Monte Carlo and Quasi Monte Carlo Methods

October 13, 2025

Outline

Cold War Origins

- **1946**: Stanislaw Ulam conceives Monte Carlo at Los Alamos
- Manhattan Project: neutron diffusion for nuclear weapons
- **Geopolitical Context**:
  - Cold War begins: US vs Soviet Union
  - US controls 50% of world industrial capacity
  - Government invests heavily in "big science"
- **Key Contributors**: Ulam, von Neumann, Metropolis

Computing Technology in 1946

- **ENIAC (1946)**: First electronic digital programmable computer in the U.S
  - 18,000 vacuum tubes (invented 1904-1907)
  - Developed for ballistics calculations
  - Rare and extremely expensive
- Only governments and major labs had access
- Monte Carlo was *enabled* by early computers
- Named "Monte Carlo" as security codename

Early Milestones

- **1950**: Kahn-Ulam random walk methods
- **1950s**: Koksma-Hlawka inequality (theoretical foundation)
- **1953**: Metropolis algorithm (MCMC)
- **Key Property**: $O(N^{-1/2})$ convergence, dimension-independent
- Applications: neutron transport, physics, fluid dynamics

Hardware Revolution

Three Key Innovations

- **Transistors (1947)**: Replace vacuum tubes
- **MOSFETs (1959)**: Enable chip integration
- **Microprocessors (1970s)**: Affordable personal computers

Impact: Democratization of computing power

- From military labs to universities to homes
- Monte Carlo spreads across disciplines
- Variance reduction techniques developed

Quasi-Monte Carlo Emerges

- **Goal**: Improve upon $O(N^{-1/2})$ convergence
- Use *deterministic* low-discrepancy sequences
- **Key Sequences**:
  - 1950s: Koksma-Hlawka Inequality
  - 1960: Halton sequence (J. Halton)
  - 1967: Sobol' sequence (I. M. Sobol')
  - 1982: Faure sequence (H. Faure)
  - 1992: Niederreiter sequences (H. Niederreiter)
- **Theoretical Rate**: $O((\log N)^d/N)$ for smooth functions

Caflisch's Survey (1998)

Author: Russel E. Caflisch

- Born 1954
- PhD NYU Courant (1978)
- Stanford, UCLA, NYU
- Leader in applied math

1998 World:

- Cold War ended (1991)
- Dot-com boom
- Democratized computing
- Complex financial markets

Key Work: Mid-1990s Brownian-bridge QMC for finance

Historical Shift: 1946 vs 1998

Aspect	**1946**	**1998**
Era	Cold War begins	Post-Cold War
Focus	Military/nuclear	Finance/internet
Computing	Rare, expensive	Widespread, cheap
Access	Government labs	Universities, industry
Applications	Weapons physics	Markets, engineering

Context: Asian Financial Crisis (1997-98), Japan's 1990s crisis

Leading Researchers (1990s)

- **Art B. Owen** (Stanford): Randomized QMC
  - Combines QMC convergence + MC error estimation
- **Harald Niederreiter** (Austria): Generalized sequences
- **Ilya M. Sobol'** (Russia): Sequence refinement + sensitivity analysis
- **Applications**: Financial derivatives, risk management, high-dimensional integration

Competing Methods

Sparse Grids

- Smolyak (1963): Original formulation
- Bungartz & Griebel (1990s): Computational implementation
- **Advantage**: Good for moderate dimensions
- **Limitation**: QMC better for very high dimensions

Curse of Dimensionality (Bellman, 1961): Central challenge for all methods

Should Science Have Boundaries?

The Dilemma

"Science does not have boundaries, but scientists do"

Historical Examples:

- Manhattan Project: Weapons $\leftrightarrow$ Medicine/Energy
- Monte Carlo: Military origins $\rightarrow$ Peaceful applications
- AI today: Innovation $\leftrightarrow$ Job displacement, fraud, warfare

Question: Can we separate knowledge from its uses when funding shapes research?

Modern Challenges

- **AI Impact**:
  - Job displacement
  - Enabling fraud and surveillance
  - Military applications
- **National Competition**:
  - Technology restrictions between nations
  - Race for technological superiority
- **Sustainable Development**:
  - Genuine solution or rhetorical device?
  - Tools enable both efficiency and extraction

Funding and Accountability

Reality

Science depends on funding $\rightarrow$ Funding shapes priorities

Historical Pattern:

- 1946: Military needs accelerated computing
- Today: Corporate/government sponsors shape AI, biotech

The Real Question: How do we ensure science serves human flourishing?

- Public deliberation on research priorities
- Transparent governance of dangerous tech
- International cooperation on safety standards

Conclusion

From 1946 to Present:

- Monte Carlo: Cold War child of necessity
- QMC: Post-Cold War optimization
- Computing: From military to universal

Ethical Imperative:

- Technology grows more powerful
- Responsibilities grow correspondingly
- Question: Not *whether* to constrain, but *how*
- Goal: Promote beneficial development while minimizing harm

Introduction and Motivation

- **Versatility vs. Speed Trade-off**: Monte Carlo is one of the most versatile and widely used numerical methods, but suffers from slow $O(N^{-1/2})$ convergence rate.
- **Dimension Independence**: Unlike grid-based methods that deteriorate exponentially with dimension, Monte Carlo's convergence is dimension independent making it the only viable approach for high-dimensional problems (atomic physics to finance).
- **Computational Impact**: Enormous amounts of computer time spent on Monte Carlo computations create significant opportunity for improvement through better methods.
- **Acceleration Strategies**: Two main approaches: variance reduction (reducing the constant in $O(N^{-1/2})$) and quasi-Monte Carlo (improving convergence to $O((\log N)^k N^{-1})$ through deterministic low-discrepancy sequences).
- **Key Applications**: Critical for financial derivatives pricing, rarefied gas dynamics simulations, and path integrals where traditional methods fail.

Detailed Explanation of $O(N^{-1/2})$ Convergence

Setup: We want to estimate the expectation $\mu = \mathbb{E}[f(X)]$ where:

- X is a random variable
- $f(X)$ is some function we're evaluating
- We don't know μ exactly, so we approximate it using Monte Carlo

Step 1: Sample Average (Monte Carlo Estimator)

We generate N independent samples: $x_1, x_2, \ldots, x_N$

Our estimate is:

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

This is just the average of N evaluations of f.

Step 2: Variance of the Sample Average

Here's where the key mathematics happens:

Individual samples:

- Each $f(x_i)$ has variance $\sigma^2 = \text{Var}[f(X)]$
- The samples are **independent**

Variance of the sum:
Since the samples are independent:

$$\text{Var}\left[\sum_{i=1}^{N} f(x_i)\right] = \sum_{i=1}^{N} \text{Var}[f(x_i)] = N \cdot \sigma^2$$

Step 2: Variance of the Sample Average (continued)

Variance of the average:
When you divide by N:

$$\begin{aligned}
\mathrm{Var}[\hat{\mu}] &= \mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N} f(x_i)\right] \\
&= \frac{1}{N^2} \cdot \mathrm{Var}\left[\sum_{i=1}^{N} f(x_i)\right] \\
&= \frac{1}{N^2} \cdot N \cdot \sigma^2 \\
&= \frac{\sigma^2}{N}
\end{aligned}$$

Step 3: Standard Error

The **standard error** (SE) is the standard deviation of our estimator:

$$\text{SE} = \sqrt{\text{Var}[\hat{\mu}]} = \sqrt{\frac{\sigma^2}{N}} = \frac{\sigma}{\sqrt{N}}$$

This tells us the typical magnitude of error in our estimate.

Step 4: Error Scales as $O(N^{-1/2})$

The error in our estimate is proportional to the standard error:

$$\varepsilon \propto \frac{\sigma}{\sqrt{N}} = \sigma \cdot N^{-1/2}$$

Therefore: $\varepsilon = O(N^{-1/2})$

Key Insight: The error decreases with the square root of the number of samples, not linearly. This means:

- To halve the error, you need $4\times$ more samples
- To reduce error by $10\times$, you need $100\times$ more samples

Why This Matters: Practical Implications

The $O(N^{-1/2})$ convergence rate means:

- **Dimension-independent**: Same rate in 2D or 1000D
- **Relatively slow**: Need many samples for high precision
- **Universal**: Applies to any Monte Carlo integration

Example calculation:

- Current error with $N = 1000$ samples: ε_0
- Want error $\varepsilon_0/10$ ($10\times$ better accuracy)
- Need: $N = 1000 \times 10^2 = 100{,}000$ samples

This is why variance reduction and quasi-Monte Carlo methods are so valuable!

Monte Carlo Integration

Basic Formula: The integral can be expressed as an expectation:

$$I[f] = \int_0^1 f(x)\,dx = \bar{f} = E[f(x)]$$

Monte Carlo Approximation: Sample $\{x_n\}$ from uniform distribution, then:

$$I_N[f] = \frac{1}{N}\sum_{n=1}^{N} f(x_n) \quad \text{converges to } I[f]$$

Error Measures:

- Error: $\epsilon_N[f] = I[f] - I_N[f]$ (difference between true and estimated value)
- Bias: $E[\epsilon_N[f]]$ - expected value of error (measures systematic error)
- Root Mean Square Error (RMSE): $E[\epsilon_N[f]^2]^{1/2}$ (typical error magnitude)
- Convergence rate: $O(N^{-1/2})$ independent of dimension

Monte Carlo is Unbiased

What is Bias? Bias measures whether your estimator is systematically off-target on average.

For Monte Carlo: Monte Carlo integration is **unbiased**, meaning:

$$E[I_N[f]] = I[f]$$

Proof:

$$\begin{aligned} E[I_N[f]] &= E\left[\frac{1}{N}\sum_{n=1}^{N} f(x_n)\right] \\ &= \frac{1}{N}\sum_{n=1}^{N} E[f(x_n)] \\ &= \frac{1}{N}\sum_{n=1}^{N} I[f] = I[f] \end{aligned}$$

Therefore, Bias $= E[\epsilon_N[f]] = E[I[f] - I_N[f]] = I[f] - E[I_N[f]] = 0$

What this means: On average, Monte Carlo estimates are centered around the true value some high, some low, but averaging to the correct answer.

Grid-Based Methods vs Monte Carlo

The Basic Idea Behind Grids:

- Since we can't solve most integrals analytically, we approximate them by:
  1. Sampling the function at specific grid points
  2. Estimating the area using these sample values
  3. Using geometric shapes (rectangles, trapezoids) to approximate area under the curve

Common Grid-Based Integration Rules:

- **Rectangle Rule**: Order 1, $O(N^{-1/d})$
- **Trapezoidal Rule**: Order 2, $O(N^{-2/d})$ - uses trapezoids
- **Simpson's Rule**: Order 4, $O(N^{-4/d})$ - uses parabolas
- **Gaussian Quadrature**: Higher order, optimal point placement

General convergence rate for grid methods: $O(N^{-k/d})$ where k is method order, d is dimension

The Curse of Dimensionality

Grid Methods Deteriorate Exponentially with Dimension:

Number of Points Needed:

- 1D: N points
- 2D: N^2 points (for same accuracy per dimension)
- 3D: N^3 points
- d-dimensions: N^d points

Example: For 100 points per dimension:

- 1D: 100 points
- 2D: 10,000 points (100×100 grid)
- 3D: 1,000,000 points ($100 \times 100 \times 100$ grid)
- 20D: 10^{40} points (completely impractical!)

The computational explosion: Number of function evaluations grows exponentially with dimension, making grid methods impractical in high dimensions.

Why Higher Order Methods Fail in High Dimensions

Why Higher Order is Better (in low dimensions):

- Higher order methods approximate the function more accurately in each interval

The Dimensional Disaster:

- 1D with order 4: $O(N^{-4})$ — excellent!
- 10D with order 4: $O(N^{-4/10}) = O(N^{-0.4})$ — terrible!

As dimension d increases, even high-order methods (large k) become ineffective because k/d gets smaller and smaller.

This is why Monte Carlo becomes superior: Monte Carlo's $O(N^{-1/2})$ doesn't depend on dimension at all, making it superior in high dimensions despite being "only" order 1 equivalent.

Monte Carlo Advantage: Dimension Independence

Key Advantages of Monte Carlo:

- **Same Computational Cost:** Uses random points, not a grid
- **Always** converges at $O(N^{-1/2})$ regardless of dimension
- Same 10,000 random points work equally well in 1D, 20D, or 100D

- **Practical Efficiency:**
  - No exponential growth in required points
  - Easy to implement and parallelize
  - Can handle irregular domains naturally

- **When MC is Essential:**
  - High-dimensional problems ($d > 4$)
  - Financial derivatives pricing
  - Particle physics simulations
  - Complex probability distributions

Conclusion: Grid methods become impossibly expensive in high dimensions, while Monte Carlo maintains the same computational cost regardless of dimension.

Generation and Sampling Methods

Random Number Generators

Key Concept: Number generation by computers is **not truly random**.

How it works:

- Computers use a **seed** value to start the sequence
- Functions like `rand(0)`, `rand(1)` generate numbers
- The numbers are **fake but look scattered**
- If someone knows the seed, they can **predict every next number**
- That's why we call them **pseudorandom**

Is it safe for Monte Carlo?

- **Yes!** Monte Carlo does not care if numbers are truly random
- It only cares if **the average settles right**
- As long as the sequence has good statistical properties, Monte Carlo will converge correctly

Sampling Methods

Problem: Standard random number generators produce **uniform** random variables. But we often need to sample from **non-uniform** distributions with density $p(x)$.

Mathematical Framework: For a non-uniform random variable with density $p(x)$ (how likely is the value of x), the expectation of function $f(x)$ is:

$$E[f] = I[f] = \int f(x)p(x)\,dx$$

For sequence $\{x_n\}$ distributed according to density p, the empirical approximation is:

$$I_N[f] = \frac{1}{N}\sum_{n=1}^{N} f(x_n)$$

The resulting quadrature error follows Central Limit Theorem:

$$e_N[f] \sim N^{-1/2}\sigma v \quad \text{where} \quad \sigma^2 = \int (f-\bar{f})^2 p(x)\,dx$$

Variance Reduction

What is Variance Reduction?

Making Monte Carlo more efficient by reducing the "noise" in estimates without increasing sample points.

The Core Problem:

- Monte Carlo error: $O(\sigma N^{-1/2})$ where σ is standard deviation
- Convergence rate $N^{-1/2}$ is fixed
- But we CAN reduce σ to get smaller errors with same computational cost

The Key Insight:

- Error = Rate $\times$ Constant = $N^{-1/2} \times \sigma$
- Can't easily change the rate
- CAN change the constant σ through clever techniques

Goal: Use mathematical cleverness instead of brute-forcing with more samples to get same accuracy with fewer function evaluations.

Types of Variance Reduction Methods

Five Main Techniques:

1. **Antithetic Variables** - Use paired samples that partially cancel errors
2. **Control Variates** - Subtract function with known integral
3. **Stratification** - Divide domain and sample systematically from each region
4. **Importance Sampling** - Sample more where function is large/important
5. **Russian Roulette** - For infinite series, randomly truncate while staying unbiased

Trade-off: Extra setup work vs. fewer samples needed

Antithetic Variables

Basic Idea: For each sample x, also use $-x$ to create pairs that cancel errors.

Formula:

$$I_N[f] = \frac{1}{2N} \sum_{n=1}^{N} \{f(x_n) + f(-x_n)\}$$

Why it works: For symmetric distributions, errors from $f(x)$ and $f(-x)$ tend to cancel out, especially for the linear terms in Taylor expansion.

Variance Reduction:

- Standard MC: error $\propto \sigma$
- Antithetic: error $\propto \sigma^2$ (for small variance)
- Linear terms in error cancel exactly
- Most effective for smooth, symmetric problems

Example: Random walk - for each path $(x_1, x_2, \ldots, x_k)$, also use path $(-x_1, -x_2, \ldots, -x_k)$

Control Variates

Basic Idea: Use a similar function g with known integral to reduce variance.

Rewrite the integral:

$$I[f] = \int f(x)\,dx = \int (f(x) - g(x))\,dx + \int g(x)\,dx$$

Monte Carlo formula:

$$I_n[f] = \frac{1}{N}\sum_{n=1}^{N}(f(x_n) - g(x_n)) + I[g]$$

where $I[g]$ is known exactly.

Variance:

$$\sigma^2_{f-g} = \int (f(x) - g(x) - I[f-g])^2\,dx$$

Effective when $\sigma_{f-g} \ll \sigma_f$ (i.e., g is close to f)

Optimal version: Use $\lambda g(x)$ with optimal $\lambda = E[fg]/E[g^2]$

Stratification

Basic Idea: Divide integration domain into M subsets Ω_k and sample systematically from each.

Domain partition:

$$\Omega = \bigcup_{k=1}^{M} \Omega_k, \quad |\Omega_k| = 1/M$$

Stratified formula:

$$I_N[f] = \frac{1}{N} \sum_{k=1}^{M} \sum_{i=1}^{N/M} f(x_i^{(k)})$$

where $x_i^{(k)}$ are sampled uniformly from Ω_k

Variance:

$$\sigma_s^2 = \sum_{k=1}^{M} \int_{\Omega_k} (f(x) - \bar{f}_k)^2 \, dx$$

Key result: Always $\sigma_s \leq \sigma$ (stratification always helps!)

Why: Reduces variance by ensuring uniform coverage of domain

Importance Sampling

Basic Idea: Sample more frequently where the integrand is large.

Rewrite integral with density $p(x)$:

$$I[f] = \int f(x)\,dx = \int \frac{f(x)}{p(x)} p(x)\,dx$$

Sample from density $p(x)$:

$$I_N[f] = \frac{1}{N} \sum_{n=1}^{N} \frac{f(x_n)}{p(x_n)}$$

where $x_n \sim p(x)$

Variance:

$$\sigma_p^2 = \int \left(\frac{f(x)}{p(x)} - I[f] \right)^2 p(x)\,dx$$

Optimal choice: $p(x) \propto |f(x)|$ makes f/p nearly constant

Use case: Emphasize rare but important events (e.g., option pricing with extreme market moves)

Russian Roulette

Basic Idea: For infinite sums, randomly truncate while maintaining unbiased estimate.

Problem: Computing infinite sum $S = \sum_{n=0}^{\infty} a_n$ with $|a_n| < c\lambda^n$

Method:

1. Choose $\lambda < \kappa < 1$
2. Let M be random with $\text{Prob}(M > n) = \kappa^n$
3. Define random sum: $\tilde{S} = \sum_{n=0}^{M} \kappa^{-n} a_n$

Key property:

$$E[\tilde{S}] = \sum_{n=0}^{\infty} \text{Prob}(M > n) \cdot \kappa^{-n} a_n = \sum_{n=0}^{\infty} a_n = S$$

Result: Finite sum (uniformly bounded) with correct expectation

Application: Iteration of integral equations, neutron transport

Quasi MC vs Simple MC - Comparison

Quasi MC = Uniform distribution points
MC → Random Sampling

Quasi MC

1. Uses **deterministic** sequences that are designed to fill the space more evenly
2. These sequences are called **low-discrepancy sequences** & they help in covering the integration domain more uniformly
3. QMC aims for more uniform distribution of points

Simple MC

1. Random sampling - relies on randomness
2. High discrepancy sequence
3. MC relies on purely random points, which can sometimes cluster or leave gaps

MC vs QMC: Practical Comparison - Estimating π

Problem: Estimate π by randomly sampling points in unit square and checking if they fall inside unit circle.

Monte Carlo (Random Points)

```
import random
import math

n = 1000000
points_in_mc = 0

for _ in range(n):
    x = random.random()
    y = random.random()
    if math.sqrt(x**2 + y**2) <= 1:
        points_in_mc += 1

pi_mc = 4 * (points_in_mc / n)
```

Result:

- MC $\pi \approx 3.141176$
- Time: 0.2874s
- Uses pure random sampling

Quasi-Monte Carlo (Halton Sequence)

```
def halton_sequence(index, base):
    result = 0.0
    f = 1.0 / base
    i = index
    while i > 0:
        result += f * (i % base)
        i = math.floor(i / base)
        f /= base
    return result

def generate_halton_point(i):
    x = halton_sequence(i+1, 2)
    y = halton_sequence(i+1, 3)
    return x, y

points_in_qmc = 0
for i in range(n):
    x, y = generate_halton_point(i)
    if math.sqrt(x**2 + y**2) <= 1:
        points_in_qmc += 1

pi_qmc = 4 * (points_in_qmc / n)
```

Result:

- QMC $\pi \approx 3.141572$
- Time: 3.0317s
- Uses deterministic low-discrepancy sequence

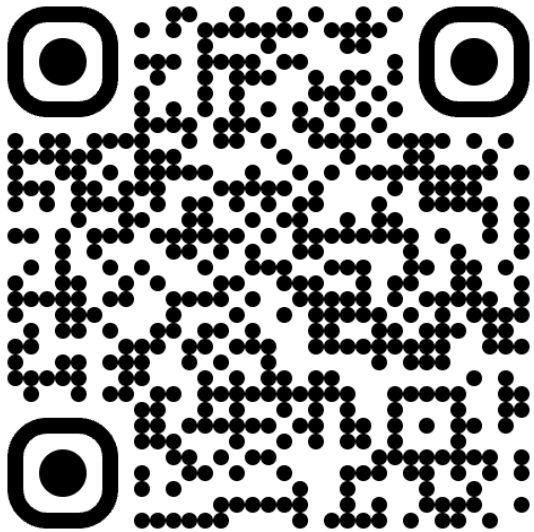
Analysis: MC vs QMC for π Estimation

Key Observations:

- **Accuracy:**
  - True value: $\pi = 3.141592653...$
  - MC Error: $|3.141176 - 3.141593| = 0.000417$
  - QMC Error: $|3.141572 - 3.141593| = 0.000021$
  - QMC is **20× more accurate** for same number of points
- **Speed:**
  - MC is faster (0.29s vs 3.03s)
  - Halton sequence generation adds overhead
  - But QMC needs fewer points for same accuracy
- **Trade-off:**
  - MC: Fast per sample, but needs more samples
  - QMC: Slower per sample, but needs far fewer samples
  - For high accuracy requirements, QMC wins overall

Implementation and Collaboration

- All experiments implemented in Python.
- Collaboration: **Vaibhav Mangroliya** and **Boaz Habite**.
- Scan the QR code to head to the implementations of various Monte Carlo (MC) and Quasi-Monte Carlo (QMC) variance reduction techniques for estimating the value of π.



Experiments Overview

- Reproduction of results from Caflisch (1998) on Monte Carlo (MC) and Quasi-Monte Carlo (QMC) methods.
- Experiments implemented in Python to compare sampling and convergence behaviors.

- Core objectives:
  - Demonstrate sampling differences (MC vs QMC)
  - Estimate π across multiple dimensions
  - Apply Variance Reduction (Antithetic, Control Variates)
  - Evaluate QMC (Sobol) and MC with Variance Reduction on continuous 4D integral

Sampling Techniques

- **Monte Carlo (MC):** Pseudo-random sampling, Points are independent, some clustering and gaps.
- **Quasi-Monte Carlo (QMC):** Low-discrepancy sequences (Sobol), Points are dependent, Evenly spread.
- QMC achieves better uniformity in $[0, 1]^d$, reducing clustering and gaps.

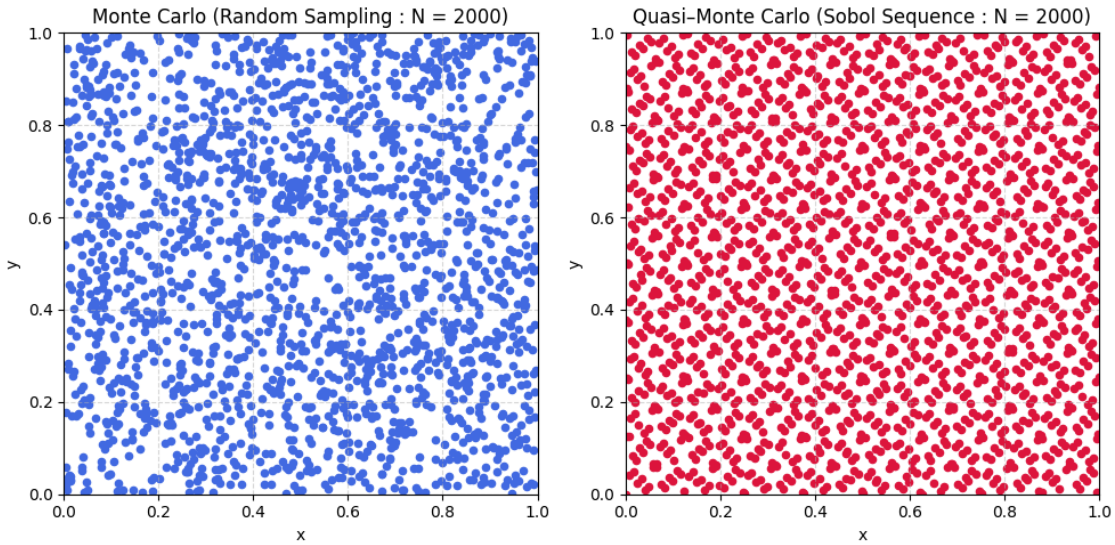


Figure: Comparison of MC (left) and QMC (right) sampling distributions.

Monte Carlo Estimation of π in 2D

- Randomly sample points in $\mathcal{C}_2 = [-1, 1]^2$.
- Count points inside the unit circle $x^2 + y^2 \leq 1$.
- The ratio of inside to total points gives:

$$\text{Fraction inside} = \frac{\text{Area of Circle}}{\text{Area of Square}} = \frac{\pi}{4}$$

$$\Rightarrow \pi = 4 \times \mathit{Fractioninside} \boxed{\approx 4 \times \frac{M_{\text{inside}}}{N_{\text{total}}}}$$

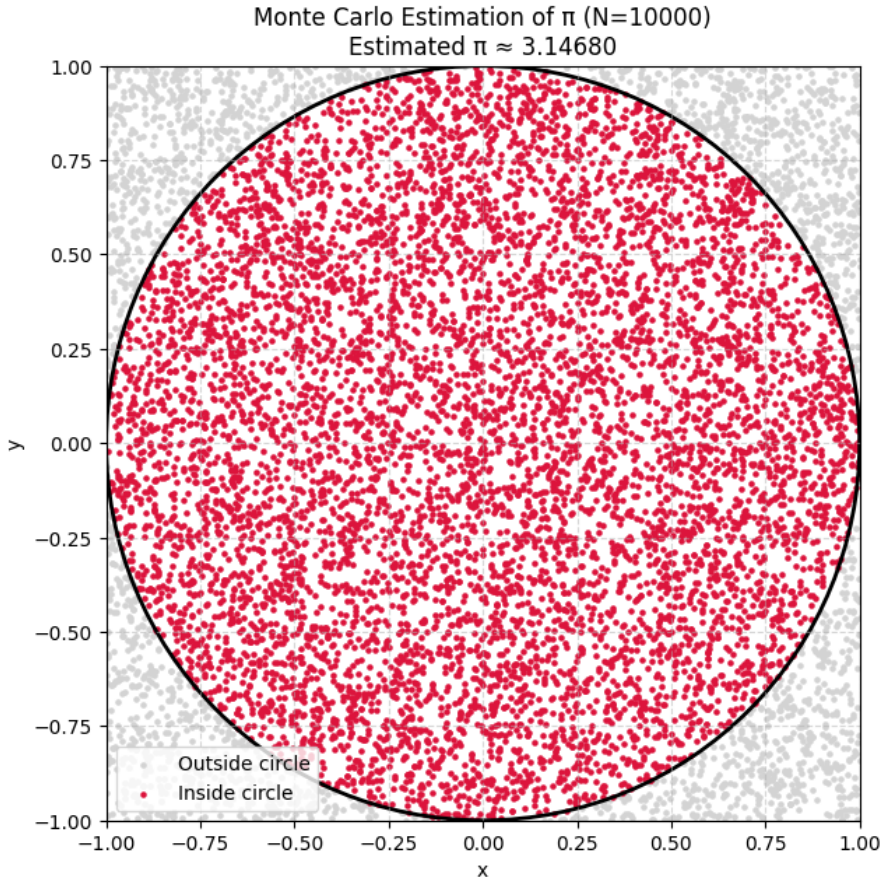


Figure: π estimation ($N = 10{,}000$). Red: inside circle, Gray: outside.

Monte Carlo Estimation of π in Higher Dimensions

- π is then approximated using the hypersphere-to-hypercube ratio:

$$\pi \approx \left(2^d \, \text{frac}_{\text{inside}} \, \Gamma\left(\frac{d}{2}+1\right)\right)^{\frac{2}{d}}$$

- As dimensionality d increases, fewer points fall inside the hypersphere , leading to slower convergence (*curse of dimensionality*).

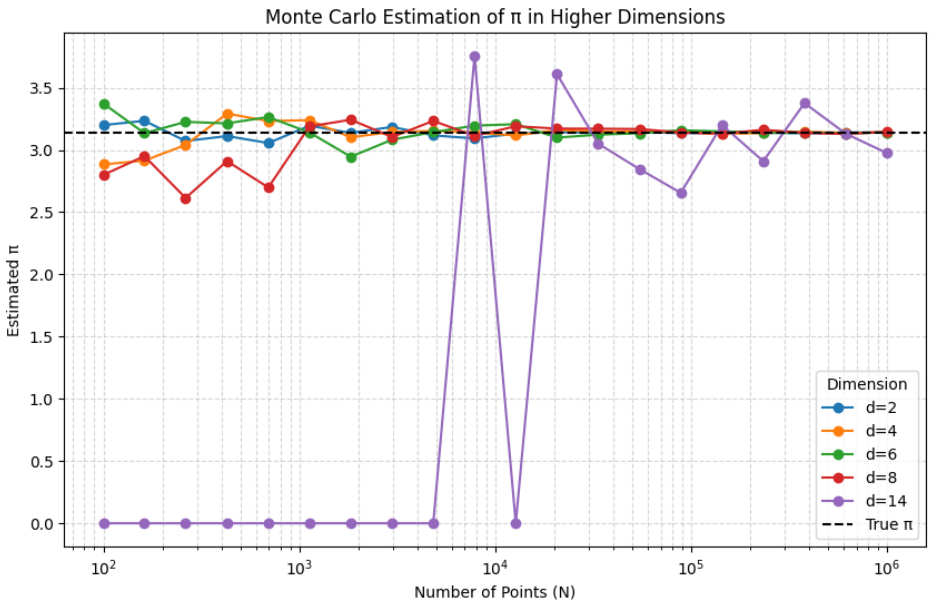


Figure: Convergence of π estimation for dimensions $\{2, 4, 6, 8, 14\}$. Accuracy decreases with higher d.

Variance Reduction in Monte Carlo π Estimation

- Variance reduction improves convergence without increasing sample size.
- Techniques used:
  - **Antithetic Variables (AV):** Uses negatively correlated samples to cancel noise.
  - **Control Variates (CV):** Exploits known expectations to reduce variance.
  - **Quasi-Monte Carlo (QMC):** Replaces randomness with low-discrepancy sequences.
- QMC consistently achieves the lowest error and best asymptotic rate.

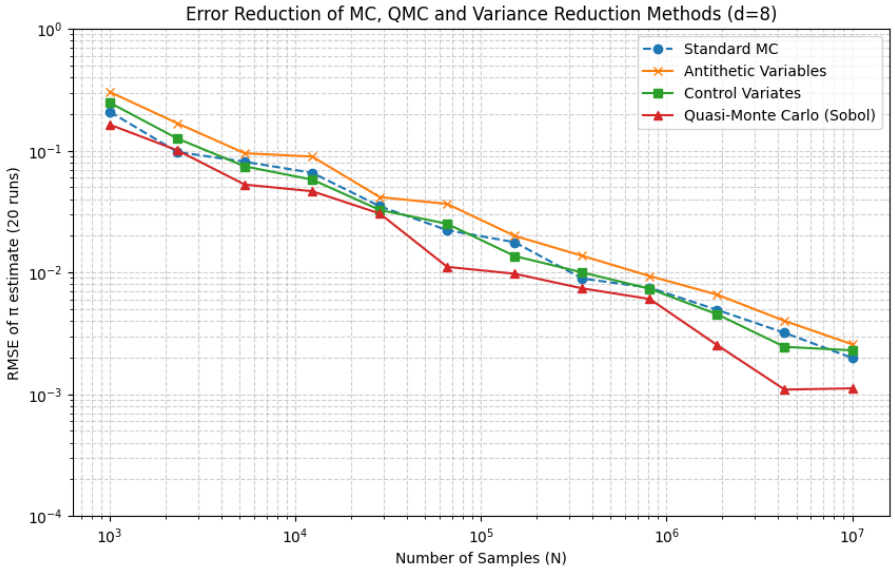


Figure: RMSE of π estimation for $d = 8$ across methods (20 runs). QMC achieves better error reduction approach.

Application to Continuous Integration Problem ($d = 4$)

Problem Setup:

$$f(\mathbf{x}) = e^{-\sum_{i=1}^{4} x_i^2}, \quad \mathbf{x} \in [-1, 1]^4, \quad I = \int_{\mathcal{C}_4} f(\mathbf{x})\, d\mathbf{x}$$

- **CV:** Pronounced and sustained reduction in error; effectively lowers initial variance (σ) for smooth integrals.
- **QMC:** Steepest convergence and lowest RMSE.
- Both methods show strong, noticeable error reduction compared to simpler problems like π estimation.

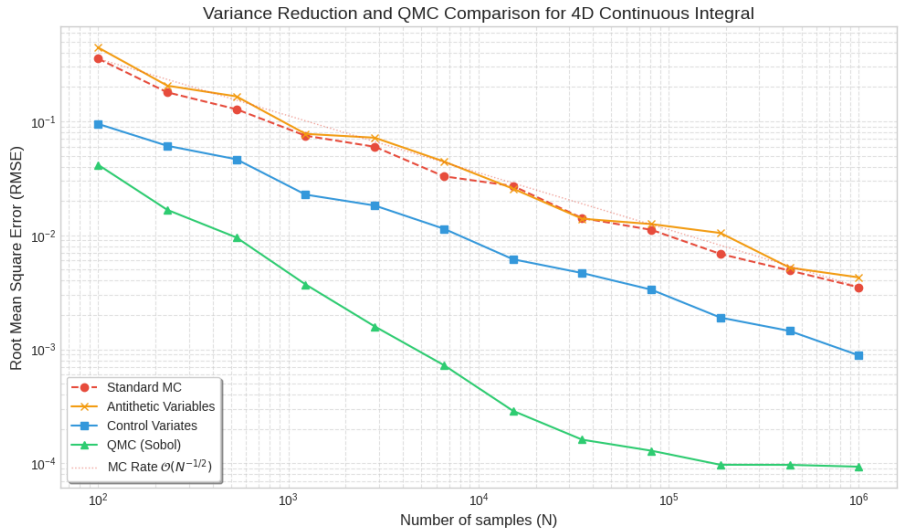
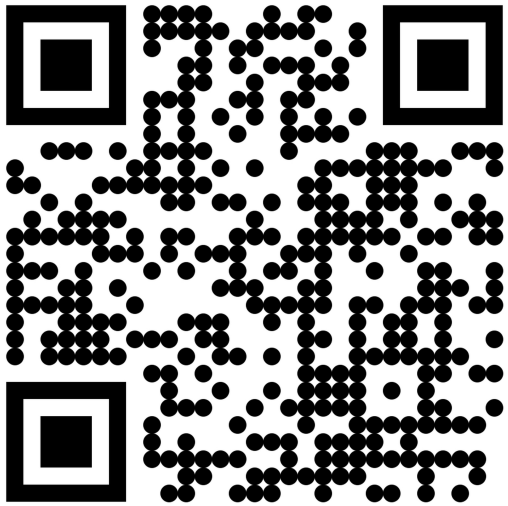


Figure: RMSE Convergence for the 4D Continuous Integral. CV and QMC show significant error reduction.

Implementation and Collaboration

- All experiments implemented in Python.
- Collaboration: **Md Jahid Hassan** and **Mohammad Amin**.
- Scan the QR code to access the experimental notebooks.



What was clear?

- **MC as expectation:** Clear link $\int f(x)\,dx = E[f(X)]$ makes integration an averaging problem.
- **Error rate:** LLN/CLT justify MC error $O(N^{-1/2})$; rate is dimension-agnostic.
- **Comparative insight:** Contrast with grid quadrature explains MC viability in high dimensions.

What was not clear?

- **Dense measure theory:** Hardy–Krause variation and Brownian sheet measure lack intuitive explanation.
- **Notation load:** Multidimensional formulas have minimal interpretation.

Is the information laid out in a good order?

- **Strength:** MC basics → sampling → variance reduction → QMC → application.
- **Break in flow:** Abrupt jump from QMC theory to gas-dynamics application.
- **Placement issue:** Sobol'/Halton/Niederreiter generators appear too late.

Are there any omissions?

- **No runtime evidence:** Only asymptotic rates; no error–vs–time plots or timing tables.
- **No confidence intervals (CIs):** Variance derived, but uncertainty for MC estimators not reported.

Are the arguments strong?

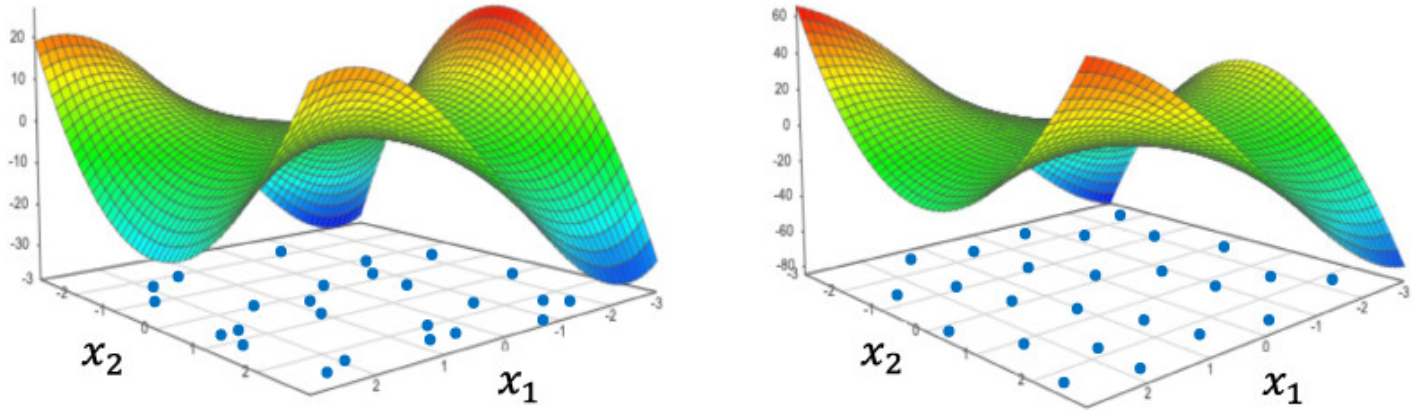
- **Theory:** Strong — Koksma–Hlawka + low discrepancy $\Rightarrow$ $O((\log N)^d/N)$ deterministic rate.
- **Evidence:** Figures show QMC's superior uniformity/convergence visually.

What could be improved?

- **Interpretive scaffolding:** Insert brief summaries after derivations; add a notation/glossary box.
- **Reordering:** Place QMC generators/software immediately after §5.
- **Practice-oriented results:** Error–vs–time plots, CIs for MC

Envisioning the Future

- Projected trends: AI/ML integration, quantum hybrids, high-dim scalability.
- Open problems evolving: Metropolis QMC, diffusion limits, ultra-high dims.



Monte Carlo:
A random sampling scheme

Quasi-Monte Carlo:
A deterministic sampling scheme

Figure: Visualization of low-discrepancy sampling in future high-dim spaces.

Future Academic Trajectories

- AI-guided QMC: LLM evolution for point sets (e.g., 2030 benchmarks).
- Neural flows in TQMC: Tailored for implicit densities, reducing errors 10-20%.
- RQMC convergence: Owen's bounds enabling ultra-high-dim UQ by 2035.

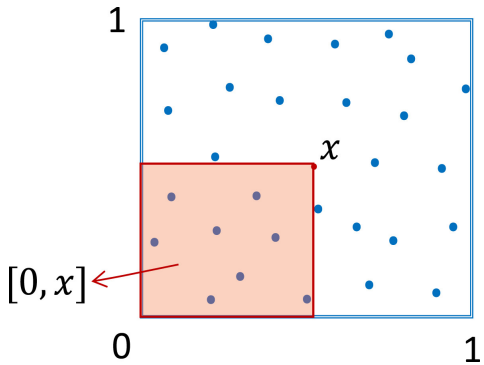


Figure: Projected convergence graphs for AI-enhanced QMC.

Emerging Industrial Applications

- Finance: AI-QMC for real-time risk in volatile markets (post-2030).
- Engineering: QMC in pavement/climate simulations, hybrid with ML for reliability.
- Challenges: Hybrids for discontinuities, quantum for exponential speedups.

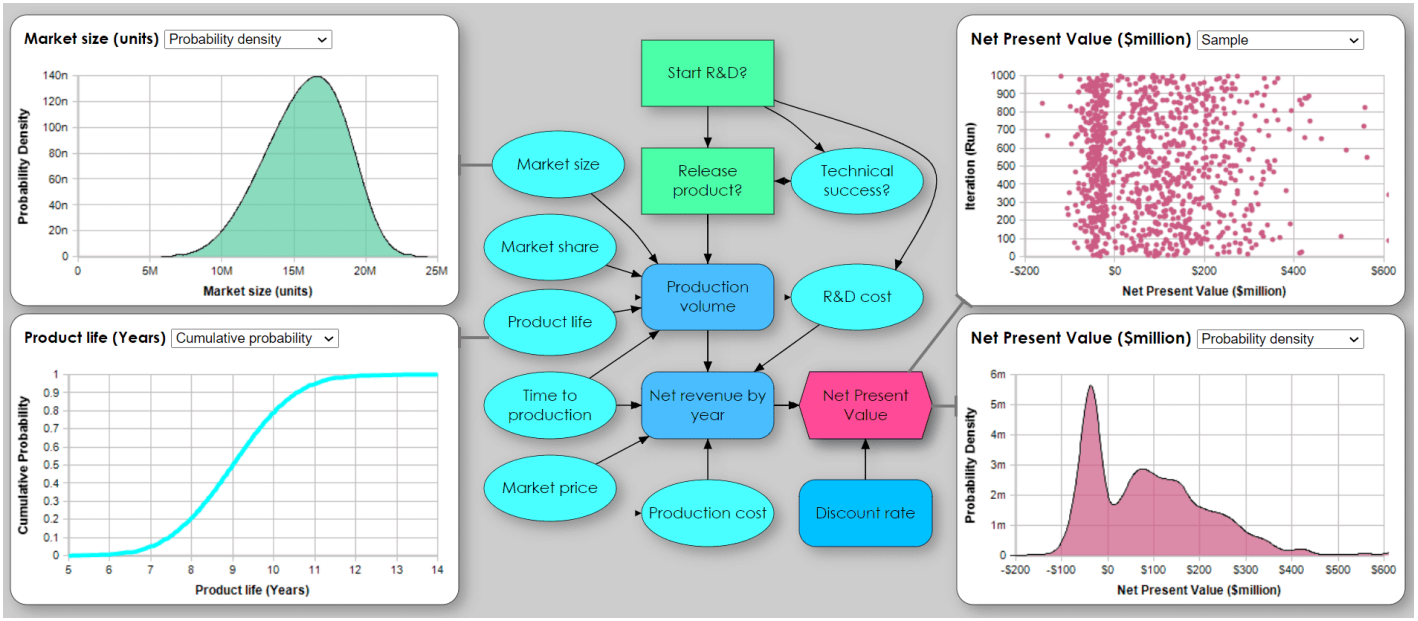


Figure: Simulation of future QMC in engineering uncertainty.

Evolution in Future Textbooks and Resources

- From theory to AI tools: Python/ML examples in MCQMC by 2030.
- Quantum integrations: Chapters on QQMC for many-body physics.
- Hands-on: Repositories for adaptive sequences in high dims.

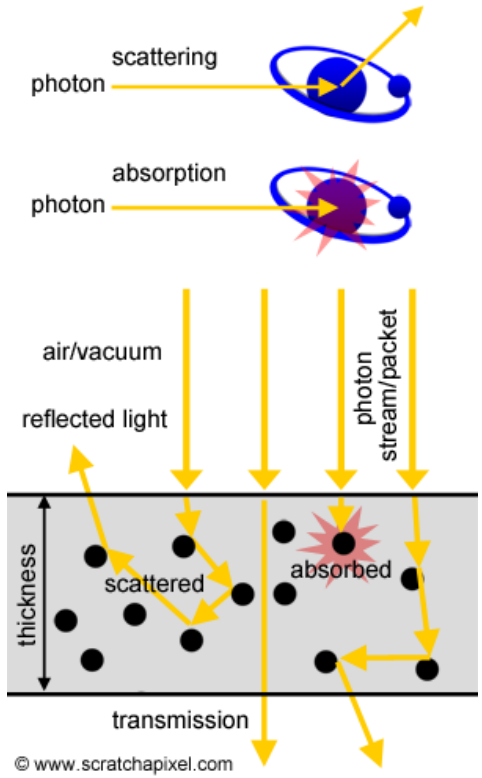


Figure: Diagram of Monte Carlo evolution toward quantum hybrids.

Open Problems and Horizons

- QMC-Metropolis: Unbiased hybrids for MCMC by 2040.
- Diffusion limits: SHDMC for 2D materials, scaling to 4D.
- High dims: Combinatorial discrepancy for tractable QMC.
- Legacy: Balanced innovation, empathetic to limitations.

Thank You!

Questions?