Lec 17: Integration and Differentiation

15-369/769: Numerical Methods (Foundations, ML & Visual Computing)

Instructor: Minchen Li

Motivation

Numerical Integration and Differentiation

- Interpolation techniques focused on predicting values of $f(x)$ 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 ϕ 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:
$$\int_a^b f(x)\,dx = \sum_i a_i \int_a^b \phi_i(x)\,dx = \sum_i c_i a_i, \qquad c_i := \int_a^b \phi_i(x)\,dx.$$
- 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:**

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

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

Quadrature

Method of Undetermined Coefficients

- Fix n sample points $x_1, \ldots, x_n$ 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 $f_1, \ldots, f_n$.
- This leads to a linear system:

$$\int_a^b f_j(x)\,dx = \sum_i w_i f_j(x_i), \qquad j = 1, \ldots, n.$$

- **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

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} b-a \\ \frac{1}{2}(b^2-a^2) \\ \frac{1}{3}(b^3-a^3) \\ \vdots \\ \frac{1}{n}(b^n-a^n) \end{pmatrix}.$$

- 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

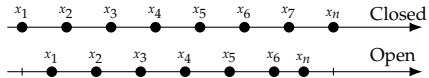
- Newton–Cotes quadrature integrates the interpolating polynomial through evenly spaced samples $x_i \in [a, b]$.
- Node placement:

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

- General form:

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

where ℓ_i are the Lagrange basis functions.



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 $(b,f(b))$.

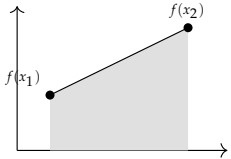
- **Simpson's rule ($n = 3$):**

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

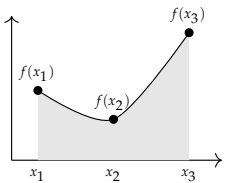
  Integrate the quadratic interpolant:

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

- Trapezoid is exact for degree 1 polynomials.
- Simpson integrates a quadratic interpolant, but symmetry makes the quadrature rule exact for all polynomials up to degree 3.



Trapezoidal rule



Simpson's rule

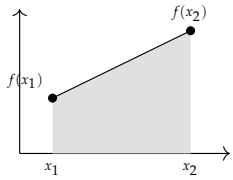
Quadrature

Open Newton–Cotes: Midpoint and Higher-Order Rules

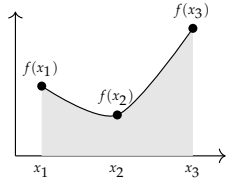
- **Midpoint rule ($n = 1$):**

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

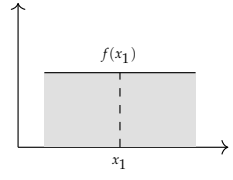
- This is the simplest open Newton–Cotes formula, using one interior node.
- Higher-n open rules follow the same pattern but omit endpoints, using evenly spaced interior points.



Trapezoidal rule



Simpson's rule



Midpoint rule

Quadrature

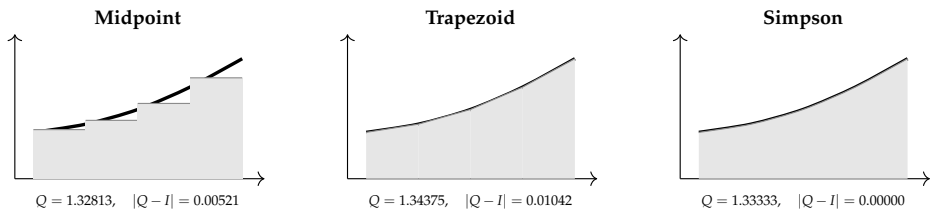
Composite Newton–Cotes Rules

- 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:** $\displaystyle\int_a^b f(x)\,dx \approx \sum_{i=1}^{k} f\left(\frac{x_{i+1} + x_i}{2}\right)\Delta x.$
- **Composite trapezoid:** $\displaystyle\int_a^b f(x)\,dx \approx \Delta x\left[\frac{1}{2}f(a) + \sum_{i=2}^{k} f(x_i) + \frac{1}{2}f(b)\right].$
- **Composite Simpson** (requires an even number k of subintervals):
  $\displaystyle\int_a^b f(x)\,dx \approx \frac{\Delta x}{3}\left[f(a) + 4f(x_2) + 2f(x_3) + \cdots + 4f(x_k) + f(b)\right].$

Quadrature

Composite Rules: Same Function, Same Grid

- Compare $f(x) = 1 + x^2$ on $[0, 1]$ using four equal subintervals ($\Delta x = 1/4$).
- Exact value: $I = \displaystyle\int_0^1 (1 + x^2)\, dx = \frac{4}{3} \approx 1.33333$.



Quadrature

Accuracy via Taylor Expansion

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

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

- **Midpoint rule:**

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

- **Trapezoid rule:**

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

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

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

Quadrature

Composite Accuracy Orders

- Single-interval accuracies:

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

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

$$\text{Composite midpoint: } O(\Delta x^2),$$
$$\text{Composite trapezoid: } O(\Delta x^2),$$
$$\text{Composite Simpson: } O(\Delta x^4).$$

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

Quadrature

Gaussian Quadrature: Concept

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

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

- 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 $1, x, x^2, x^3$:

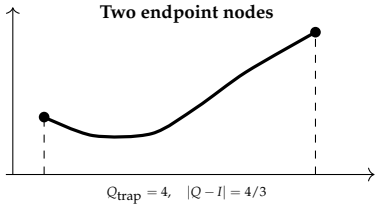
$$\begin{aligned} w_1 + w_2 &= 2, \\ w_1 x_1 + w_2 x_2 &= 0, \\ w_1 x_1^2 + w_2 x_2^2 &= \tfrac{2}{3}, \\ w_1 x_1^3 + w_2 x_2^3 &= 0. \end{aligned}$$

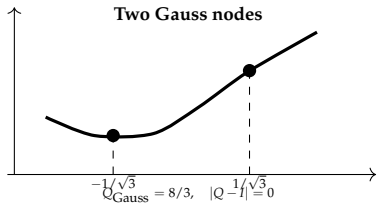
- Solution: $x_{1,2} = \pm \frac{1}{\sqrt{3}}$, $w_1 = w_2 = 1$. This is the classical two-point Gauss–Legendre rule.
- Properties:
  - Integrates all polynomials up to degree 3 exactly.
  - Not progressive (new nodes differ for each n).
  - Kronrod rules extend Gaussian quadrature by adding points for adaptive accuracy.

Quadrature

Why Optimizing Node Locations Helps

- Example: $f(x) = x^3 + x^2 + 1$ on $[-1, 1]$, with exact integral $I = 8/3$.
- Two endpoint samples (trapezoid) give 4, while two Gauss–Legendre samples at $\pm 1/\sqrt{3}$ give $8/3$ exactly.





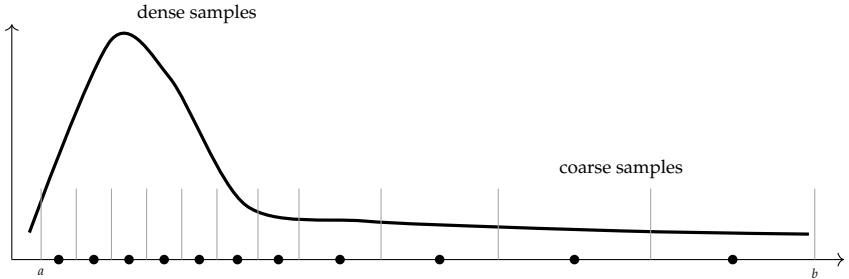
same number of samples $\Rightarrow$ much higher polynomial exactness

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 refine 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: Refine Only Where Needed



4. Final mesh:

samples concentrate near the sharp feature.

Quadrature

Multiple Integrals and Dimensionality

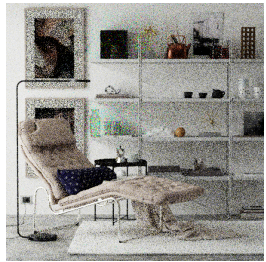
- For $f : \mathbb{R}^n \to \mathbb{R}$, we often seek $\int_\Omega f(\mathbf{x})\, d\mathbf{x}$, $\Omega \subseteq \mathbb{R}^n$.
- Example ($n = 2$): rectangular region integration: $\int_a^b \int_c^d f(x, y)\, dy\, dx$.
- **Curse of dimensionality:** Required sample count grows exponentially with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

Quadrature
Monte Carlo Integration

- For high-dimensional Ω, estimate

$$\int_\Omega f(\mathbf{x})\, d\mathbf{x} \approx \frac{|\Omega|}{k} \sum_{i=1}^{k} f(\mathbf{x}_i).$$

- **Convergence:** $O(1/\sqrt{k})$, independent of dimension.
- To reduce error by $2\times$, we need roughly $4\times$ as many samples.
- In Monte Carlo rendering, increasing samples per pixel progressively removes visible variance (noise).
- For complicated domains or distributions, rejection or importance sampling can improve efficiency.



8 spp: high variance



8192 spp: low variance

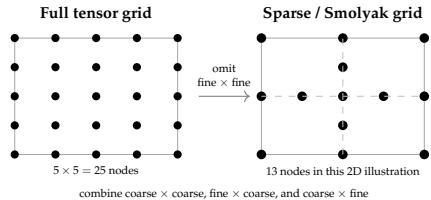
Source: Pharr, Jakob & Humphreys, *Physically Based Rendering*, 4th ed., Fig. 13.4; Kroken scene courtesy Angelo Ferretti.

Quadrature
Sparse and Smolyak Grids

- **Problem:** m quadrature nodes per dimension give m^n tensor-product samples.
- **Smolyak idea:** combine nested 1D rules at different refinement levels instead of refining every dimension simultaneously.
- In 2D, keep level pairs such as

$$i_1 + i_2 \leq q,$$

  so *fine* $\times$ *coarse* and *coarse* $\times$ *fine* rules are included, while the expensive *fine* $\times$ *fine* grid is omitted.
- For sufficiently smooth functions, this retains much of the accuracy with far fewer samples; it is most useful in moderate dimensions.



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:** High-degree Newton–Cotes rules can develop large alternating weights and become poorly conditioned.
- **Practical rule:** Prefer low-degree composite Newton–Cotes or Gaussian quadrature rather than one very high-degree Newton–Cotes rule.

Quadrature

Application: NeRF / Neural Volume Rendering

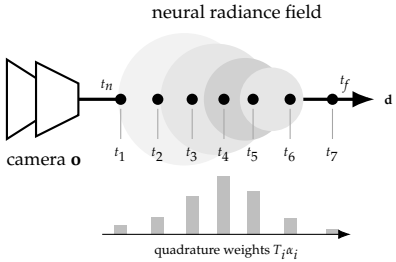
- A pixel casts a camera ray

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d},$$

  where $\mathbf{o}$ is the camera center, $\mathbf{d}$ points through the pixel, and t is distance along the ray.
- At a sampled 3D point, NeRF predicts σ: **density / opacity per unit distance**, and $\mathbf{c}$: **RGB radiance** viewed along $\mathbf{d}$.
- The pixel color integrates contributions from near t_n to far t_f:

$$\mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} \underbrace{T(t)}_{\text{visibility}} \underbrace{\sigma(\mathbf{r}(t))}_{\text{density}} \underbrace{\mathbf{c}(\mathbf{r}(t), \mathbf{d})}_{\text{color}} \, dt.$$

- $T(t)$ is **transmittance**: the fraction of light that reaches t without being absorbed.



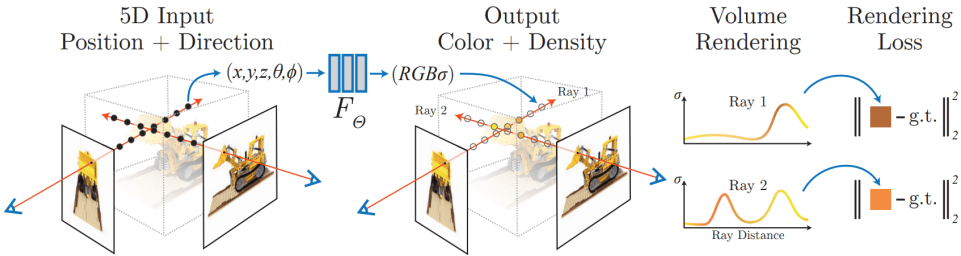
$$\widehat{\mathbf{C}} = \sum_i T_i \alpha_i \mathbf{c}_i, \qquad \alpha_i = 1 - e^{-\sigma_i \delta_i}.$$

$\delta_i = t_{i+1} - t_i$: sample spacing; $T_i\alpha_i$: quadrature weight.

Quadrature

Application: NeRF in Practice

- Training images with known camera poses supervise a continuous radiance field; a novel pixel is rendered by sampling and integrating along its camera ray.



Pipeline: posed input images → ray samples → MLP predicts $(\sigma, \mathbf{c})$ → volume-rendering quadrature → novel view.

Source: Mildenhall et al., "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," ECCV 2020, Fig. 2.

Quadrature

Application: Monte Carlo Integration → Path Tracing

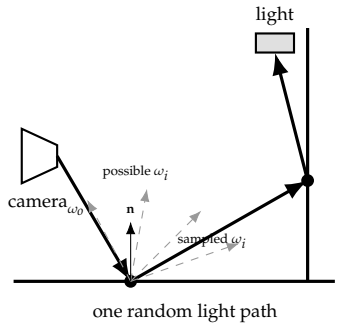
- At surface point $\mathbf{x}$, outgoing radiance in direction ω_o is

$$L_o = L_e + \int_{\mathcal{H}^2} f_r \, L_i(\mathbf{x}, \omega_i) \, \max(0, \mathbf{n} \cdot \omega_i) \, d\omega_i.$$

- Symbols: L_e = emitted light; L_i = incoming light; f_r = material/BRDF; $\mathbf{n}$ = surface normal; ω_i = incoming direction; ω_o = outgoing direction; $\mathcal{H}^2$ = hemisphere above the surface.
- Monte Carlo samples directions $\omega_j \sim p(\omega)$:

$$\widehat{I}_N = \frac{1}{N} \sum_{j=1}^{N} \frac{f_r L_i \, \max(0, \mathbf{n} \cdot \omega_j)}{p(\omega_j)}.$$

- $p(\omega)$ is the sampling probability density. **Importance sampling** gives more probability to directions likely to contribute more light, reducing variance.



Quadrature

Application: Path Tracing in Practice

- A photorealistic pixel is an average over many random paths that transport light through multiple reflections and occlusions.



Path tracing handles indirect illumination naturally because each random path samples a possible sequence of light-scattering events.

Source: Pharr, Jakob & Humphreys, *Physically Based Rendering*, 4th ed., Chapter 13 opener; Kroken scene courtesy Angelo Ferretti.

Table of Content

- Quadrature
- Differentiation

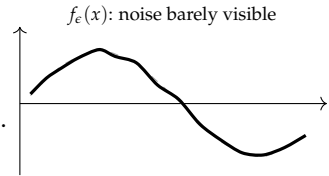
Differentiation

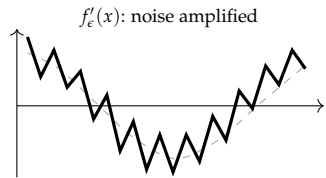
Why Differentiation Amplifies Noise

- A tiny high-frequency perturbation may barely change $f(x)$ but strongly change $f'(x)$.
- Example:

$$f_\epsilon(x) = \sin x + 0.02\sin(25x), \qquad f'_\epsilon(x) = \cos x + 0.5\cos(25x).$$

- Perturbation amplitude grows from 0.02 to 0.5: a $25\times$ amplification.
- In Fourier language, differentiation multiplies frequency ω by $i\omega$.
- This is why conditioning and noise matter for numerical derivatives.

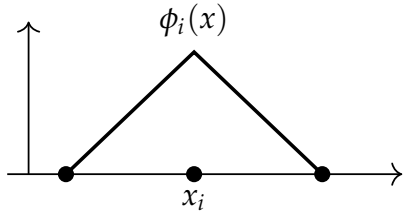


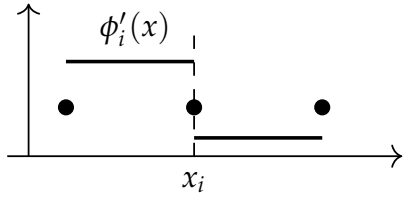


Differentiation

Differentiating Basis Functions

- For $f(x) = \sum_i a_i \phi_i(x)$, differentiation is linear:
$$f'(x) = \sum_i a_i \phi_i'(x).$$
- $\{\phi_i'\}$ spans the derivative space of functions represented by the original basis (some derivatives may be zero or linearly dependent).
- 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: $f'(x) = \lim_{h \to 0} \dfrac{f(x+h) - f(x)}{h}$.
- For finite h, $f'(x) \approx \dfrac{f(x+h) - f(x)}{h}$.
- Using Taylor expansion:

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

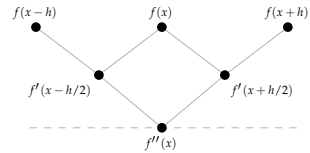
$$\Rightarrow \quad f'(x) = \frac{f(x+h) - f(x)}{h} + O(h).$$

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

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 is second-order accurate ($O(h^2)$).
- 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:
  $$\frac{f(x+h) - f(x)}{h} \Rightarrow f'(x + h/2), \qquad \frac{f(x) - f(x-h)}{h} \Rightarrow f'(x - h/2).$$
  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 and solve for $f'(x)$:

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

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

Differentiation

Example: Richardson Extrapolation in Practice

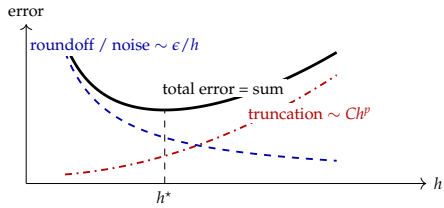
- Suppose $f(x) = \sin(x^2)$, approximate $f'(1)$.
- Use:
$$D(h) = \frac{\sin((1+h)^2) - \sin(1^2)}{h}.$$
- For $h = 0.1, \alpha = 0.5$: $D(0.1) = 0.94145$, $D(0.05) = 1.01735$.
- Apply Richardson formula:
$$f'(1) \approx \frac{1}{1-0.5}\left[-0.5D(0.1) + D(0.05)\right] = 1.09325.$$
- True value: $f'(1) = 2\cos(1) \approx 1.0806$.
- **Result:** Richardson extrapolation improves accuracy significantly.

Differentiation
Choosing Step Size

- As $h \to 0$, 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:

$$\text{Total error} \approx \frac{2\epsilon}{h} + O(h),$$

  where ϵ is machine precision or noise level.
- **Optimal h:** balance $\frac{2\epsilon}{h}$ 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:**
  - Simple and works with black-box function evaluations.
  - A full gradient in n variables typically needs $O(n)$ function evaluations.
  - Prone to truncation, rounding, and measurement-noise errors.
- **Automatic differentiation (AD), e.g. PyTorch:**
  - Computes derivatives of the implemented program accurately up to floating-point error using the chain rule.
  - Reverse-mode AD computes gradients of scalar objectives efficiently, often at a small multiple of the forward-pass cost.
  - Requires access to differentiable operations and stores intermediate information, increasing memory usage.
  - Numerical finite differences remain useful for checking AD implementations.

Differentiation

Integrated Quantities and Structure Preservation

- Focus: geometric and physical structure preservation in discrete settings.
- As $\Delta x \to 0$, 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 (arc-length parameter s):
  $T(s) = \gamma'(s)$ is the unit tangent and $\kappa(s) = \theta'(s)$.
- For a simple closed curve:

$$\int_\Gamma \kappa \, ds = 2\pi.$$

- For a polygon, put curvature at vertices using turning angles θ_i:

$$\int_\Gamma \kappa \, ds = \sum_i \theta_i = 2\pi.$$

- As the polygon is refined, its pointwise shape improves, but the **total turning is already exact at every resolution**.



shape converges; total turning is exact throughout

Differentiation

Structure-Preserving Discretization: Discussion

- Example shows a structure-preserving derivative quantity.
- The turning number theorem holds for discrete Γ without $\Delta x \to 0$.
- However:
  - Pointwise convergence ($\theta_i / |\Gamma_i| \to \kappa$) 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.

Differentiation

Application: Finite Differences as Gradient Checking

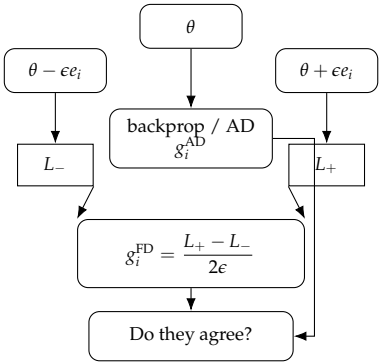
- Automatic differentiation gives

$$g_i^{\mathrm{AD}} = \frac{\partial L}{\partial \theta_i}.$$

- Independently estimate the same component with a centered finite difference:

$$g_i^{\mathrm{FD}} \approx \frac{L(\boldsymbol{\theta} + \epsilon \mathbf{e}_i) - L(\boldsymbol{\theta} - \epsilon \mathbf{e}_i)}{2\epsilon}.$$

- If g_i^{AD} and g_i^{FD} disagree, suspect an implementation bug, a nondifferentiable operation, or a poor ϵ.
- Step-size trade-off appears again: ϵ too large gives truncation error; too small gives roundoff/cancellation error.



Finite differences verify gradients; AD supplies them for optimization.

Table of Content

- Quadrature
- Differentiation