Demystifying Multi-Dimensional Integration

see also Lecture 6 :-)

Lecture Overview

1. Geometric Interpretation
2. Classical Quadrature vs. Monte Carlo
3. Efficiency Visualization

The Domain vs. Height Confusion

People frequently get confused about dimensions: *"We are writing a double integral ($\iint$). Why does it sometimes yield a 2D Area and other times a 3D Volume?"*

The Dimensional Jump

An integral always adds **one extra dimension** to the geometric domain you integrate over. The function inside dictates the **height** profile of that extra dimension.

The Domain vs. Height Confusion

People frequently get confused about dimensions: *"We are writing a double integral ($\iint$). Why does it sometimes yield a 2D Area and other times a 3D Volume?"*

The Dimensional Jump

An integral always adds **one extra dimension** to the geometric domain you integrate over. The function inside dictates the **height** profile of that extra dimension.

Scenario A: $f(x, y) = 1$

$$\iint_R 1 \, dx \, dy = \text{2D Area} \tag{1}$$

You are measuring the flat ground domain R. There is no variable height.

Scenario B: $f(x, y)$

$$\iint_R f(x, y) \, dx \, dy = \text{3D Volume} \tag{2}$$

The function acts as a variable height (a bending ceiling) above the ground domain R.

The "Flat Cake" Proof

If somebody insists a double integral must strictly output a volume, you can prove Scenario A using a 3D prism perspective:

- Imagine an object built above a 2D region R where the ceiling height is **exactly 1 unit everywhere** ($f(x, y) = 1$).
- The formula for the physical volume of a flat-topped cylinder or prism is:

$$\text{Volume} = \text{Area of Base} \times \text{Height} \tag{3}$$

- Substituting Height $= 1$:

$$\text{Volume (3D)} = \text{Area of Base (2D)} \times 1 \tag{4}$$

Takeaway

The double integral of 1 mathematically yields a 3D volume, but because the height is unity, its numerical value is **identical to the 2D area** of the domain!

Can a Volume be Negative? The Basement Effect

What happens mathematically when a function drops below zero?

- Calculus treats integration as a blind accumulation tool.
- The ground plane (xy-plane) is defined as exactly height $z = 0$.

Can a Volume be Negative? The Basement Effect

What happens mathematically when a function drops below zero?

- Calculus treats integration as a blind accumulation tool.
- The ground plane (xy-plane) is defined as exactly height $z = 0$.

The Basement Concept

- **When $f(x, y) > 0$:** The ceiling is above the floor. The function outputs a **positive volume**.
- **When $f(x, y) < 0$:** The ceiling drops below the floor, creating a **basement**. The function outputs a **negative volume**.

Net Result

If a function crosses the ground plane, the wiskundige machinery automatically subtracts the lower region from the upper region, returning the **net volume**:

$$I = \text{Volume}_{\text{above}} - \text{Volume}_{\text{below}} \tag{5}$$

Clearing Up the Monte Carlo "Volume" Term

In Monte Carlo integration, the multi-dimensional approximation formula contains a capital V:

$$I \approx \mathbf{V} \cdot \frac{1}{M} \sum_{i=1}^{M} f(X_i) \tag{6}$$

People get confused: *"If the integral outputs a 3D volume, why are we multiplying by another volume V? Are we squaring volumes?"*

Clearing Up the Monte Carlo "Volume" Term

In Monte Carlo integration, the multi-dimensional approximation formula contains a capital V:

$$I \approx \mathbf{V} \cdot \frac{1}{M} \sum_{i=1}^{M} f(X_i) \tag{6}$$

People get confused: *"If the integral outputs a 3D volume, why are we multiplying by another volume V? Are we squaring volumes?"*

The True Definition of V

The V in the Monte Carlo formula is **NOT** the 3D volume under the graph! V represents the **hypervolume (size) of the bounding domain** over which you sample.

- In a **1D integral** ($\int_a^b$): V is a **1D Length** $(b-a)$.
- In a **2D integral** ($\iint_R$): V is a **2D Area** of the bounding floor.
- In a **3D integral** ($\iiint_R$): V is a **3D Volume** of the bounding box.

Monte Carlo Intuition: Domain Size $\times$ Average Value

Instead of physical geometric shapes, Monte Carlo treats integration as finding the **statistical average value** of a function:

$$\text{Integral (I)} = \textbf{Size of Bounding Domain (V)} \times \textbf{Average Sample Height} \quad (7)$$

Monte Carlo Intuition: Domain Size $\times$ Average Value

Instead of physical geometric shapes, Monte Carlo treats integration as finding the **statistical average value** of a function:

$$\text{Integral (I)} = \textbf{Size of Bounding Domain (V)} \times \textbf{Average Sample Height} \quad (7)$$

Let's check the dimensions for a **double integral**:

- The random samples are chosen inside a 2D floor domain. The size of this bounding domain is a **2D Area (V)**.
- At each sample position, we evaluate $f(x, y)$, which represents a **1D Height**.
- The arithmetic mean $\frac{1}{M}\sum f(X_i)$ gives the **Average 1D Height**.

Dimensional Verification

$$\text{2D Area (V)} \times \text{1D Average Height} = \text{3D Volume (I)} \quad (8)$$

The dimensions match perfectly. The word "Volume" (V) in Monte Carlo is simply a generalized term for "domain content size".

Classical 1D Numerical Integration Methods

When an analytical solution is impossible, we divide a 1D domain into N intervals of width $\Delta x = \frac{b-a}{N}$ to compute approximations:

- **First-Order Approximation (Left/Right Riemann Sum):** Approximates each interval with a flat step matching the boundary value.

$$\text{Local Error: } \mathcal{O}(\Delta x^2) \quad \rightarrow \quad \text{Global Error: } \mathcal{O}(\mathbf{\Delta x^1})$$

- **Midpoint Rule (Constant Polynomial):** Approximates each interval with a rectangle evaluated at the center.

$$\text{Local Error: } \mathcal{O}(\Delta x^3) \quad \rightarrow \quad \text{Global Error: } \mathcal{O}(\mathbf{\Delta x^2})$$

- **Trapezoidal Rule (Linear Polynomial):** Connects subinterval boundaries with straight lines to form trapezoids.

$$\text{Local Error: } \mathcal{O}(\Delta x^3) \quad \rightarrow \quad \text{Global Error: } \mathcal{O}(\mathbf{\Delta x^2})$$

- **Simpson's Rule (Quadratic Polynomial):** Fits smooth quadratic parabolas across triplets of points.

$$\text{Local Error: } \mathcal{O}(\Delta x^5) \quad \rightarrow \quad \text{Global Error: } \mathcal{O}(\mathbf{\Delta x^4})$$

The Scaling Trap: Grid Points vs. Dimension d

To extend these deterministic rules to a d-dimensional space, we must build a rigid coordinate **grid**. If we place N points along each axis, the total number of evaluations M explodes:

$$M = N^d \quad \Longrightarrow \quad N = M^{1/d} \tag{9}$$

Grid Size Explosion ($N = 50$ points per axis)

- **1D Space:** $50^1 = \mathbf{50}$ computations.
- **3D Space:** $50^3 = \mathbf{125,000}$ computations.
- **6D Space:** $50^6 = \mathbf{15.6 \times 10^9}$ computations.
- **20D Space:** $50^{20} \approx \mathbf{9.5 \times 10^{33}}$ computations *(Impossible!)*

This exponential growth of computational cost is known as the **Curse of Dimensionality**.

The Collapse of Accuracy in Higher Dimensions

Because grid refinement forces our axis resolution to shrink ($N = M^{1/d}$), the global error decay rate degrades rapidly as the dimension d increases.

Global Error Scaling as a Function of Total Points M

- **First-Order Method:** $\mathcal{O}(N^{-1}) \quad \rightarrow \quad \mathcal{O}(\mathbf{M^{-1/d}})$
- **Midpoint & Trapezoidal Rules:** $\mathcal{O}(N^{-2}) \quad \rightarrow \quad \mathcal{O}(\mathbf{M^{-2/d}})$
- **Simpson's Rule:** $\mathcal{O}(N^{-4}) \quad \rightarrow \quad \mathcal{O}(\mathbf{M^{-4/d}})$

The Collapse of Accuracy in Higher Dimensions

Because grid refinement forces our axis resolution to shrink ($N = M^{1/d}$), the global error decay rate degrades rapidly as the dimension d increases.

Global Error Scaling as a Function of Total Points M

- **First-Order Method:** $\mathcal{O}(N^{-1}) \quad \rightarrow \quad \mathcal{O}(\mathbf{M^{-1/d}})$
- **Midpoint & Trapezoidal Rules:** $\mathcal{O}(N^{-2}) \quad \rightarrow \quad \mathcal{O}(\mathbf{M^{-2/d}})$
- **Simpson's Rule:** $\mathcal{O}(N^{-4}) \quad \rightarrow \quad \mathcal{O}(\mathbf{M^{-4/d}})$

The Monte Carlo Independence

Monte Carlo integration does not use grid steps Δx. Relying entirely on the Central Limit Theorem, its error decay rate is **strictly constant**:

$$\text{Monte Carlo Error} \approx \mathcal{O}(\mathbf{M^{-0.5}}) = \frac{\sigma}{\sqrt{M}} \tag{10}$$

The convergence rate is completely decoupled from the dimension d!

Efficiency Comparison Matrix

This matrix displays the global error convergence rates across different dimensions. Notice how the classical exponent shrinks toward zero as d grows.

Method	**1D Error**	**2D Error**	**5D Error**	**d-Dimensional Error**
First-Order	$\mathcal{O}(M^{-1.0})$	$\mathcal{O}(M^{-0.5})$	$\mathcal{O}(M^{-0.2})$	$\mathcal{O}(M^{-1/d})$
Trapezoidal / Mid	$\mathcal{O}(M^{-2.0})$	$\mathcal{O}(M^{-1.0})$	$\mathcal{O}(M^{-0.4})$	$\mathcal{O}(M^{-2/d})$
Simpson's Rule	$\mathcal{O}(M^{-4.0})$	$\mathcal{O}(M^{-2.0})$	$\mathcal{O}(M^{-0.8})$	$\mathcal{O}(M^{-4/d})$
Monte Carlo	$\mathcal{O}(\mathbf{M^{-0.5}})$	$\mathcal{O}(\mathbf{M^{-0.5}})$	$\mathcal{O}(\mathbf{M^{-0.5}})$	$\mathcal{O}(\mathbf{M^{-0.5}})$ **(Winner)**

Table: Error dependency comparison based on total computational cost M.

The Strategic Turning Point

- For $d < 8$, Simpson's rule is superior because $-4/d < -0.5$.
- For $d = 8$, Simpson's rule and Monte Carlo break even ($\mathcal{O}(M^{-0.5})$).
- For $d > 8$, Monte Carlo is the only viable mathematical option.

Visualizing Cost vs. Error Convergence

The log-log graph below shows the error decay as computation cost M increases. Notice how the deterministic 10D line flatlines due to the grid trap.

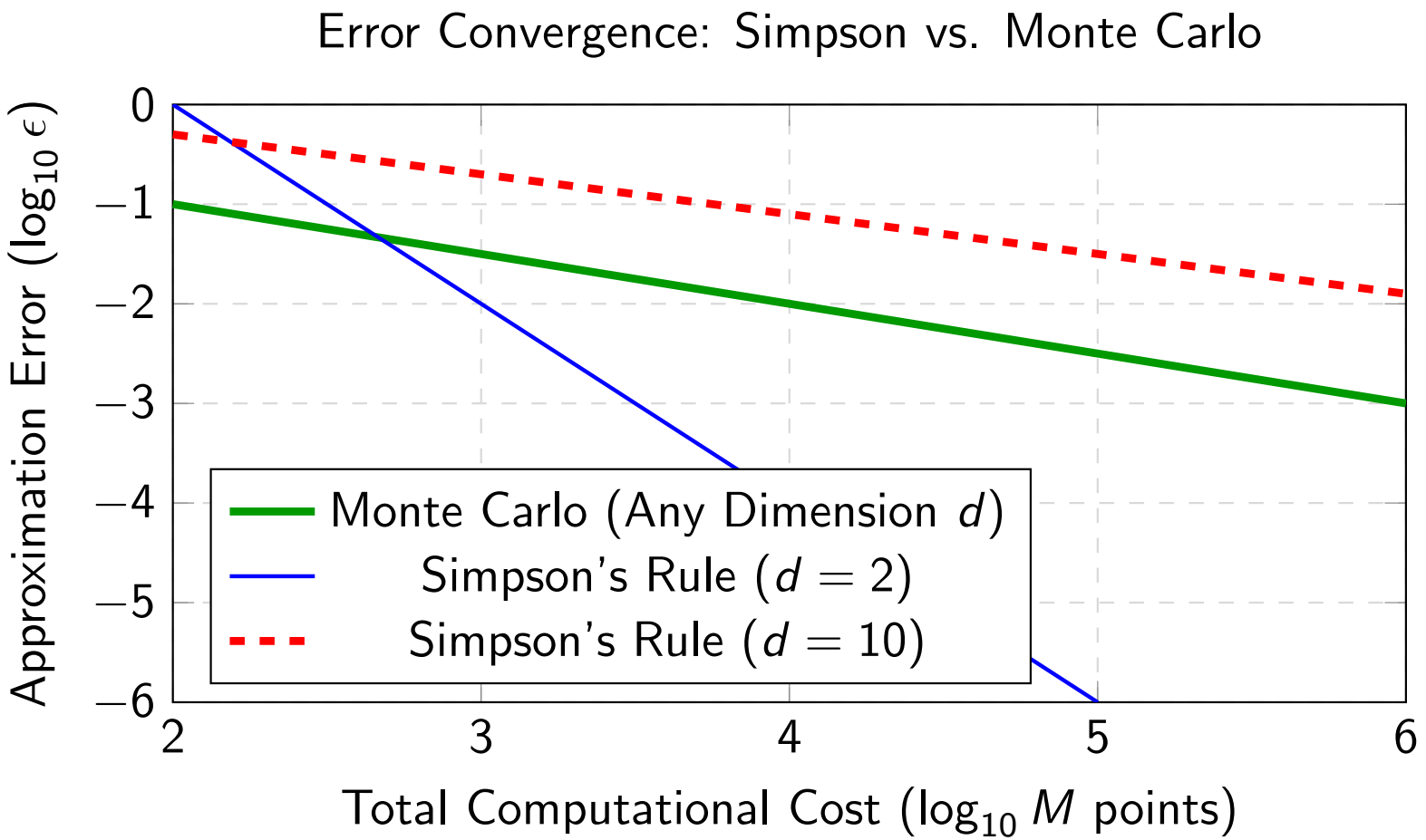


Figure: Logarithmic convergence slopes. Steeper downward paths mean higher efficiency.

Core Summary for the Exam

- **Low Dimensions ($d \leq 3$):** Always use grid methods (Simpson). They exploit local smoothness and converge incredibly fast.
- **The Boundary Line ($d = 8$):** Traditional grid calculus hits a performance wall.
- **High Dimensions ($d > 8$):** Classical integration rules suffer from a severe accuracy breakdown. Monte Carlo is immune to this scaling penalty because it completely bypasses the grid step restriction.