

# **Lec 17: Integration and Differentiation**

**15-369/669/769: Numerical Computing**

**Instructor: Minchen Li**

# Motivation

## Numerical Integration and Differentiation

- Interpolation techniques focused on predicting values of  $f(\mathbf{x})$  from samples.
- Many practical problems instead require computing **derived quantities** – integrals or derivatives of  $f$ .
- Applications span physics, statistics, optimization, and signal processing.
- **Example: Error Function.** The error function, important in probability and diffusion models, is defined by

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

Evaluating  $\operatorname{erf}(x)$  requires numerical integration since it lacks closed-form solutions. Approximation accuracy depends on sampling and discretization strategies.

# Motivation

## More Examples

- **Integral Equation Example:** In computational electrodynamics, many problems take the form:

$$f(\mathbf{x}) = \int_{\mathbb{R}^n} K(\mathbf{x}, \mathbf{y}) \phi(\mathbf{y}) d\mathbf{y},$$

where  $K$  is a kernel function. Solutions require discretizing  $\phi$  and approximating the integral numerically.

- **Image Processing:** Convolution integrals

$$(I * g)(x, y) = \iint_{\mathbb{R}^2} I(u, v) g(x - u, y - v) du dv.$$

- **Optimization:** Derivative and Hessian estimation (e.g., BFGS, Newton's method).

# Table of Content

- Quadrature
- Differentiation

# Table of Content

- Quadrature
- Differentiation

# Quadrature

## The Integration Problem

- The general goal of **quadrature** is to approximate:

$$\int_a^b f(x) dx.$$

- Depending on context:
  - Endpoints  $a, b$  may be fixed or vary.
  - $f(x)$  may be known analytically or only via samples  $(x_i, f_i)$ .
- Design goal: find an efficient approximation scheme  $Q[f]$  that minimizes error given limited samples.

# Quadrature

## Interpolator-Based Quadrature

- Express  $f(x)$  in a basis  $\{\phi_i(x)\}$ :

$$f(x) = \sum_i a_i \phi_i(x).$$

- Integrate term-by-term:

$$\int_a^b f(x) dx = \sum_i a_i \int_a^b \phi_i(x) dx = \sum_i c_i a_i,$$

where  $c_i := \int_a^b \phi_i(x) dx$ .

- Thus, integration of  $f$  is a **weighted sum** of coefficients  $a_i$ .

- **Example: Monomial Basis.** If  $f(x) = \sum_k a_k x^k$ , then  $\int_0^1 f(x) dx = \sum_k \frac{a_k}{k+1}$ .

# Quadrature

## General Quadrature Rules

- Quadrature approximations typically take the form:

$$Q[f] = \sum_i w_i f(x_i),$$

where  $x_i$  are sample points and  $w_i$  are weights.

- Choices of  $\{x_i, w_i\}$  define a **quadrature rule**.
- As sampling becomes denser,  $Q[f]$  approaches the true integral.
- **Riemann Integral:**

$$\int_a^b f(x) dx = \lim_{\Delta x_k \rightarrow 0} \sum_k f(\tilde{x}_k) (x_{k+1} - x_k).$$

A quadrature rule generalizes this discrete summation using strategically chosen  $x_i$  and  $w_i$ .

# Quadrature

## Method of Undetermined Coefficients

- Fix  $n$  sample points  $x_1, \dots, x_n$  and determine  $w_i$  such that

$$\sum_i w_i f(x_i) \approx \int_a^b f(x) dx$$

holds exactly for chosen functions  $f_1, \dots, f_n$ .

- This leads to a linear system:  $\int_a^b f_j(x) dx = \sum_i w_i f_j(x_i), \quad j = 1, \dots, n.$
- **Polynomial Basis Example:** Choosing  $f_k(x) = x^{k-1}$  ensures exactness for polynomials up to degree  $n - 1$ :

$$\int_a^b x^k dx = \frac{b^{k+1} - a^{k+1}}{k+1}.$$

# Quadrature

## Vandermonde System for Quadrature Weights

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} b - a \\ \frac{1}{2}(b^2 - a^2) \\ \frac{1}{3}(b^3 - a^3) \\ \vdots \\ \frac{1}{n}(b^n - a^n) \end{pmatrix}.$$

- This linear system (a transposed Vandermonde matrix) yields the weights  $w_i$ .
- Basis for many classical quadrature schemes (e.g., Newton–Cotes rules).

# Quadrature

## Newton–Cotes Quadrature: Setup

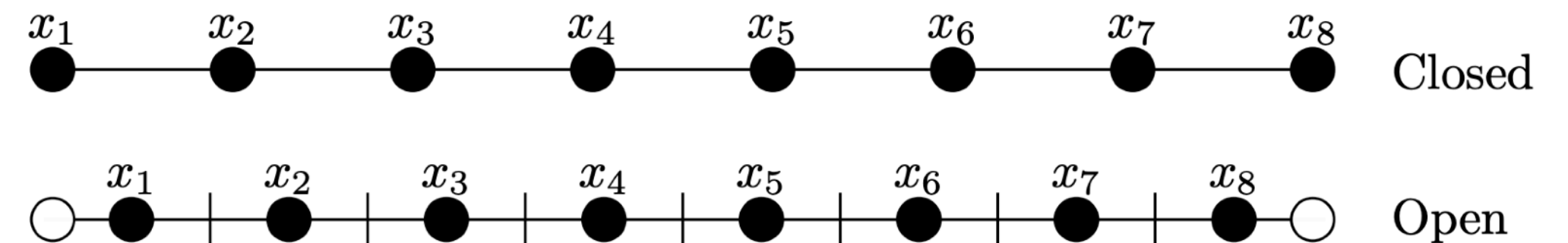
- Newton–Cotes quadrature integrates the interpolating polynomial through evenly spaced samples  $x_i \in [a, b]$ .
- Node placement:

$$\text{Closed: } x_k = a + \frac{(k-1)(b-a)}{n-1}, \quad \text{Open: } x_k = a + \frac{k(b-a)}{n+1}.$$

- General form:

$$\int_a^b f(x) dx \approx \sum_{i=1}^n w_i f(x_i),$$

where  $w_i = \int_a^b \ell_i(x) dx$  from the Lagrange basis  $\ell_i$ .



# Quadrature

## Closed Newton–Cotes: Trapezoid and Simpson

- Trapezoid rule ( $n = 2$ ):

$$\int_a^b f(x) dx \approx (b - a) \frac{f(a) + f(b)}{2}.$$

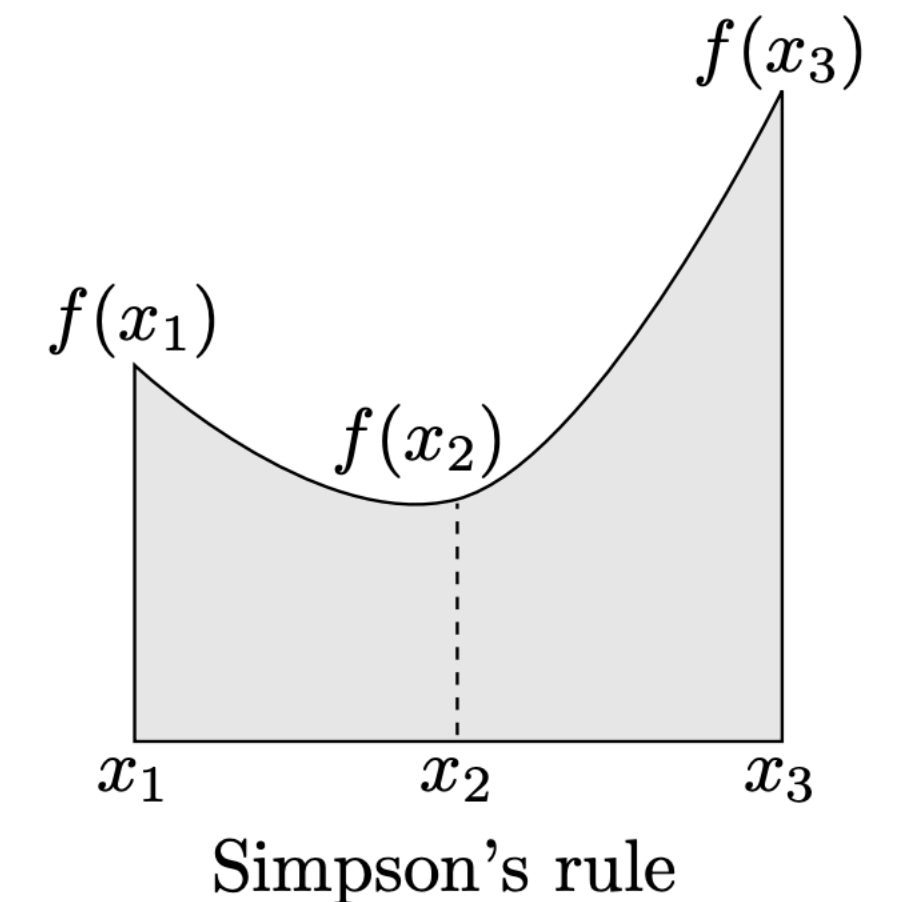
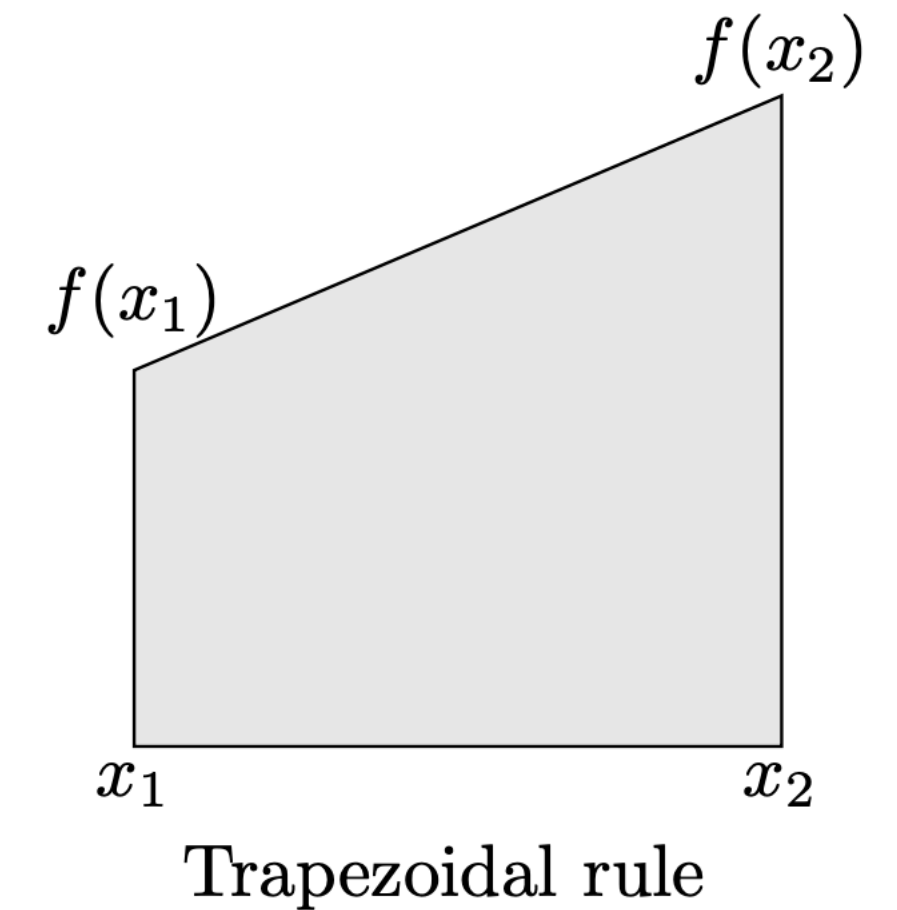
Derived from linear interpolation between  $(a, f(a))$  and  $(b, f(b))$ .

- Simpson's rule ( $n = 3$ ):

$$x_1 = a, \quad x_2 = \frac{a+b}{2}, \quad x_3 = b.$$

Integrate the quadratic interpolant:

$$\int_a^b f(x) dx \approx \frac{b-a}{6} (f(a) + 4f(\frac{a+b}{2}) + f(b)).$$



- Each rule integrates the polynomial interpolant  $\sum f(x_i) \ell_i(x)$  exactly up to degree  $n - 1$ .

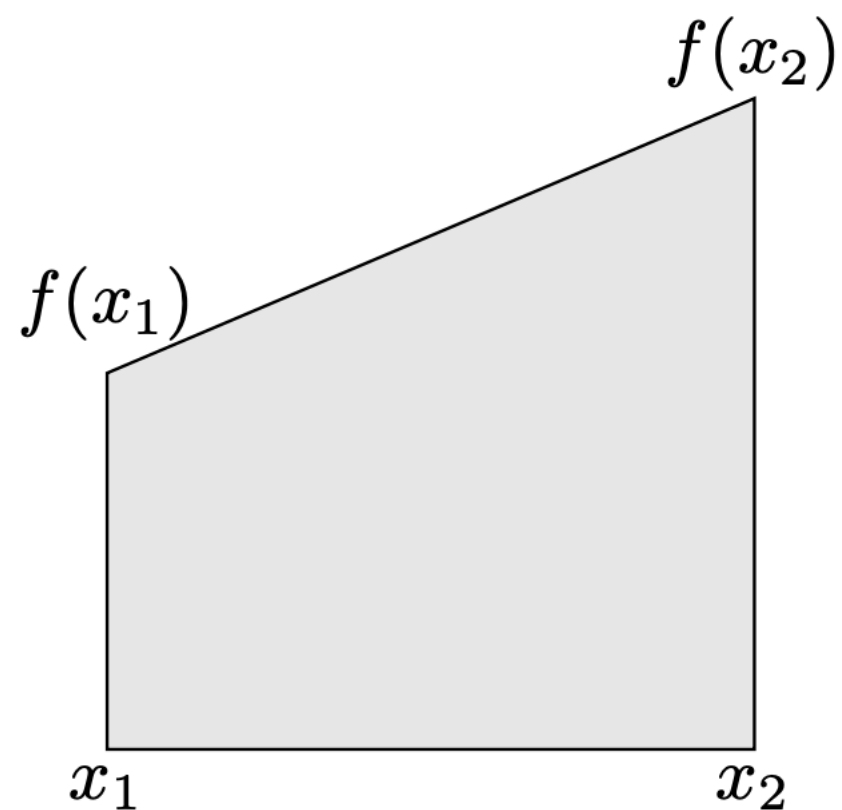
# Quadrature

## Open Newton–Cotes: Midpoint and Higher-Order Rules

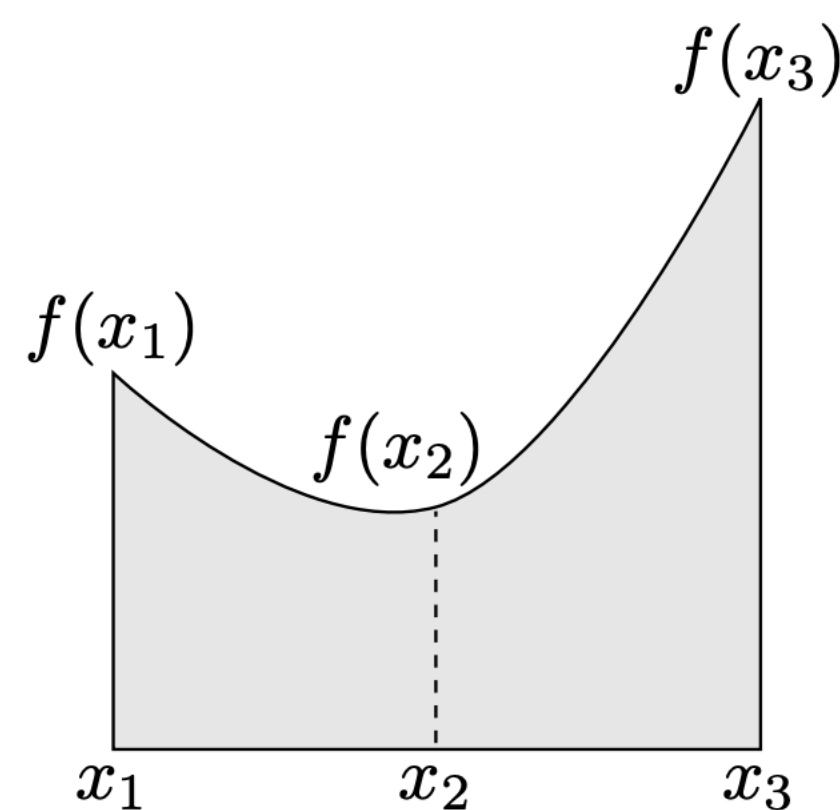
- Midpoint rule ( $n = 1$ ):

$$x_1 = \frac{a+b}{2}, \quad \int_a^b f(x) dx \approx (b-a) f\left(\frac{a+b}{2}\right).$$

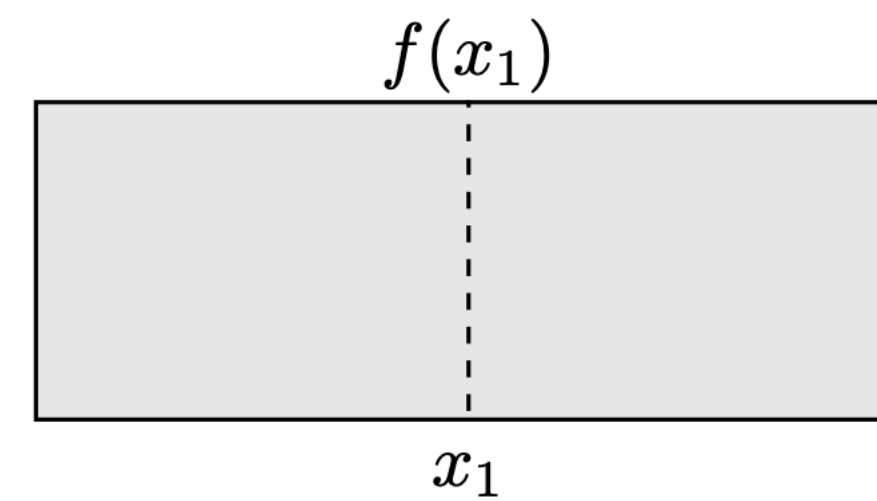
- This is the simplest open Newton–Cotes formula, using one interior node.
- Higher- $n$  open rules follow the same pattern but omit endpoints, using evenly spaced interior points.



Trapezoidal rule



Simpson's rule



Midpoint rule

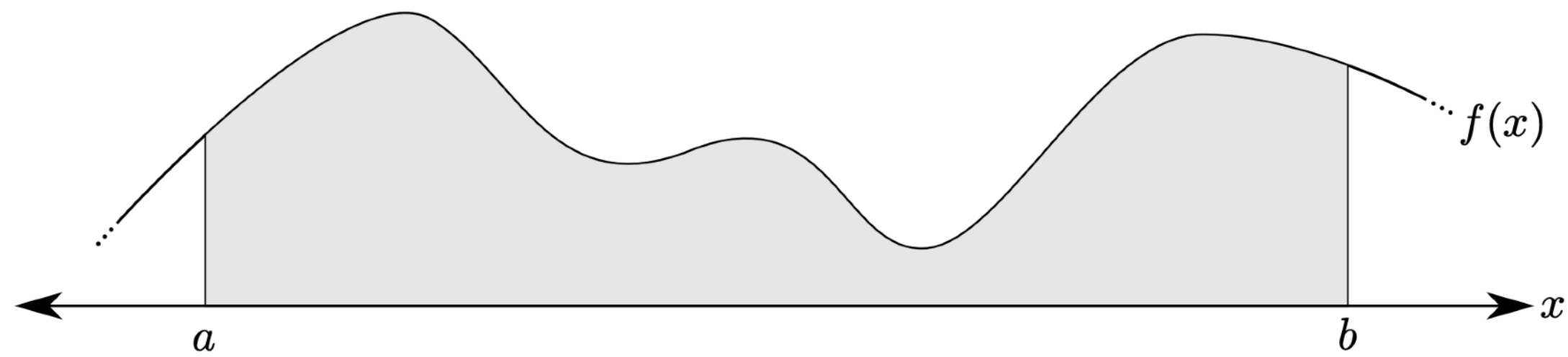
# Quadrature

## Composite Newton–Cotes Rules

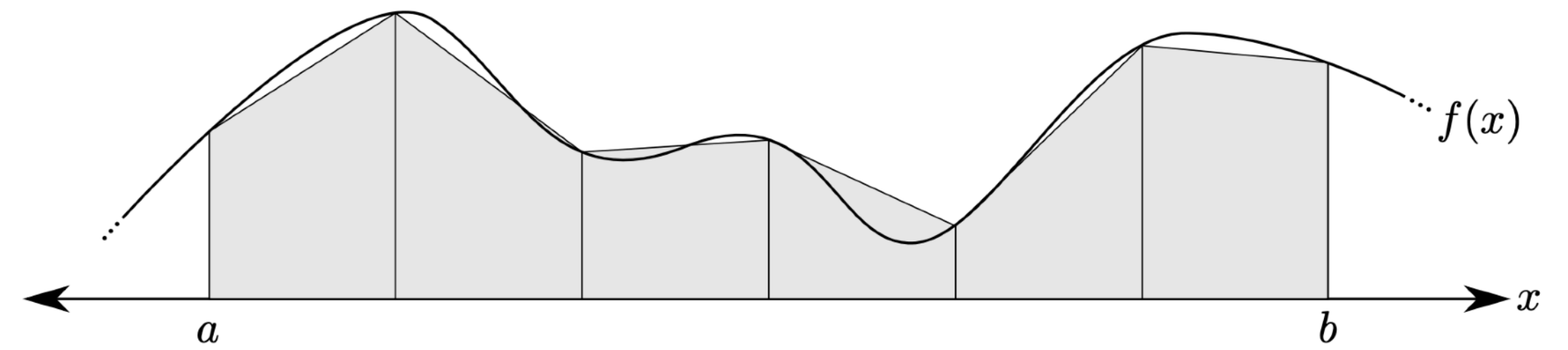
- Divide  $[a, b]$  into  $k$  subintervals of width  $\Delta x = \frac{b-a}{k}$ ,  $x_i = a + (i-1)\Delta x$ .
- Apply a low-order rule on each subinterval, summing the results.
- **Composite midpoint:**  $\int_a^b f(x) dx \approx \sum_{i=1}^k f\left(\frac{x_{i+1}+x_i}{2}\right)\Delta x$ .
- **Composite trapezoid:**  $\int_a^b f(x) dx \approx \Delta x \left[ \frac{1}{2}f(a) + \sum_{i=2}^k f(x_i) + \frac{1}{2}f(b) \right]$ .
- **Composite Simpson:**  $\int_a^b f(x) dx \approx \frac{\Delta x}{6} \left[ f(a) + 4f(x_2) + 2f(x_3) + \cdots + 4f(x_k) + f(b) \right]$ .

# Quadrature

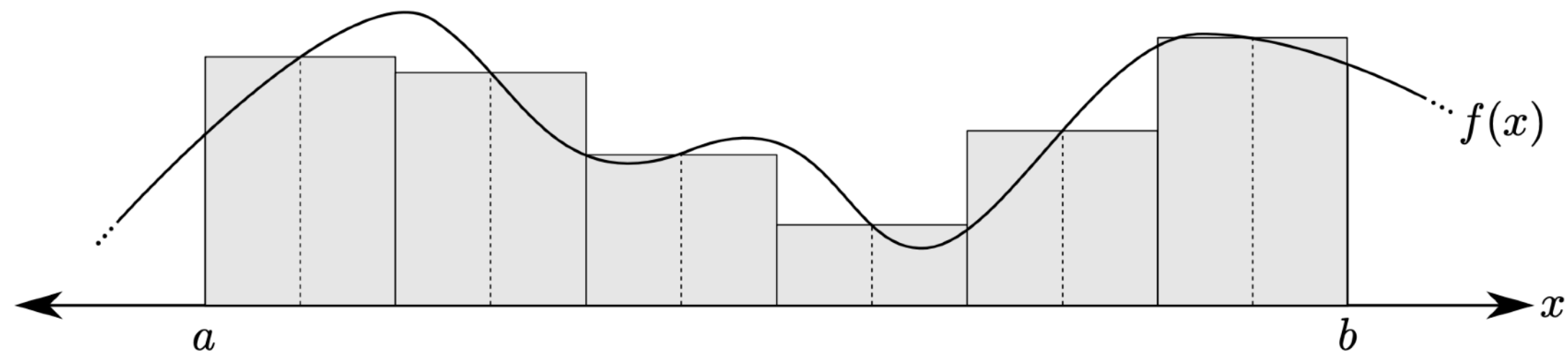
## Composite Newton–Cotes Rules: Example



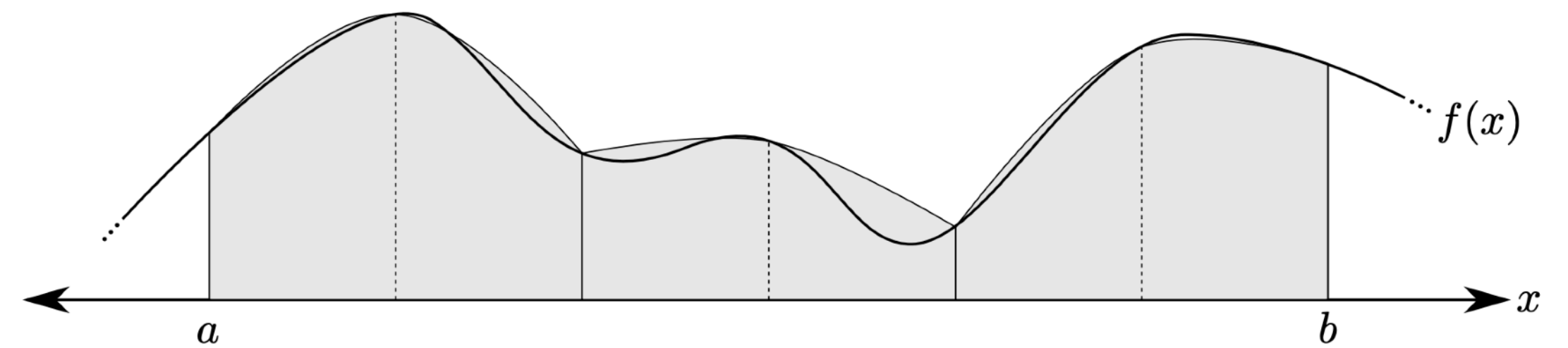
Actual integral



Composite trapezoidal rule (7 samples)



Composite midpoint rule (6 samples)



Composite Simpson's rule (7 samples)

# Quadrature

## Accuracy via Taylor Expansion

- Expand  $f$  about midpoint  $c = \frac{a+b}{2}$ :

$$f(x) = f(c) + f'(c)(x - c) + \frac{1}{2}f''(c)(x - c)^2 + \frac{1}{6}f^{(3)}(c)(x - c)^3 + \dots$$

- **Midpoint rule:**

$$\int_a^b f(x) dx = (b - a)f(c) + \frac{1}{24}f''(c)(b - a)^3 + \dots \Rightarrow O(\Delta x^3).$$

- **Trapezoid rule:**

$$f(a) + f(b) = 2f(c) + \frac{1}{4}f''(c)(b - a)^2 + \dots$$

$$(b - a) \frac{f(a) + f(b)}{2} = (b - a)f(c) + \frac{1}{8}f''(c)(b - a)^3 + \dots \Rightarrow O(\Delta x^3).$$

- Both midpoint and trapezoid rules yield the same error order  $O(\Delta x^3)$ .

# Quadrature

## Composite Accuracy Orders

- Single-interval accuracies:

Midpoint:  $O(\Delta x^3)$ , Trapezoid:  $O(\Delta x^3)$ , Simpson:  $O(\Delta x^5)$ .

- For  $k = (b - a) / \Delta x$  panels, global errors are one order lower:

Composite midpoint:  $O(\Delta x^2)$ ,

Composite trapezoid:  $O(\Delta x^2)$ ,

Composite Simpson:  $O(\Delta x^4)$ .

- Midpoint often slightly more accurate due to smaller error constant.

# Quadrature

## Gaussian Quadrature: Concept

- Choose both  $\{x_i\}$  and  $\{w_i\}$  to maximize polynomial exactness.
- Require exactness for  $2n$  test functions:

$$\int_a^b f_k(x) dx = \sum_{i=1}^n w_i f_k(x_i), \quad k = 1, \dots, 2n.$$

- For suitable  $f_k$  (e.g., monomials), Gaussian quadrature integrates all polynomials up to degree  $2n - 1$  exactly.
- This achieves the highest possible order for  $n$  nodes.

# Quadrature

## Gaussian Quadrature: Example ( $n = 2$ on $[-1, 1]$ )

- Enforce exactness for  $1, x, x^2, x^3$ :

$$w_1 + w_2 = 2,$$

$$w_1 x_1 + w_2 x_2 = 0,$$

$$w_1 x_1^2 + w_2 x_2^2 = \frac{2}{3},$$

$$w_1 x_1^3 + w_2 x_2^3 = 0.$$

- Solution:  $x_{1,2} = \pm \frac{1}{\sqrt{3}}$ ,  $w_1 = w_2 = 1$ . This is the classical two-point Gauss–Legendre rule.

- Properties:

- Integrates all polynomials up to degree 3 exactly.
- Not progressive (new nodes differ for each  $n$ ).
- Kronrod rules extend Gaussian quadrature by adding points for adaptive accuracy.

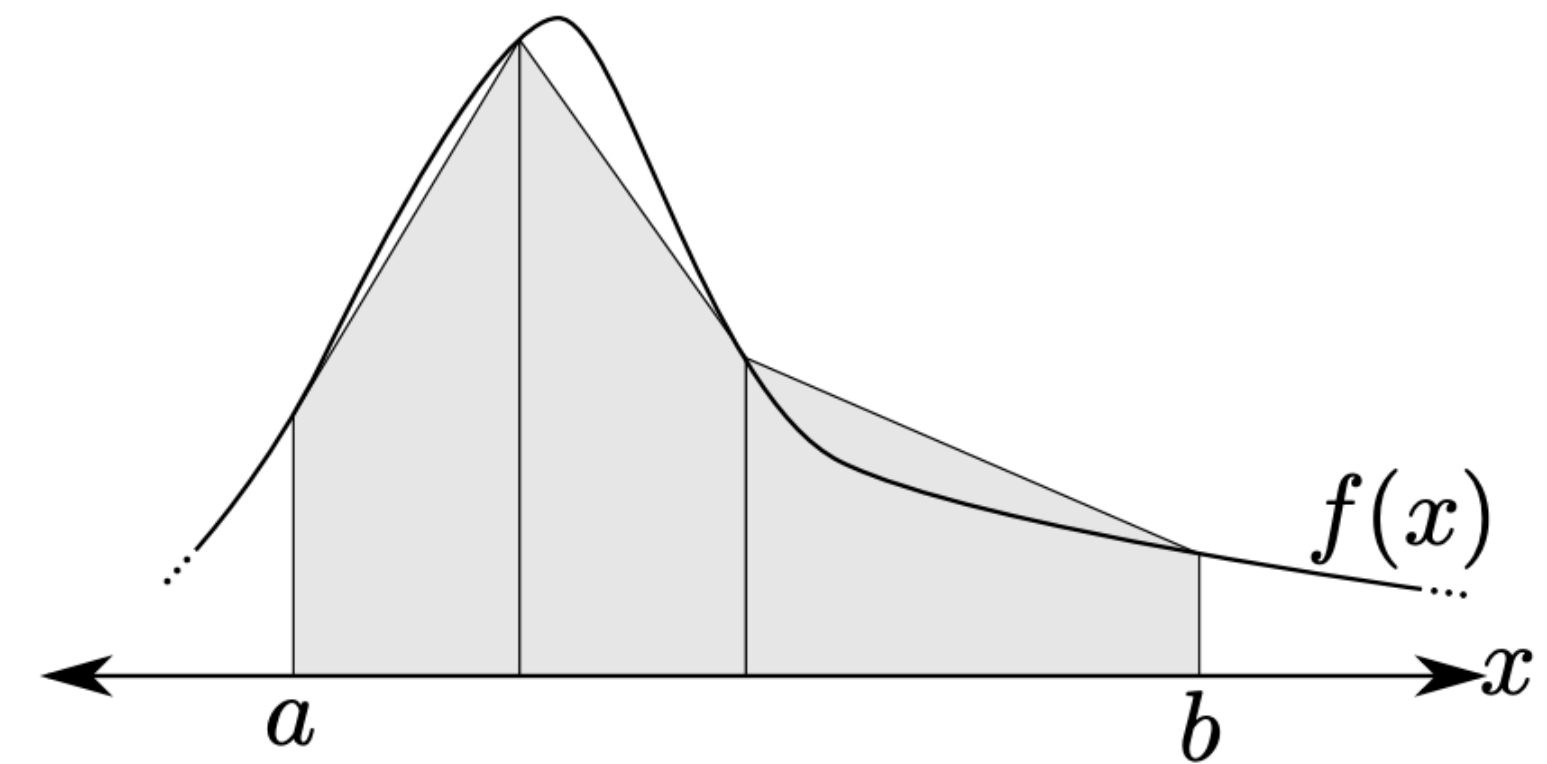
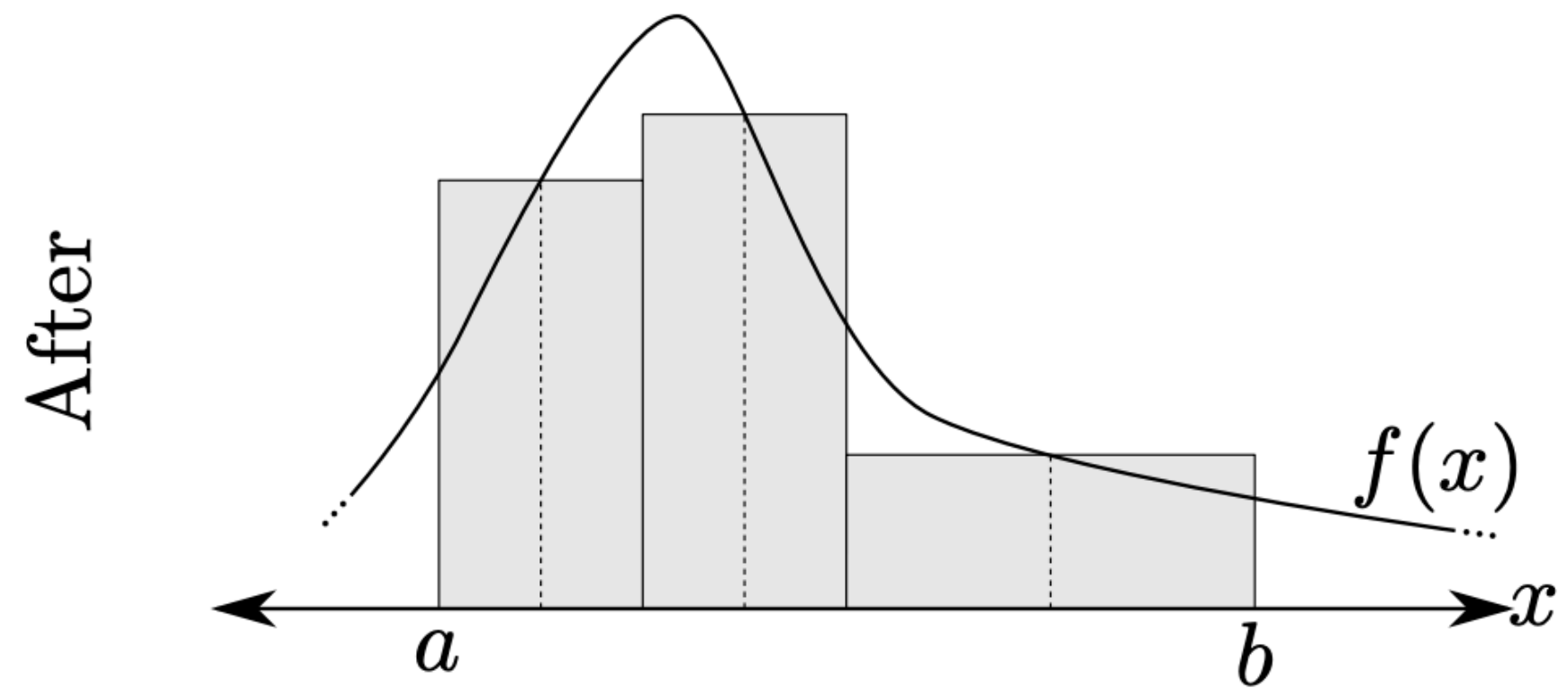
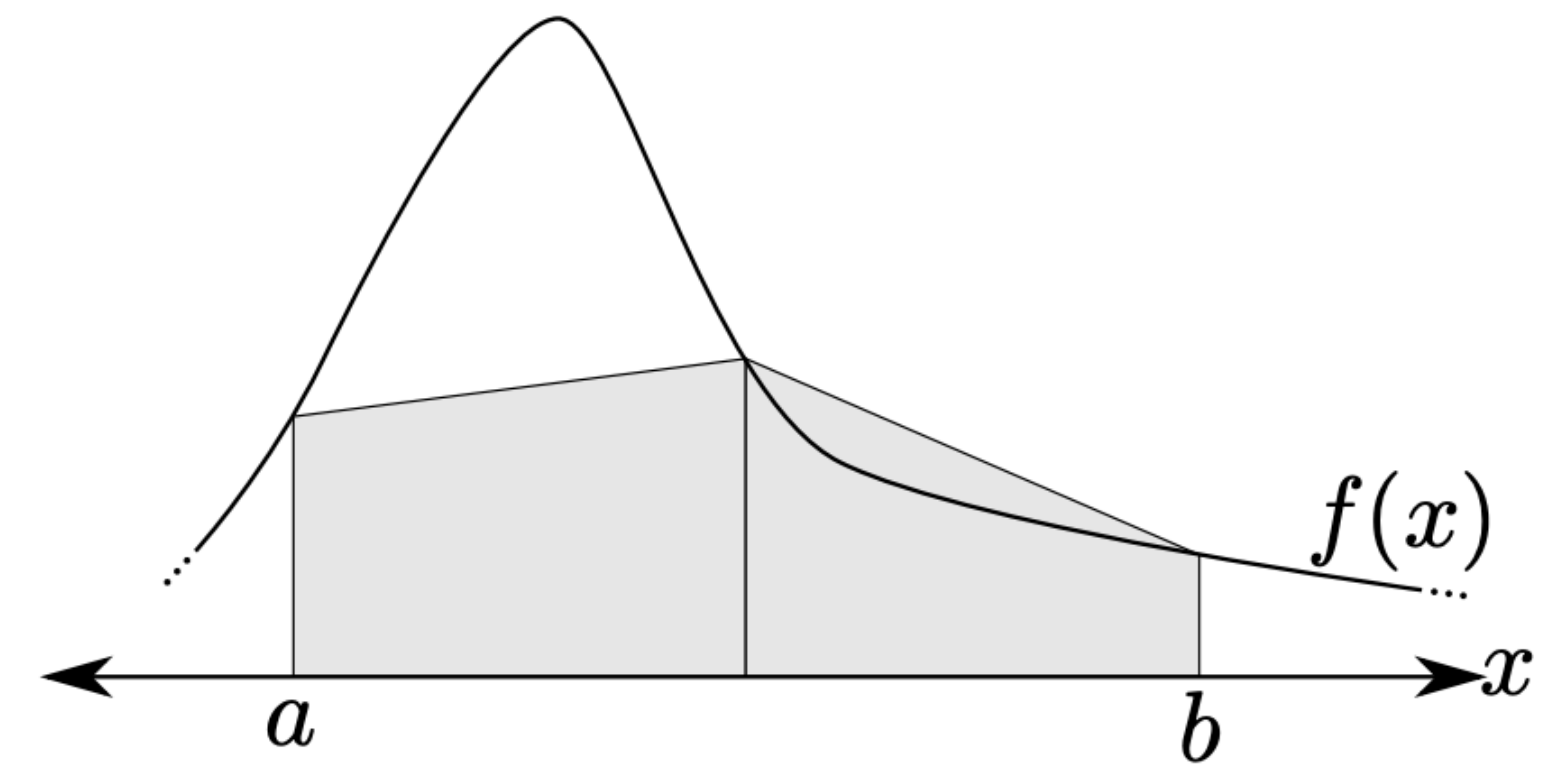
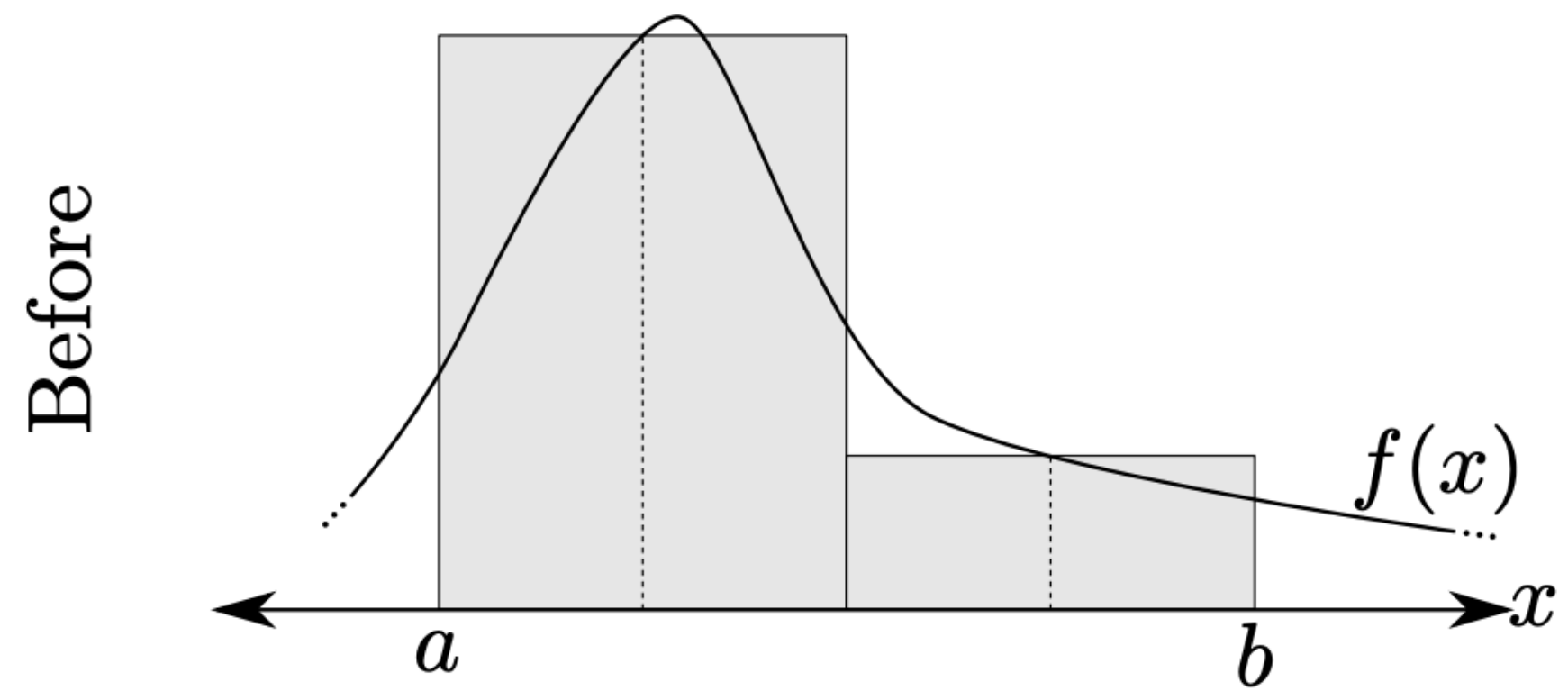
# Quadrature

## Adaptive Quadrature

- In previous quadrature schemes, nodes  $x_i$  were predetermined (e.g., equally spaced or optimized).
- **Observation:** The function values  $f(x_i)$  contain information about where the integrand changes rapidly or is undersampled.
- **Idea:** Dynamically refines the sampling density based on the integrand's local behavior.
  - Estimate the integral over a subinterval using two quadrature rules (e.g., trapezoid and midpoint).
  - Compare their results — if they differ by more than a tolerance, subdivide that subinterval.
  - Continue recursively until all subintervals meet the tolerance criterion.
- **Advantages:** Allocates samples efficiently where  $f(x)$  varies most.
- **Caveat:** Too-deep recursion or noisy data may produce instability or excessive refinement.

# Quadrature

## Adaptive Quadrature: Example



Midpoint rule

Trapezoidal rule

# Quadrature

## Multiple Integrals and Dimensionality

- For  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , we often seek  $\int_{\Omega} f(\mathbf{x}) d\mathbf{x}$ ,  $\Omega \subseteq \mathbb{R}^n$ .
- Example ( $n = 2$ ): rectangular region integration:  $\int_a^b \int_c^d f(x, y) dx dy$ .
- **Curse of dimensionality:** Required sample count grows exponentially with  $n$ .
- **Strategy 1: Iterated integral.**
  - Approximate  $\int_c^d f(x, y) dx$  using 1D quadrature for each  $y$ .
  - Then integrate these results over  $y$  with another quadrature rule.
- Both inner and outer integrations induce numerical errors, and more samples are often required.

# Quadrature

## Monte Carlo Integration

- When  $n$  is large, deterministic subdivision is impractical.
- **Monte Carlo integration:**
  - Generate  $k$  random points  $\mathbf{x}_i \in \Omega$  uniformly.
  - Approximate:  $\int_{\Omega} f(\mathbf{x}) d\mathbf{x} \approx \frac{|\Omega|}{k} \sum_{i=1}^k f(\mathbf{x}_i)$ .
- **Convergence rate:**  $O(1/\sqrt{k})$ , independent of dimension  $n$ .
- **Advantages:** Scales well to high dimensions; Easy to implement and extend.
- **Limitations:**
  - Convergence is slow compared to deterministic quadrature in low dimensions.
  - Requires random number generation and bias correction for restricted domains  $\Omega \subset [a, b]^n$ .

# Quadrature

## Sparse and Smolyak Grids

- **Problem:** Uniform sampling in  $[a, b]^n$  is inefficient when  $\Omega$  occupies a small portion of the hypercube.
- **Solution:** More sophisticated adaptive sampling:
  - **Sparse grid methods:** Selectively refine certain directions to reduce dimensionality effects.
  - **Smolyak construction:** Combines tensor-product quadratures of different resolutions to achieve efficiency without full randomization.
- These methods bridge the gap between deterministic integration and randomized Monte Carlo methods.

# Quadrature

## Conditioning of Quadrature Rules

- **Stability:** Sensitivity of a quadrature rule to perturbations in  $f$ .
- Quadrature operator:  $Q[f] = \sum_{i=1}^n w_i f(x_i)$ .
- For a perturbed function  $\hat{f}$ , define  $\|f - \hat{f}\|_\infty = \max_{x \in [a,b]} |f(x) - \hat{f}(x)|$ .
- Then,  $\frac{|Q[f] - Q[\hat{f}]|}{\|f - \hat{f}\|_\infty} = \frac{|\sum_i w_i (f(x_i) - \hat{f}(x_i))|}{\|f - \hat{f}\|_\infty} \leq \sum_i |w_i| = \|\mathbf{w}\|_1$ .
- **Implication:** Rules with small  $\|\mathbf{w}\|_1$  are more stable.
- **Trade-off:** Increasing polynomial degree improves accuracy but worsens conditioning.
- **Practical rule:** Use low-degree (composite) Newton–Cotes or Gaussian quadrature for stability.

# Table of Content

- Quadrature
- Differentiation

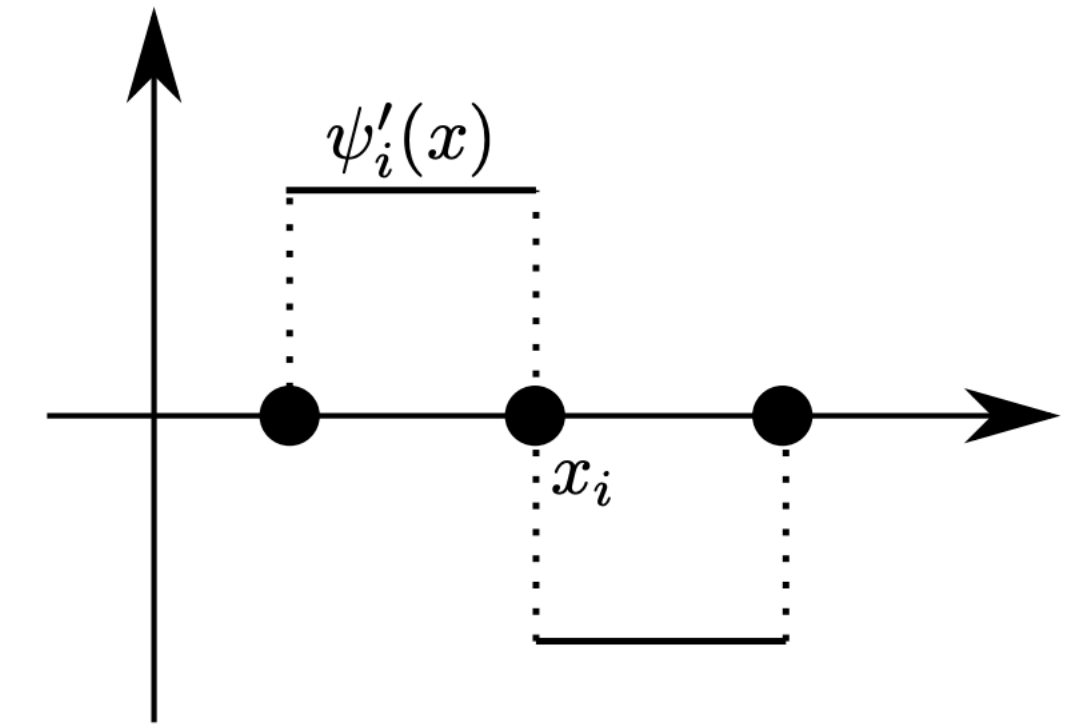
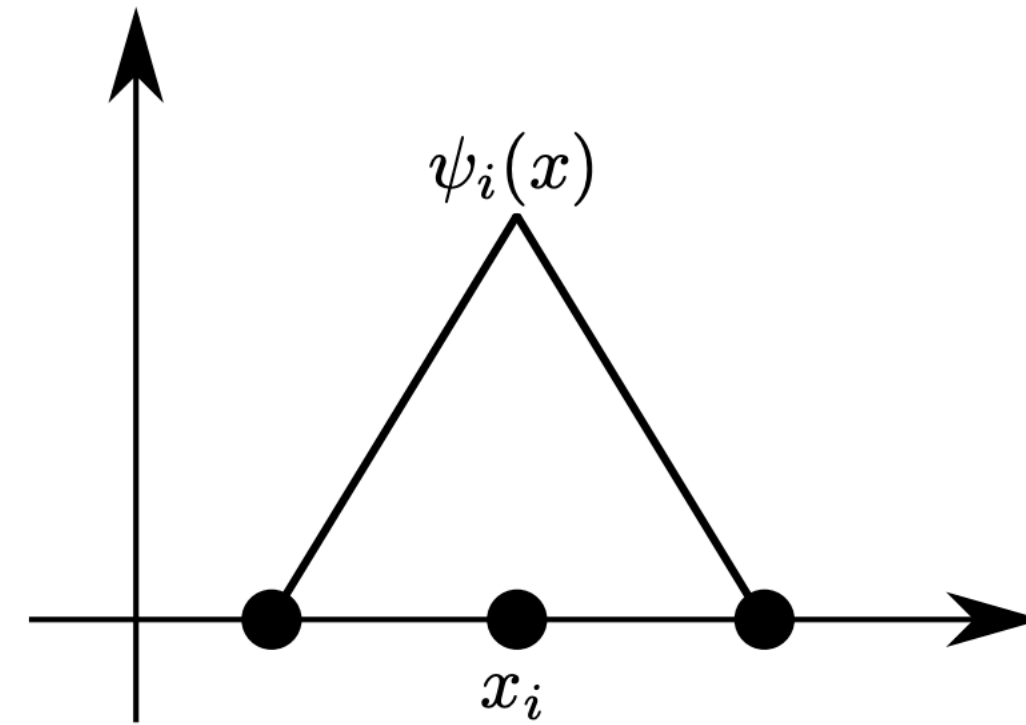
# Differentiation

## Numerical Differentiation: Overview

- **Integration** is stable — averaging reduces noise.
- **Differentiation**, however, amplifies high-frequency noise and is thus numerically sensitive.
- From a Fourier perspective:  $\int f(x) dx$  filters out high frequencies, while differentiation  $f'(x)$  increases high-frequency components.
- Hence, conditioning and stability are key issues.
- Despite this, derivative approximations are straightforward and useful in practice:
  - Used in optimization (e.g., secant rule, Broyden's method).
  - Stable when  $f$  is smooth and well-sampled.

# Differentiation

## Differentiating Basis Functions



- For  $f(x) = \sum_i a_i \phi_i(x)$ , differentiation is linear:

$$f'(x) = \sum_i a_i \phi'_i(x).$$

- $\{\phi'_i\}$  forms a basis for derivatives.
- Example interpretations:
  - Piecewise linear  $\Rightarrow$  piecewise constant derivative.
  - Polynomial  $\Rightarrow$  lower-degree polynomial derivative.
- This structure connects interpolation and discretization of differential equations.

# Differentiation

## Finite Difference Approximation

- Definition:  $f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$ .

- For finite  $h$ ,  $f'(x) \approx \frac{f(x+h) - f(x)}{h}$ .

- Using Taylor expansion:

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \dots,$$

$$\Rightarrow f'(x) = \frac{f(x+h) - f(x)}{h} + O(h).$$

- **Forward difference:**  $f'(x) \approx \frac{f(x+h) - f(x)}{h}, \quad O(h).$

- **Backward difference:**  $f'(x) \approx \frac{f(x) - f(x-h)}{h}, \quad O(h).$

# Differentiation

## Centered Difference and Higher Derivatives

- Combining forward and backward gives:  $f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$ .

- Centered difference has **quadratic convergence** ( $O(h^2)$ ).

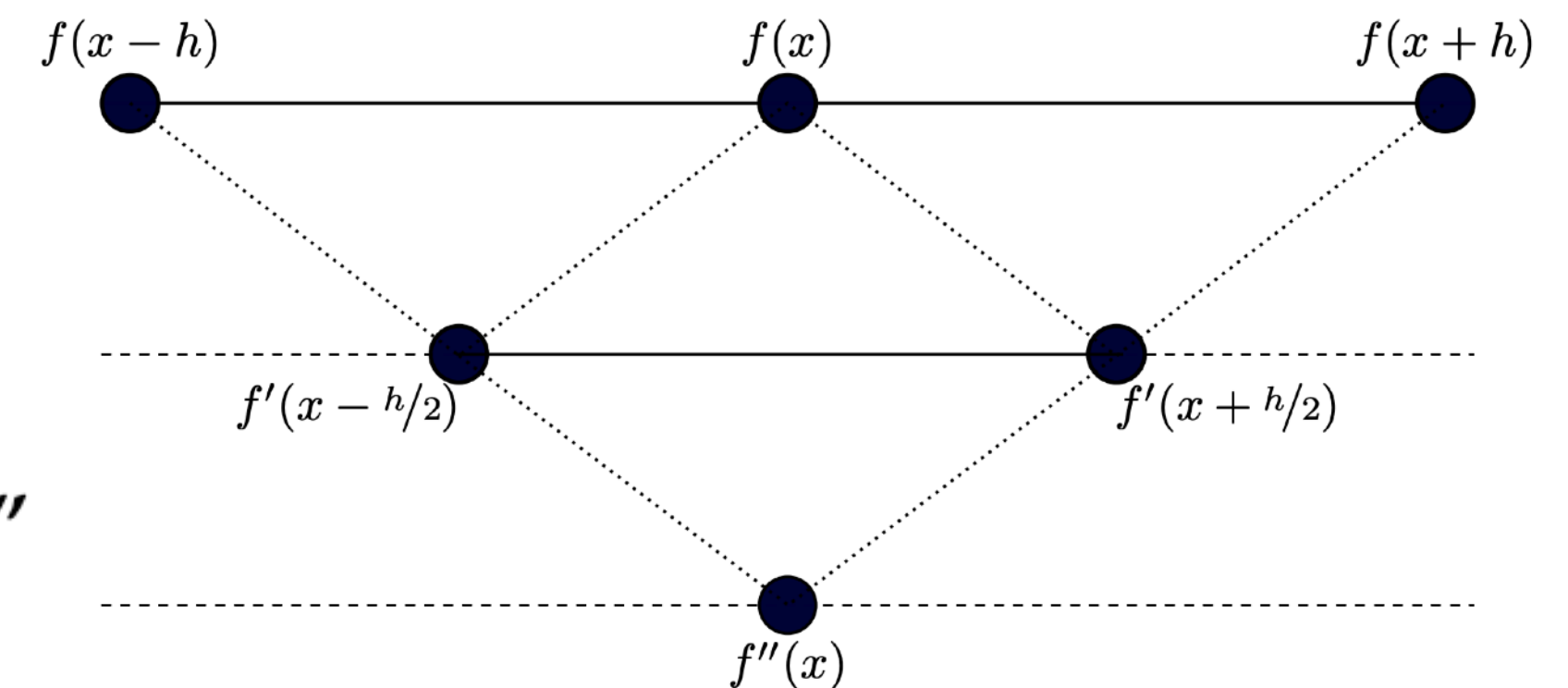
- For the second derivative:  $f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2)$ .

- This can also be viewed as a “finite difference of finite differences.”

- Conceptually:

$$\frac{f(x+h) - f(x)}{h} \Rightarrow f'(x + h/2), \quad \frac{f(x) - f(x-h)}{h} \Rightarrow f'(x - h/2).$$

Taking the difference of these slopes approximates  $f''(x)$ .



# Differentiation

## Richardson Extrapolation

- Improves accuracy of finite differences via **sequence acceleration**.

- Define:  $D(h) = \frac{f(x+h) - f(x)}{h} = f'(x) + \frac{1}{2}f''(x)h + O(h^2)$ .

- Similarly:  $D(\alpha h) = f'(x) + \frac{1}{2}f''(x)\alpha h + O(h^2)$ .

- Combine:

$$\begin{pmatrix} 1 & \frac{1}{2}h \\ 1 & \frac{1}{2}\alpha h \end{pmatrix} \begin{pmatrix} f'(x) \\ f''(x) \end{pmatrix} = \begin{pmatrix} D(h) \\ D(\alpha h) \end{pmatrix} + O(h^2).$$

- Solving yields:

$$f'(x) = \frac{1}{1-\alpha} [-\alpha D(h) + D(\alpha h)] + O(h^2).$$

- Converts two  $O(h)$  estimates into one  $O(h^2)$  estimate.

# Differentiation

## Example: Richardson Extrapolation in Practice

- Suppose  $f(x) = \sin(x^2)$ , approximate  $f'(1)$ .

- Use:

$$D(h) = \frac{\sin((1+h)^2) - \sin(1^2)}{h}.$$

- For  $h = 0.1, \alpha = 0.5$ :  $D(0.1) = 0.94145, \quad D(0.05) = 1.01735$ .

- Apply Richardson formula:

$$f'(1) \approx \frac{1}{1 - 0.5} [-0.5D(0.1) + D(0.05)] = 1.09325.$$

- True value:  $f'(1) = 2 \sin(1) \approx 1.0806$ .

- **Result:** Richardson extrapolation improves accuracy significantly.

# Differentiation

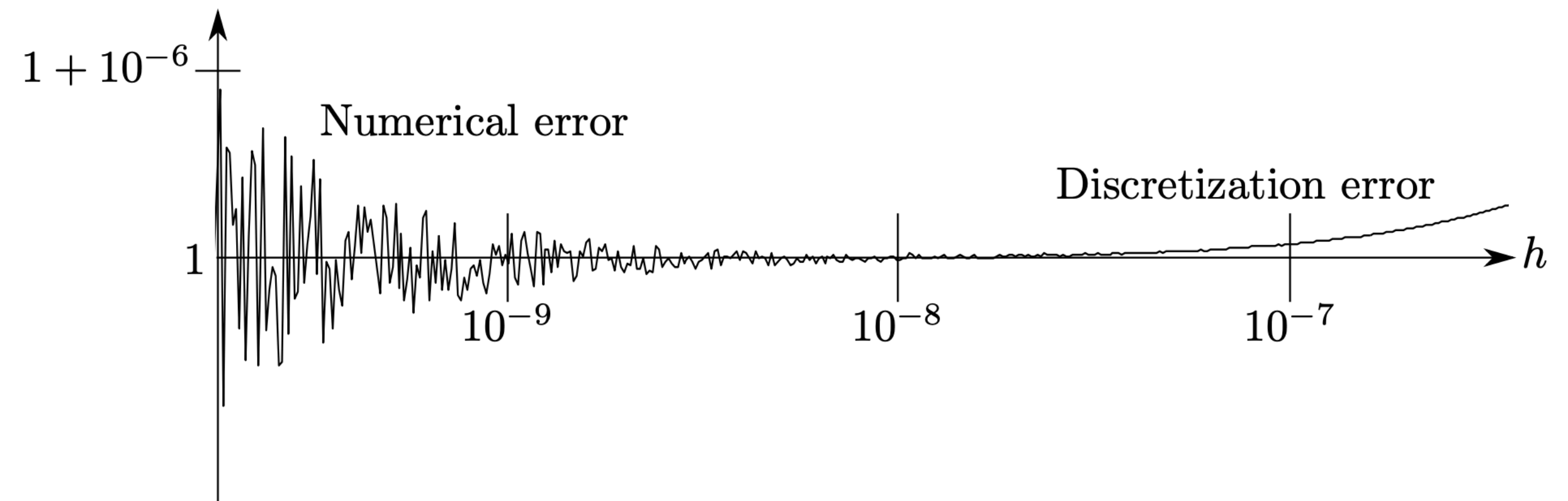
## Choosing Step Size

- As  $h \rightarrow 0$ , truncation error ( $O(h)$  or  $O(h^2)$ ) decreases.
- But floating-point roundoff error increases due to finite precision.

- The total error is a combination:

$$\text{Total error} \sim \frac{2\varepsilon}{h} + O(h),$$

where  $\varepsilon$  is machine precision or noise level.



- **Optimal  $h$ :** balance  $\frac{2\varepsilon}{h}$  and  $O(h)$  terms.
- Very small  $h \rightarrow$  roundoff dominates; Very large  $h \rightarrow$  discretization error dominates.
- Choose  $h$  near the minimum of the total error curve.

# Differentiation

## Numerical v.s. Automatic Differentiation

- **Numerical differentiation:**
  - Efficient but prone to rounding and discretization errors.
  - Limited reliability for noisy or rapidly varying functions.
- **Automatic differentiation (AD), e.g. PyTorch:**
  - Computes exact derivatives up to floating-point precision using the chain rule.
  - Implemented via operator overloading or symbolic rules.
  - Higher memory and computational overhead.
  - Does not simplify algebraic expressions; applies rules directly.

# Differentiation

## Integrated Quantities and Structure Preservation

- Focus: geometric and physical structure preservation in discrete settings.
- As  $\Delta x \rightarrow 0$ , discrete derivatives approximate continuous ones.
- In practice,  $\Delta x > 0$  – important to understand coarse approximations.
- Structure-preserving methods:
  - Ensure discrete analogs uphold continuous laws (e.g., conservation).
  - Avoid loss of geometric or physical meaning at finite resolution.

# Differentiation

## Example: Discrete Curvature of a 2D Curve

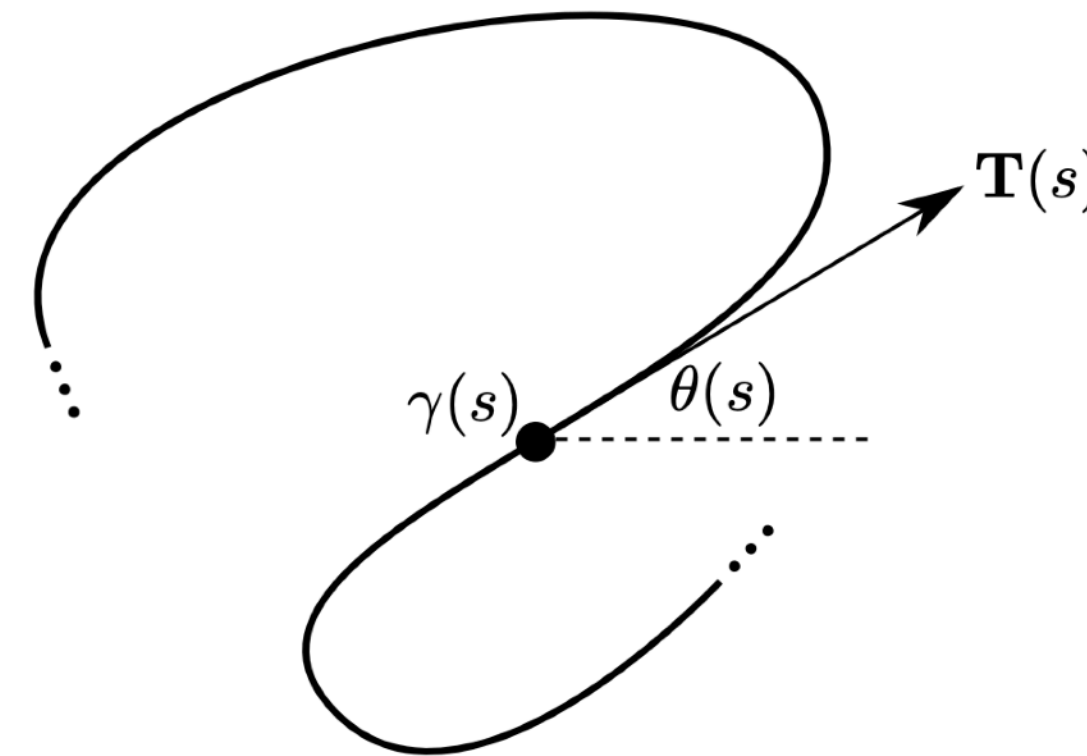
- Continuous case:  $\mathbf{T}(s) = \gamma'(s)$ ,  $\kappa(s) = \theta'(s)$ , where  $\theta$  is the tangent angle.

- **Turning number theorem:**  $\int_{s_0}^{s_1} \kappa(s) ds = 2\pi k$ .

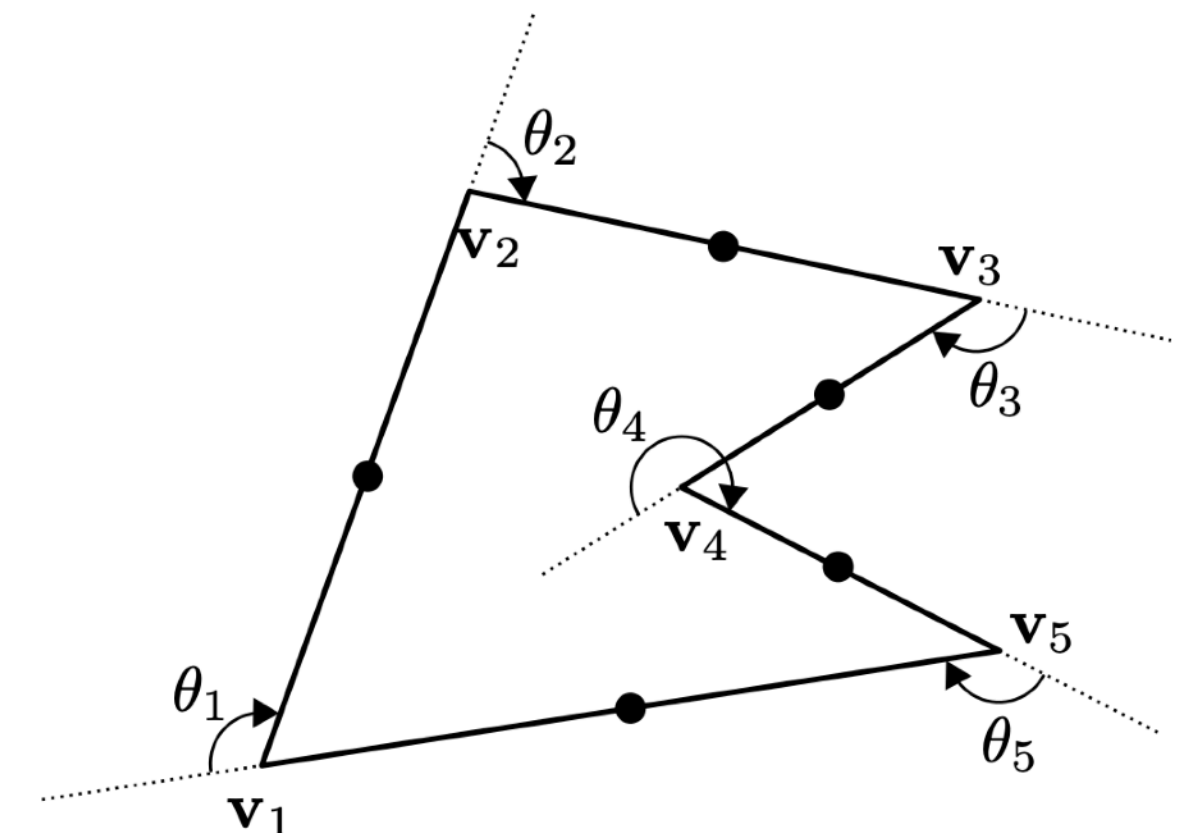
*For closed curves ( $s_0 = s_1$ )*

- Discretization:

- Represent  $\Gamma$  as line segments  $\mathbf{v}_i \leftrightarrow \mathbf{v}_{i+1}$ .
- At each vertex  $\mathbf{v}_i$ , define turning angle  $\theta_i$ .
- Integrated curvature over  $\Gamma_i$  equals  $\theta_i$ .



Continuous curve



Discrete curve

$$\int_{\Gamma} \kappa ds = \sum_i \theta_i = 2\pi k.$$

- This exact equality holds even for coarse approximations. Unlike approximating via divided differences.

# Differentiation

## Structure-Preserving Discretization: Discussion

- Example shows a **structure-preserving** derivative quantity.
- The turning number theorem holds for discrete  $\Gamma$  without  $\Delta x \rightarrow 0$ .
- However:
  - Pointwise convergence ( $\theta_i/|\Gamma_i| \rightarrow \kappa$ ) is not guaranteed.
  - Exact structure preservation and convergence may conflict.
- Ongoing research explores:
  - Trade-offs between structure preservation and numerical convergence.
  - Applications in geometry processing and computational physics.

# Table of Content

- Quadrature
- Differentiation