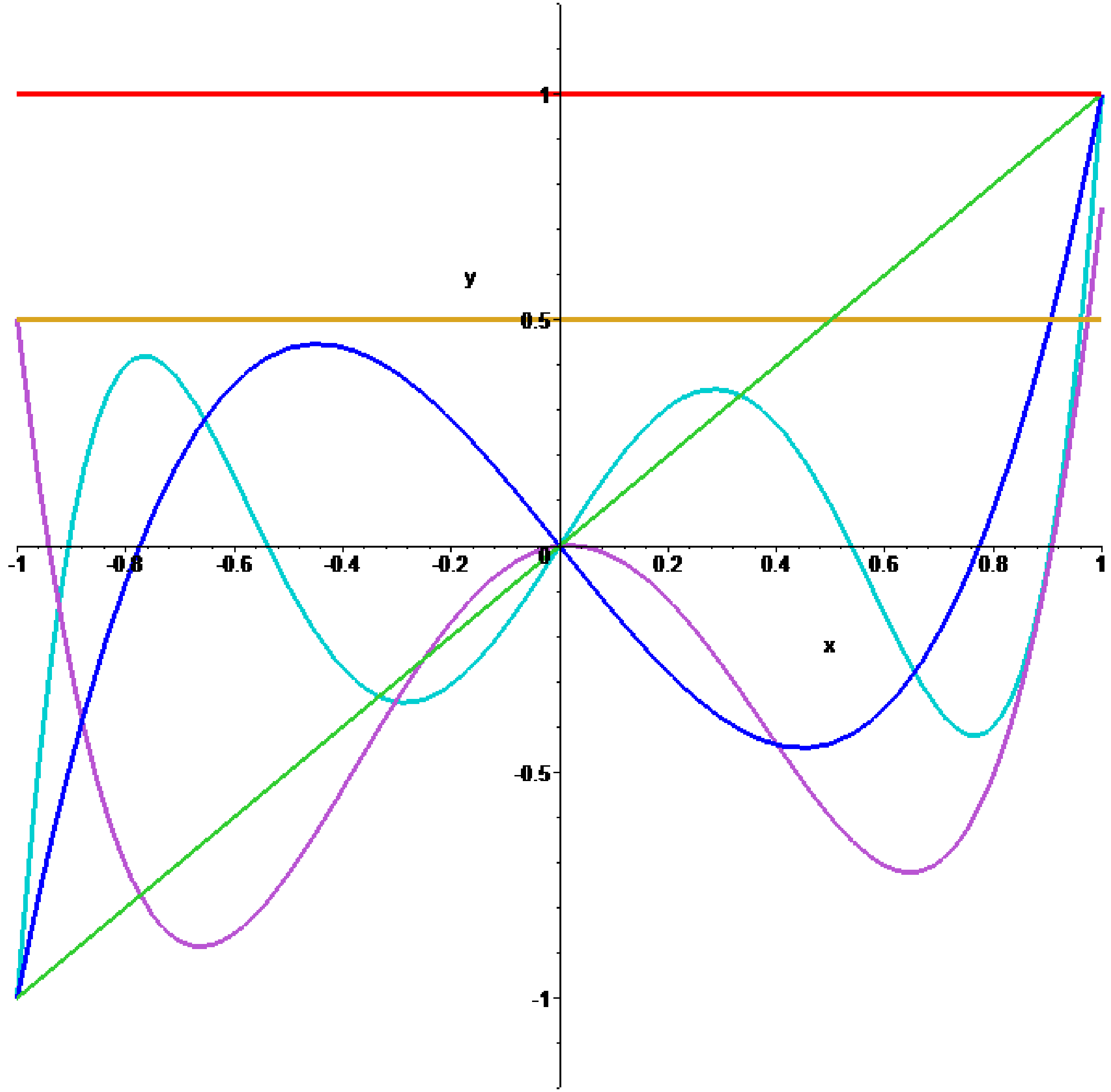
- Monomials:
  - From left to right in x-direction...
  - First 1 dominates
  - Then x grows faster
  - Then x^2 grows faster
  - Then x^3 grows faster
  - ...

Tendency:

- Well behaved functions often require alternating sequence of coefficients (left turn, right turn, left turn,...)
- *Cancellation* problems

The Cure…

- This problem can be fixed:
  - Use orthogonal polynomial basis
  - How to get one? → e.g. Gram-Schmidt orthogonalization



Alternative approach

- Can we avoid solving a system in the first place?

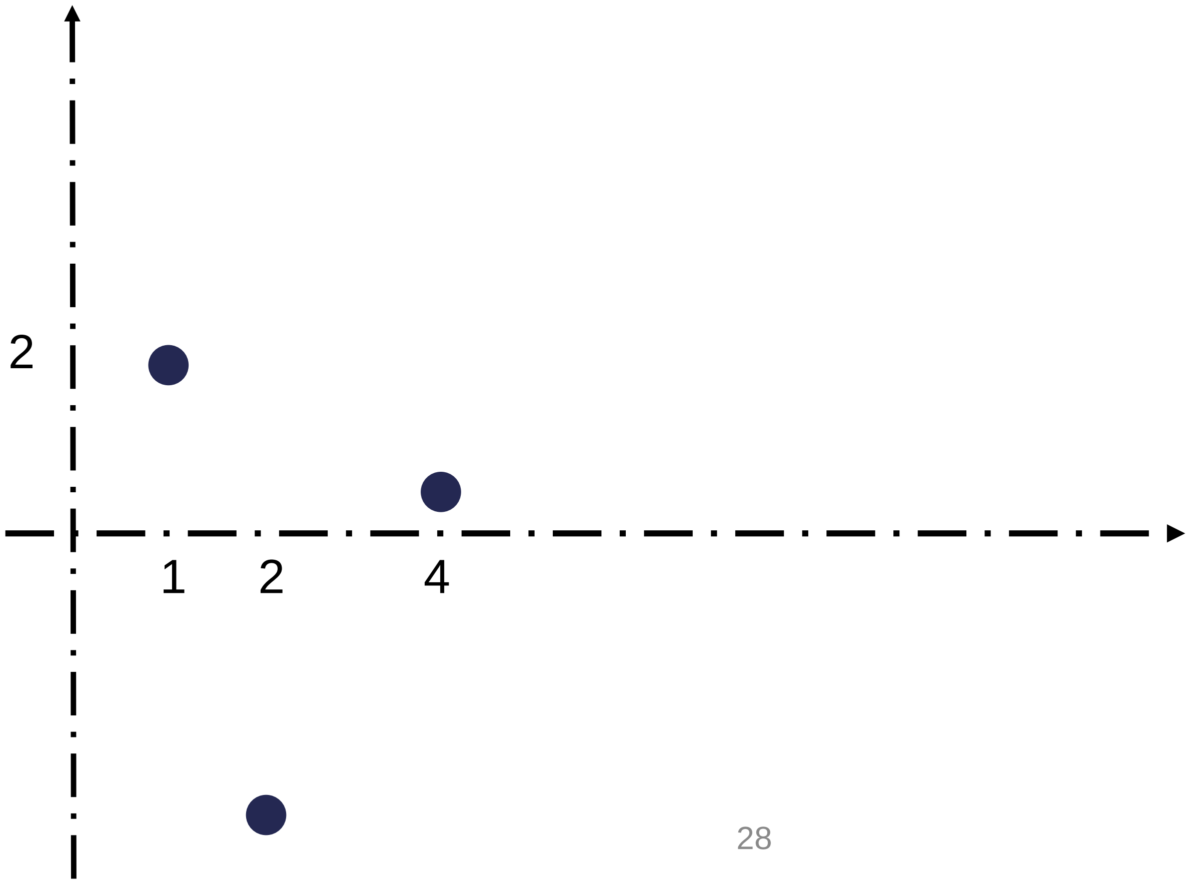
Alternative approach

- Can we avoid solving a system in the first place?

Think of a different basis!

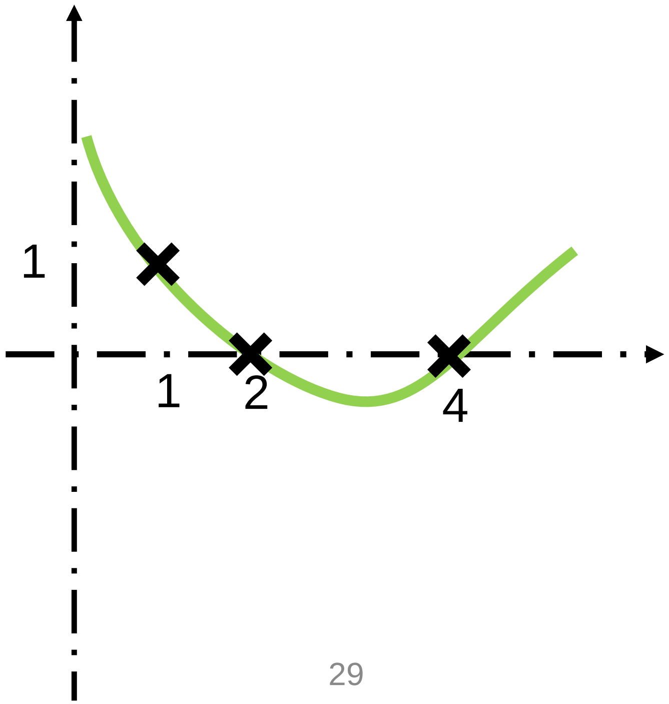
Alternative approach: Example

- Pass a quadratic polynomial through (1, 2), (2, −3), (4, 0.5)



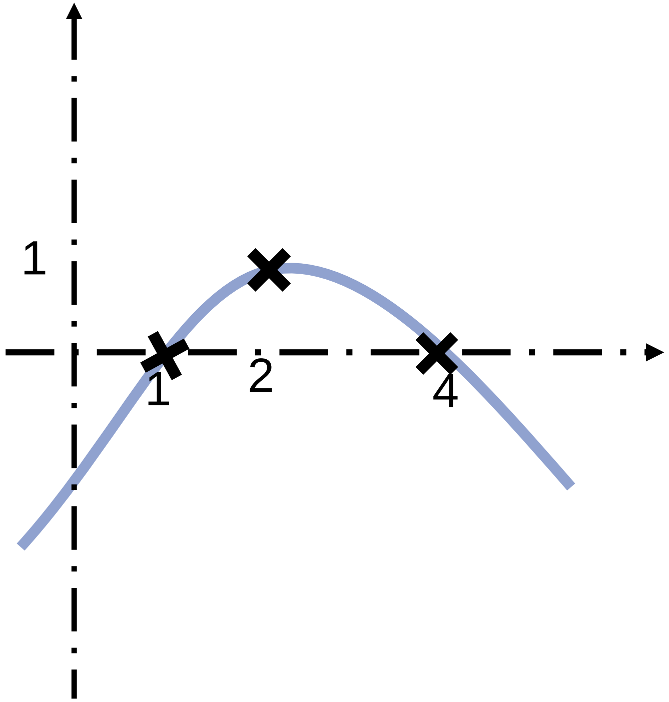
Alternative approach: Example

- Assume we can construct a quadratic polynomial P_0(x) such that it is equal to 1 at x_0, and equals zero at the other two points x_1, x_2:



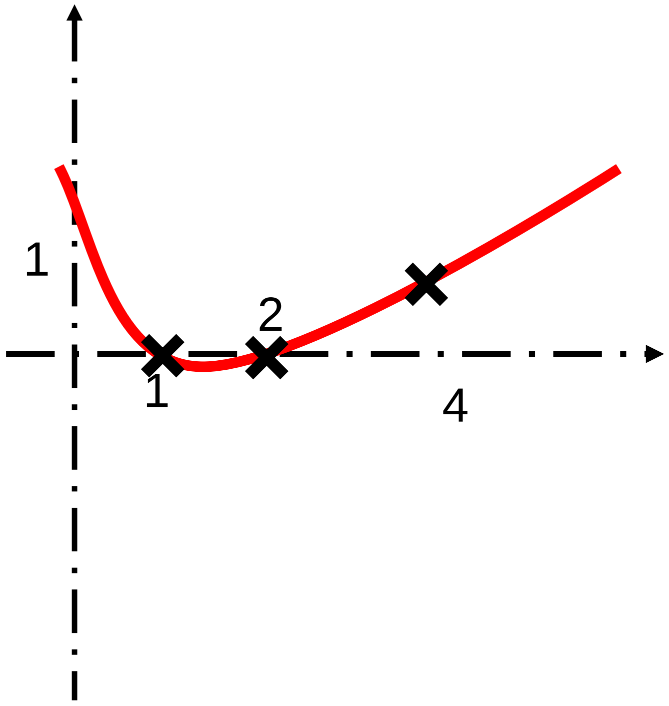
Alternative approach: Example

- $P_1(x)$, is constructed similarly and set equal to 1 at location x_1, and to zero at x_0, x_2:



Alternative approach: Example

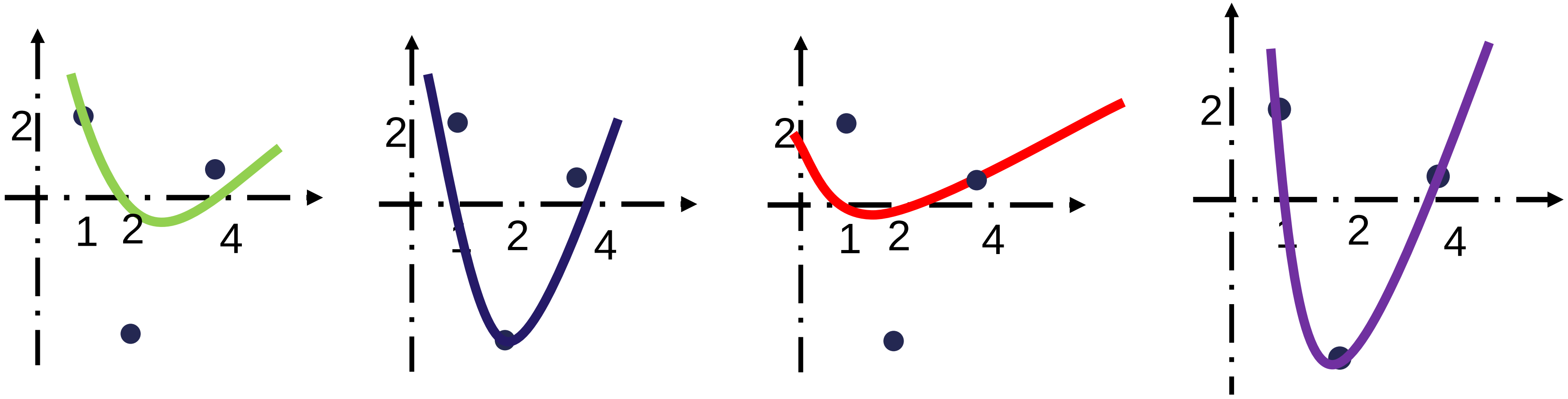
- $P_2(x)$ is set equal to 1 at location x_2, and to zero at x_0, x_1:



Alternative approach: Example

- Now, the idea is to *scale* each $P_i(x)$such that $P_i(x_i) = y_i$ and add them all together:

$$P(x) = y_0P_0(x) + y_1P_1(x) + y_2P_2(x)$$



$$2P_0(x) \quad + \quad -3P_1(x) \quad + \quad 0.5P_2(x) \quad = \quad P(x)$$

Alternative approach: general case

- Construction of general solution to the interpolation problem:
  - For a set of $n + 1$ points $\{(x_0, y_0), \ldots, (x_n, y_n)\}$, we seek a basis of polynomials l_i of degree n such that

$$l_i(x_j) = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$$

  - The solution to the interpolation problem is then given as

$$P(x) = y_0 l_0(x) + y_1 l_1(x) + \cdots + y_n l_n(x) = \sum_{i=0}^{n} y_i l_i(x).$$

Alternative approach: general case

- How can we find the polynomials $\boldsymbol{l_i(x)}$?
  - They are polynomials of degree n and have the following n roots

$$x_0, x_1, x_2, \ldots, x_{i-1}, x_{i+1}, x_{i+2}, \ldots, x_n,$$

  - They can be expressed as

$$\begin{aligned} l_i(x) &= C_i(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n) \\ &= C_i \prod_{j\neq i}(x-x_j) \end{aligned}$$

  - Since $\boldsymbol{l_i(x_i) = 1}$

$$1 = C_i \prod_{j\neq i}(x_i - x_j) \Rightarrow C_i = \frac{1}{\prod_{j\neq i}(x_i - x_j)}.$$

Alternative approach: general case

- Finally we have

$$l_i(x) \;=\; \prod_{j\neq i}\left(\frac{x - x_j}{x_i - x_j}\right)$$

- The polynomials $\boldsymbol{l_i(x)}$ are called Lagrange polynomials

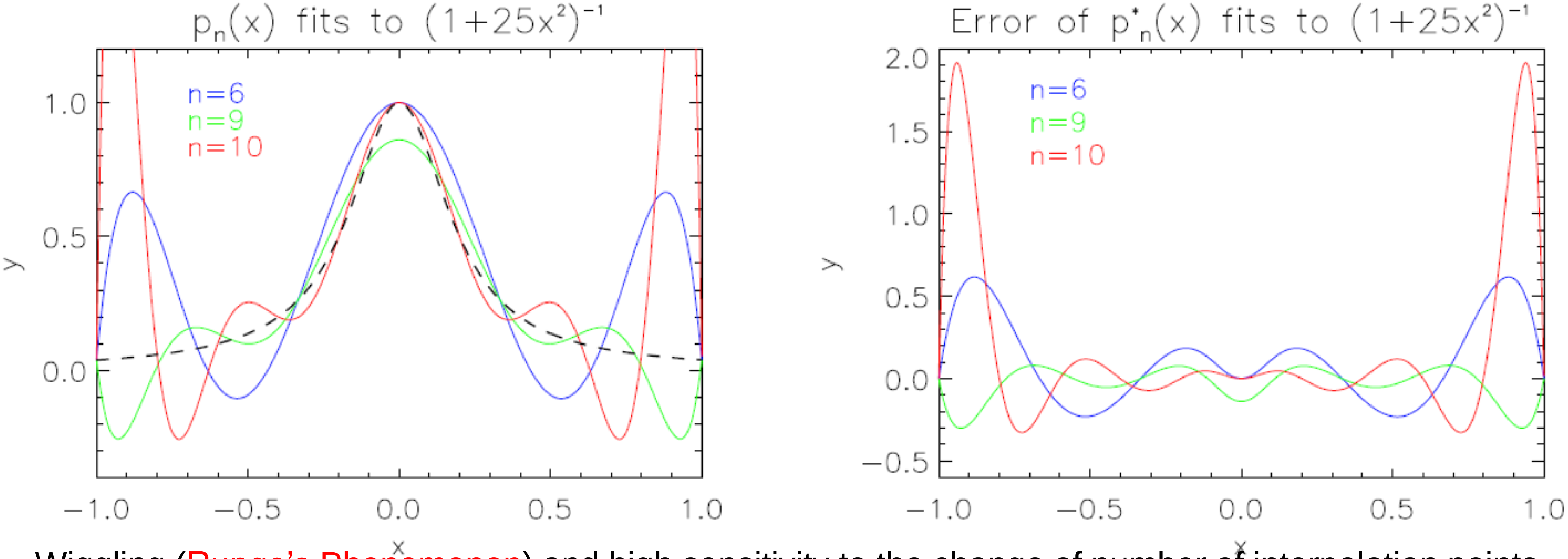
Question

- Is the solution to the interpolation problem obtained using the Lagrange polynomials different from the solution obtained using the the Vandermonde matrix (monomial basis)?

Question

- Is the solution to the interpolation problem obtained using the Lagrange polynomials different from the solution obtained using the the Vandermonde matrix (monomial basis)?
- Answer: they are the same!
  - Assume they are different. Let's denote R_n the polynomial defined by their difference. $R_n$has a degree of at most n.
  - We have $R_n(x_i) = 0$, $i = 0 \dots n$, where x_i are the distinct interpolation points. So R_n has a degree of at most n and has $n + 1$ roots ➔ $R_n = 0$.
- Of course there are many other ways of representing the same polynomial!

How good is our interpolation?



Wiggling (Runge's Phenomenon) and high sensitivity to the change of number of interpolation points. Observe the difference between $n = 9$ (10 data points) and $n = 10$ (11 data points)

Conclusion

- Polynomial interpolation is instable
- Small changes in control points can lead to very different result. x_i sequence is important.

- “Runge’s phenomenon”: Oscillating behavior
- Wiggling of the polynomial as the number of fitting points increases (even slightly).

➔ We need better basis functions for interpolation

  - For example, piecewise polynomials will work much better

Approximation

Polynomial and least squares approximation

Motivation

- Why do we need approximation:
  - Noise in the data (sample points)
  - Compact representation
  - Simpler evaluations
- Common approximating functions
  - **Polynomials**
  - Rational functions (quotient of polynomials)
  - Trigonometric functions

Why use polynomials?

- Easy to evaluate, well behaved, smooth,…
- Can be justified analytically:
  - Weierstrass’ theorem: Let f be any continuous function on a closed interval $[a, b]$, then for any ε, there exist an n and polynomial P_n s.t. $|f(x) - P_n| < \varepsilon$

  - Weierstrass only proved existence without generating the polynomials

Approximation with Bernstein Polynomials

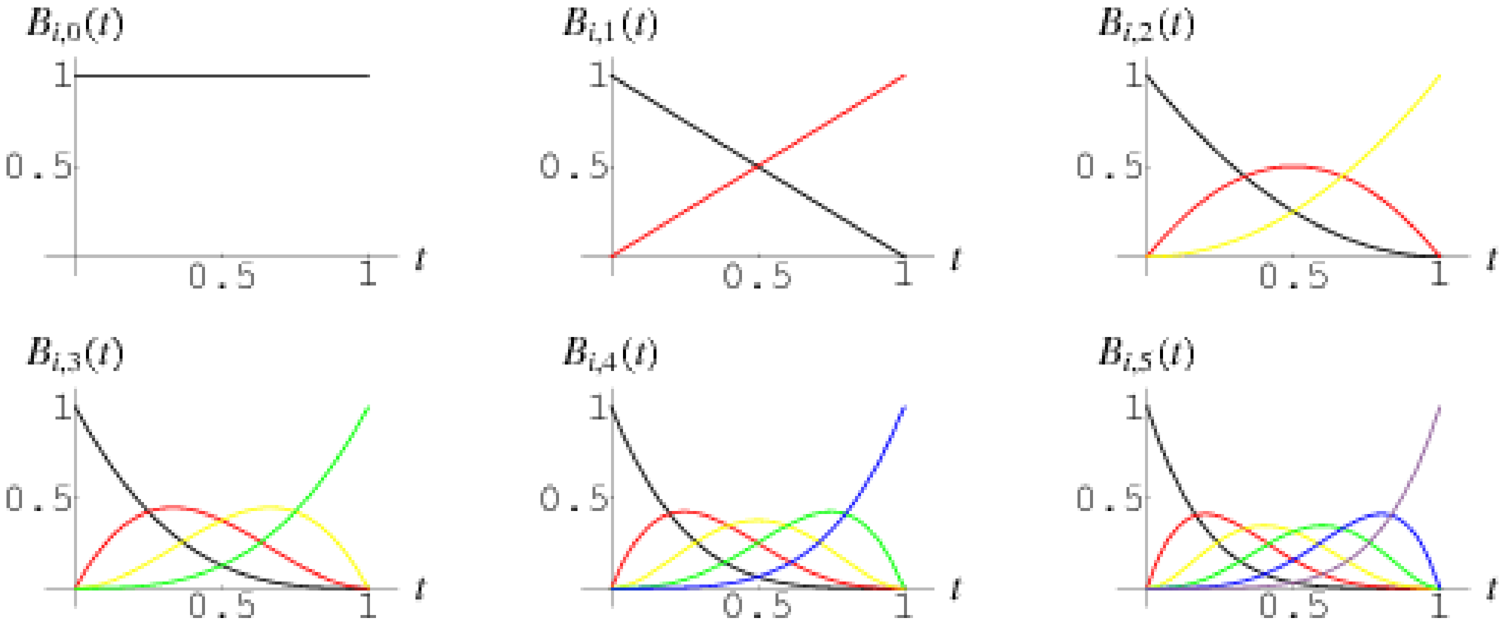
- Bernstein gave a constructive proof (Powerful!)
  - For any continuous function on **[0, 1]** and any positive integer n, we have for all x in [0, 1]

    $$|f(x) - B_n(f, x)| < \frac{9}{4}\, m_{f,n}$$

  - $m_{f,n} = \underset{\{y,y' \text{in } [0,1], |y-y'| < \frac{1}{\sqrt{n}}\}}{lower\ upper\ bound} |f(y) - f(y')|$

  - $B_n(f,x) = \sum_{j=0}^{n} f\left(x_j\right) b_{n,j}\,(x)$, where x_j are equally spaced sampling points on $[0,1]$
  - $b_{n,j} = \binom{n}{j} x^j (1-x)^{n-j}$, called Bernstein polynomials

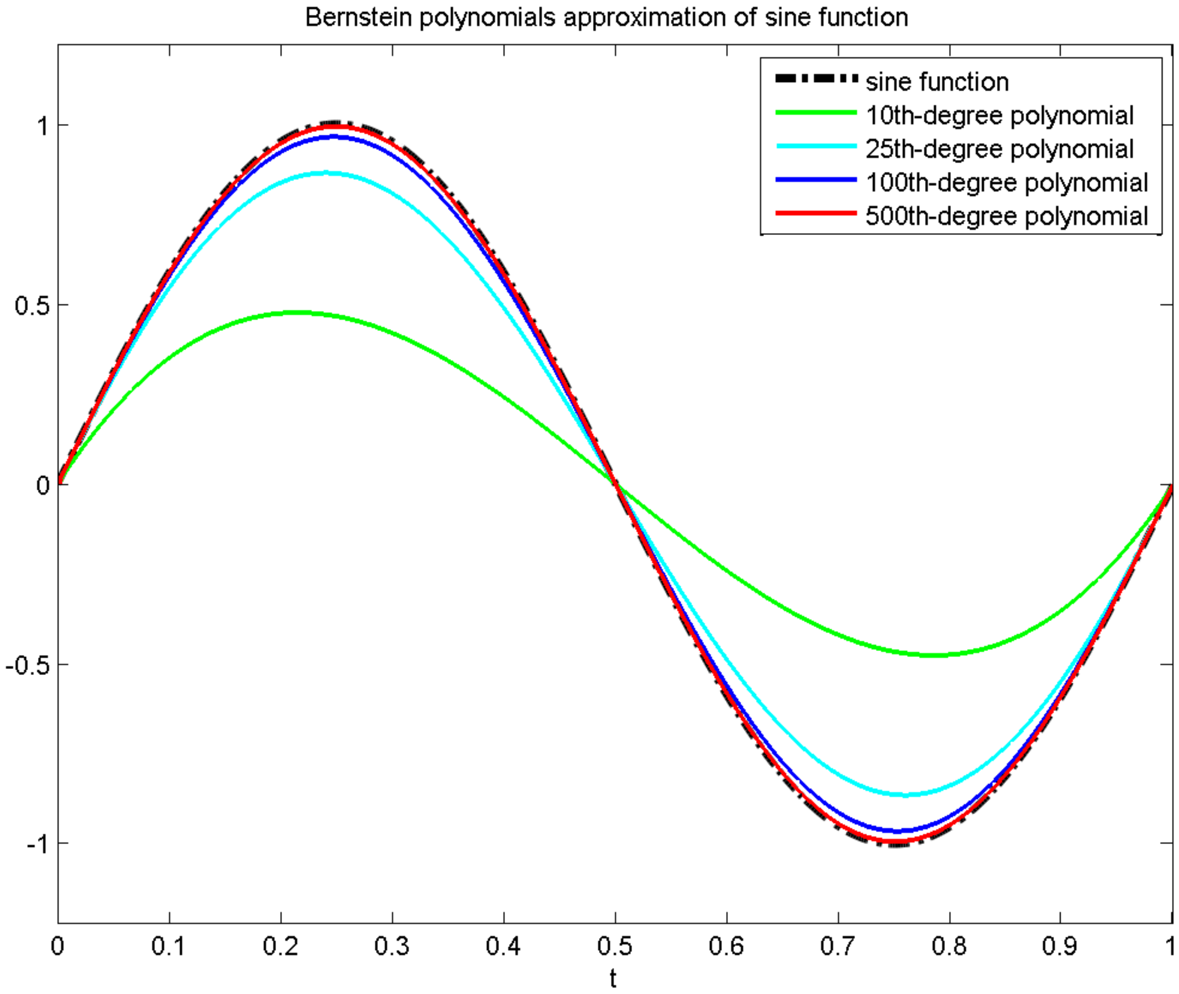
Bernstein Polynomials



$$
\begin{aligned}
&b_{0,0}(x) = 1, \\
&b_{0,1}(x) = 1 - x, \quad b_{1,1}(x) = x \\
&b_{0,2}(x) = (1-x)^2, \quad b_{1,2}(x) = 2x(1-x), \quad b_{2,2}(x) = x^2 \\
&b_{0,3}(x) = (1-x)^3, \quad b_{1,3}(x) = 3x(1-x)^2, \quad b_{2,3}(x) = 3x^2(1-x), \quad b_{3,3}(x) = x^3 \\
&b_{0,4}(x) = (1-x)^4, \quad b_{1,4}(x) = 4x(1-x)^3, \quad b_{2,4}(x) = 6x^2(1-x)^2, \quad b_{3,4}(x) = 4x^3(1-x), \quad b_{4,4}(x) = x^4
\end{aligned}
$$

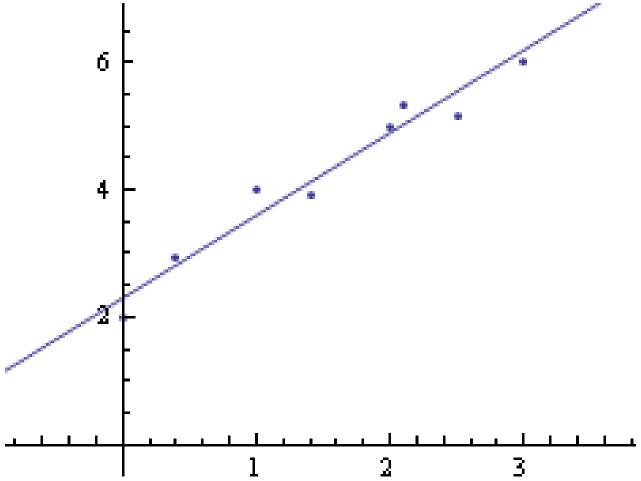
Approximation with Bernstein polynomials

- Example: approximation with Bernstein polynomials
  - Produces excellent approximation but requires a high order
  - Expensive evaluations
  - Can be prone to errors



Least-squares approximation

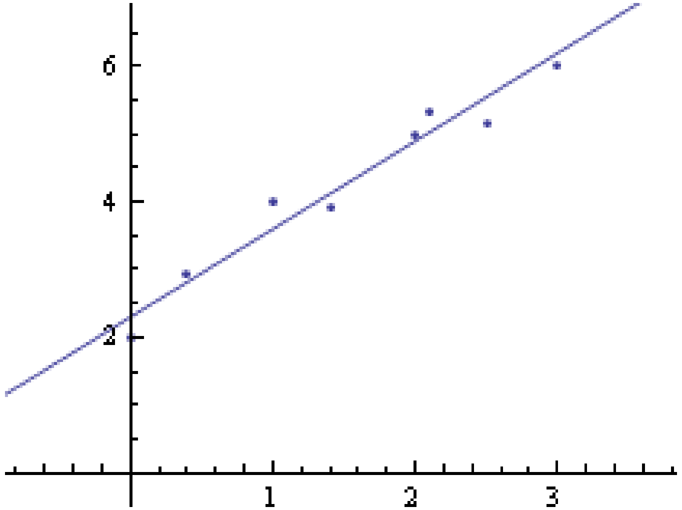
- Approximation Problem
  - Given a linearly independent set $B = \{b_1,...,b_n\}$ of continuous functions and nodes $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$ with $m > n$.
  - What function $f \in span(B)$ *best approximates* the nodes?
  - Example: Best approximating linear function for a set of nodes
  - How do we define "*best approximating*"?



What is meant by *best approximating*?

- Least-Squares Approximation

$$\underset{f\in Span(B)}{\operatorname{argmin}} \sum_{j=1}^{m} \left(f(x_j)-y_j\right)^2$$



$$\sum_{j=1}^{m}\left(f(x_j)-y_j\right)^2$$

$$= \sum_{j=1}^{m}\left(\sum_{i=1}^{n}\lambda_i b_i(x_j)-y_j\right)^2$$

$$= (M\lambda - y)^T(M\lambda - y)$$

$$= \lambda^T M^T M\lambda - y^T M\lambda - \lambda^T M^T y + y^T y$$

$$= \lambda^T M^T M\lambda - 2y^T M\lambda + y^T y$$

$$M = \begin{pmatrix} b_1(\mathbf{x}_1) & \Lambda & b_n(\mathbf{x}_1) \\ \mathrm{M} & & \mathrm{M} \\ b_1(\mathbf{x}_m) & \Lambda & b_n(\mathbf{x}_m) \end{pmatrix}$$

Solving the Problem

- This is a quadratic polynomial in $\boldsymbol{\lambda}$

$$\lambda^T M^T M\lambda - 2y^T M\lambda + y^T y$$

- Normal equation
  - The minimizer satisfies

$$M^T M\lambda = M^T y$$

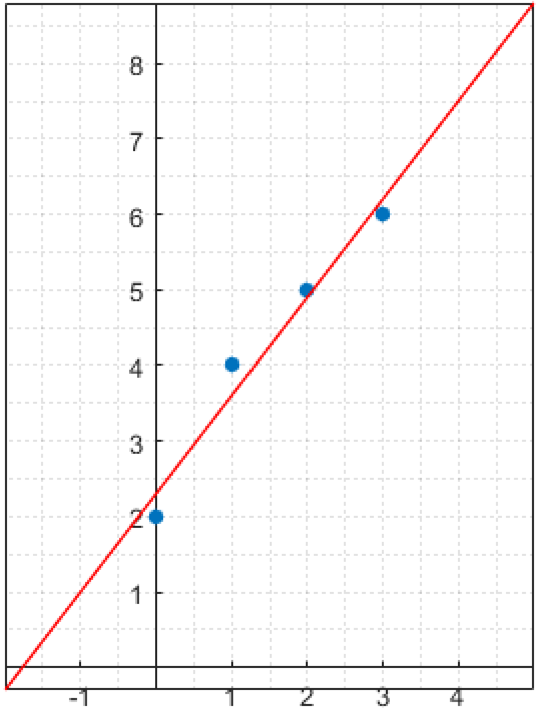
- Reminder
  - Minimize quadratic objective function $\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathrm{T}}\mathbf{x} + c$
  - Necessary and sufficient condition $2\mathbf{A}\mathbf{x} = -\mathbf{b}$

Example: linear approximation

```
%input data
x=[0 1 2 3]';
y=[2 4 5 6]';

%Setup the matrix
M=[x.^0 x.^1];

%solve the least squares
c=M'*M\(M'*y);
```

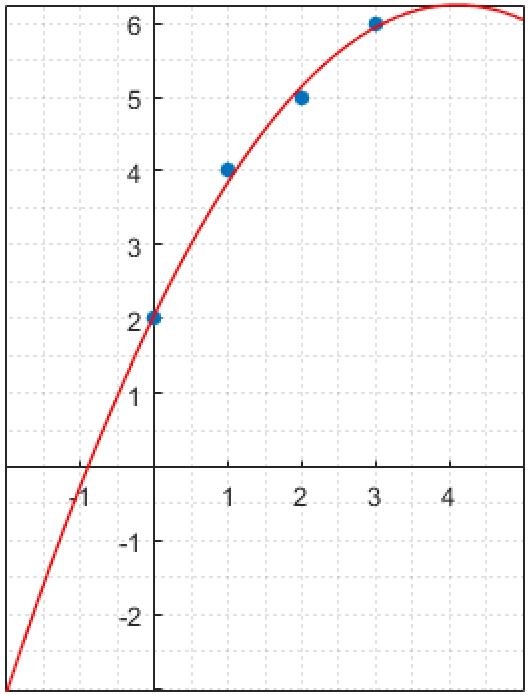


Example: Quadratic approximation

```
%input data
x=[0 1 2 3]';
y=[2 4 5 6]';

%Setup the matrix
M=[x.^0 x.^1 x.^2];

%solve the least squares
c=M'*M\(M'*y);
```



Questions?