Lecture 7: Approximating Expectations (Quadrature & Monte Carlo Integration)

Yasuyuki Sawada, Yaolang Zhong

University of Tokyo
yaolang.zhong@e.u-tokyo.ac.jp

December 3, 2025

Recap: Markov Decision Process (MDP)

- An MDP is defined by the tuple

$$(\mathcal{S}, \mathcal{A}, P, r, \beta)$$

where:

  - State space $\mathcal{S}$: Possible system states.
  - Action space $\mathcal{A}(s)$: Feasible actions when in state s.
  - Transition kernel $P_t(s' \mid s, a)$: Probability of moving from s to s' given action a.
  - Reward function $r_t(s, a)$: Instantaneous payoff from taking action a in state s. (Sometimes expressed as a *cost* $-r(s, a)$.)
  - Discount factor $\beta \in (0, 1)$: Weights future rewards relative to current ones.

- Objective: Choose a policy $\pi : \mathcal{S} \to \mathcal{A}$ to maximize expected discounted rewards:

$$\mathbb{E}_\pi \left[\sum_{t=0}^{T} \beta^t r_t(s_t, a_t) + \beta^T R_T(s_T) \right].$$

Recap: Algorithms and Key Equations

- Value Function Iteration (VFI):

$$V^{(k+1)}(s) = \max_{a \in \mathcal{A}(s)} \left\{ r(s,a) + \beta \, \mathbb{E}\left[V^{(k)}(S') \mid s, a \right] \right\}$$

- Policy Function Iteration (PFI):
  - Policy Evaluation:

$$V^{(k)}(s) = r\big(s, \pi^{(k)}(s)\big) + \beta \, \mathbb{E}\left[V^{(k)}(S') \mid s, \pi^{(k)}(s) \right]$$

  - Policy Improvement:

$$\pi^{(k+1)}(s) \in \arg\max_{a \in \mathcal{A}(s)} \left\{ r(s,a) + \beta \, \mathbb{E}\left[V^{(k)}(S') \mid s, a \right] \right\}$$

- Time Iteration (TI) / Endogenous Grid Method (EGM):
  (consumption–saving problem as example)

$$u_c(c_t^{(k+1)}) = \beta \, \mathbb{E}\left[u_c(c_{t+1}^{(k)}) \, (1 + r_{t+1}) \right]$$

Approximating the Expectation Operator $\mathbb{E}[\cdot]$

In dynamic programming, we must evaluate the expected continuation value:

$$\mathbb{E}[V(S') \mid s, a] = \int V(s')\, P(ds' \mid s, a).$$

Analytic evaluation is rarely possible. We rely on two numerical strategies:

1. Deterministic (Quadrature)

$$\int f(x)p(x)dx \approx \sum_{i=1}^{N} w_i f(x_i)$$

- Requires: Explicit PDF $p(s'|s,a)$.
- Pros: Very accurate (exponential convergence) for smooth functions.
- Cons: Suffers from Curse of Dimensionality (N^d).

2. Stochastic (Monte Carlo)

$$\mathbb{E}[f(x)] \approx \frac{1}{M} \sum_{m=1}^{M} f(x_m)$$

- Requires: Ability to *sample* from P.
- Pros: Error independent of dimension ($O(M^{-1/2})$); works without explicit PDF.
- Cons: Slow convergence (requires many samples).

Which Method to Use? Three Scenarios

The choice depends on our *knowledge of the transition kernel* $P(s'|s,a)$ and the *dimensionality* d:

Case 1: Closed-Form Density $p(s'|s,a)$ Available

- We know the formula for $p(s'|s,a)$ (e.g., AR(1) with Gaussian shocks).
- Low dimension ($d \leqslant 3$): Use Quadrature. It is far more efficient.
- High dimension ($d > 4$): Quadrature becomes infeasible (N^d nodes). Must switch to Monte Carlo or Sparse Grids.

Case 2: Simulator Available (Generative Model)

- No closed-form density $p(s'|s,a)$, but we can simulate s' given any (s,a).
- Example: Complex structural models, or bootstrap resampling from data.
- Method: Monte Carlo. We cannot use quadrature because we cannot evaluate weights w_i based on $p(x_i)$.

Deterministic Approximation: Quadrature

Quadrature approximates the expectation integral by a weighted sum of function evaluations at specific discrete points:

$$\mathbb{E}[f(x)] = \int f(x)\, p(x)\, dx \;\approx\; \sum_{i=1}^{N} w_i\, f(x_i).$$

- Interpretation:
  - choose a small set of representative points (nodes) x_i,
  - assign probability mass (weights) w_i,
  - evaluate $f(\cdot)$ only at those nodes.
- The core problem is how to optimally choose the nodes x_i and weights w_i to minimize approximation error for a given class of functions (e.g., polynomials).

1. Gauss–Hermite Quadrature (The Macro Workhorse)

- Mathematical Basis: Designed for the specific integral $\int_{-\infty}^{\infty} g(x)e^{-x^2}dx$.
- Standard Output (e.g., from Python/Matlab):
  - Input: N.
  - Output: Nodes x_i and weights w_i that satisfy $\sum w_i g(x_i) \approx \int g(x)e^{-x^2}dx$.
- Recipe for $\mathcal{N}(\mu, \sigma^2)$:
  - Goal: Compute $\mathbb{E}[f(y)] = \int f(y)\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y-\mu)^2}{2\sigma^2}}dy$.
  - Transformation: Define $x = \frac{y-\mu}{\sqrt{2}\sigma}$ so that $y = \sqrt{2}\sigma x + \mu$.
  - Algorithm:
    1. Get standard GH nodes x_i and weights w_i (inputs: N).
    2. Transform nodes back to y: $y_i = \sqrt{2}\sigma x_i + \mu$.
    3. Approximate: $\mathbb{E}[f(y)] \approx \frac{1}{\sqrt{\pi}}\sum_{i=1}^{N} w_i f(y_i)$.

2. Gauss–Legendre Quadrature (Bounded Shocks)

- Mathematical Basis: Designed for integrals on $[-1, 1]$ with uniform weight.

$$\int_{-1}^{1} g(x)dx \approx \sum_{i=1}^{N} w_i g(x_i)$$

- Recipe for Integral on $[a, b]$ (e.g., Uniform Shock):
  - Transformation: Linear map $y = \frac{b-a}{2}x + \frac{a+b}{2}$.
  - Algorithm:
    1. Get standard GL nodes x_i and weights w_i (inputs: N).
    2. Transform nodes: $y_i = \frac{b-a}{2}x_i + \frac{a+b}{2}$.
    3. Approximate: $\int_a^b f(y)dy \approx \frac{b-a}{2} \sum_{i=1}^{N} w_i f(y_i)$.

3. Handling Higher Dimensions: The Curse and Smolyak's Solution

- The "Curse of Dimensionality": Full tensor grids grow exponentially (N^d).
  - $N = 5, d = 2 \Rightarrow 25$ points (Easy).
  - $N = 5, d = 6 \Rightarrow 15,625$ points (Slow).
  - $N = 5, d = 10 \Rightarrow \sim 9.7$ million points (Impossible).
- Solution: Smolyak's algorithm (sparse grids) selects a "smart subset" of grid points.
  - Insight: Most accuracy comes from individual dimensions and low-order interactions. Points representing complex interactions of all d variables (corners) contribute little.
  - Result: Cost grows polynomially ($\sim d^k$) rather than exponentially, making $d \approx 10$ feasible.

3. Sparse Grid Integration: Recipe

- Context: Compute $\int_S f(\mathbf{s})\, d\mathbf{s}$ over $S = [a_1, b_1] \times \cdots \times [a_d, b_d]$.
- Algorithm Steps:
  1. Generate d-dimensional sparse grid nodes $\mathbf{x}_j$ and weights w_j (on reference domain $[-1, 1]^d$).
  2. Transform each node $\mathbf{x}_j$ to the actual domain $\mathbf{s}_j$:

$$s_{j,n} = \frac{b_n - a_n}{2} x_{j,n} + \frac{a_n + b_n}{2} \quad \text{for } n = 1, \ldots, d$$

  3. Evaluate and sum: Approx $= \sum_{j=1}^{M} w_j\, f(\mathbf{s}_j)$.
  4. Scale by Jacobian (change of variables):

$$\text{Final Result} = \text{Approx} \times \prod_{n=1}^{d} \frac{b_n - a_n}{2}$$

4. Monte Carlo Integration: The Basic Idea

Recall our goal: approximate $\mathbb{E}[f(x)] = \int f(x)p(x)dx$.

Monte Carlo Integration: Draw i.i.d. samples $x_1, \ldots, x_M$ from $p(x)$, then:

$$\mathbb{E}[f(x)] \approx \hat{\mu}_M = \frac{1}{M} \sum_{m=1}^{M} f(x_m)$$

Why it works:

- Law of Large Numbers: $\hat{\mu}_M \to \mathbb{E}[f(x)]$ as $M \to \infty$ (almost surely).
- Central Limit Theorem: $\sqrt{M}(\hat{\mu}_M - \mu) \xrightarrow{d} N(0, \sigma^2)$, where $\sigma^2 = \mathsf{Var}(f(x))$.
- Key property: Error rate $O(1/\sqrt{M})$ is independent of dimension d.

Application to Bellman Equation: Given $V^{(k)}$, approximate $\mathbb{E}[V^{(k)}(S') \mid s]$ by:

$$\text{Draw } s'_1, \ldots, s'_M \sim P(\cdot \mid s, a), \quad \text{compute } \frac{1}{M} \sum_{m=1}^{M} V^{(k)}(s'_m)$$

4. Monte Carlo: The Slow Convergence Problem

The $O(1/\sqrt{M})$ convergence rate is slow:

Samples M	Relative Error	Improvement
100	$\sim 10\%$	baseline
10,000	$\sim 1\%$	100× samples for 10× accuracy
1,000,000	$\sim 0.1\%$	10,000× samples for 100× accuracy

Compare to quadrature:

- Gauss quadrature with N nodes is exact for polynomials up to degree $2N - 1$.
- For smooth functions, error $\sim O(e^{-cN})$ (exponential convergence!).
- But: cost grows as N^d in d dimensions (curse of dimensionality).

Rule of thumb: For $d \leqslant 3$, quadrature is usually faster. For $d > 5$, MC wins.

4. Variance Reduction: The Key to Faster MC

The MC error depends on $\text{Var}(f(x))$. Can we reduce it? Recall: Std Error $= \frac{\sigma}{\sqrt{M}}$, where $\sigma = \sqrt{\text{Var}(f(x))}$.

Two ways to reduce error:

1. Increase M (brute force—expensive)
2. Reduce σ (smart sampling—free lunch!)

Main variance reduction techniques:

- Antithetic variates: Use negatively correlated samples.
- Control variates: Subtract a known-mean random variable.
- Importance sampling: Sample from a better distribution.
- Quasi-Monte Carlo: Replace random samples with low-discrepancy sequences.

4. Antithetic Variates

Idea: If x and x' are negatively correlated, then $\text{Var}(\frac{f(x)+f(x')}{2}) < \text{Var}(f(x))$.

For symmetric distributions (e.g., $\varepsilon \sim N(0, \sigma^2)$):

- Draw $\varepsilon_1, \ldots, \varepsilon_{M/2}$ from $N(0, \sigma^2)$.
- Use pairs: $(\varepsilon_m, -\varepsilon_m)$ for $m = 1, \ldots, M/2$.
- Estimate: $\hat{\mu} = \frac{1}{M} \sum_{m=1}^{M/2} [f(\varepsilon_m) + f(-\varepsilon_m)]$.

Why it works:

- ε and $-\varepsilon$ are perfectly negatively correlated.
- If f is monotonic, $f(\varepsilon)$ and $f(-\varepsilon)$ are also negatively correlated.
- Variance reduction can be up to 50% (often 20–40% in practice).

Cost: Free! Same number of function evaluations, just pair them smartly.

4. Control Variates

Idea: Use a correlated variable with known mean to reduce variance.

Suppose we want $\mathbb{E}[f(x)]$ and we know $\mathbb{E}[g(x)] = \mu_g$ exactly.

Control variate estimator:

$$\hat{\mu}_{CV} = \frac{1}{M}\sum_{m=1}^{M} f(x_m) - c\left(\frac{1}{M}\sum_{m=1}^{M} g(x_m) - \mu_g\right)$$

Optimal $c^* = \frac{\text{Cov}(f,g)}{\text{Var}(g)}$ minimizes variance.

Economics example:

- Want: $\mathbb{E}[V(a', z')]$ where $z' = \rho z + \varepsilon$.
- Know: $\mathbb{E}[\varepsilon] = 0$ exactly.
- Control variate: Use $g = \varepsilon$ to correct for sampling error in ε.

4. Importance Sampling: Using $P(\cdot)$ Smartly

Idea: Sample from a "better" distribution $q(x)$ instead of $p(x)$.

Key identity:

$$\mathbb{E}_p[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = \mathbb{E}_q\left[f(x)\frac{p(x)}{q(x)}\right]$$

Importance sampling estimator:

$$\hat{\mu}_{IS} = \frac{1}{M}\sum_{m=1}^{M} f(x_m)\cdot \underbrace{\frac{p(x_m)}{q(x_m)}}_{\text{importance weight}}, \qquad x_m \sim q(x)$$

When is this useful?

- When $f(x)\cdot p(x)$ is concentrated in regions where $p(x)$ is small.
- Choose $q(x)$ to sample more from "important" regions.
- Optimal $q^*(x) \propto |f(x)| \cdot p(x)$ (but often unknown).

4. Importance Sampling: Application to Rare Events

Example: Estimating probability of extreme losses.

Want: $P(Z > k) = \mathbb{E}[\mathbf{1}_{Z>k}]$ where $Z \sim N(0,1)$ and $k = 4$.

Problem with naive MC:

- $P(Z > 4) \approx 3 \times 10^{-5}$ (very rare).
- Need $\sim 10^6$ samples to see even ~ 30 events.
- Estimate has huge relative error.

Importance sampling solution:

- Sample $Z \sim N(k, 1)$ (shifted to the tail).
- Reweight: $w(z) = \frac{\phi(z)}{\phi(z-k)} = e^{-kz+k^2/2}$.
- Now most samples are in the region of interest!

Economics application: Pricing deep out-of-the-money options, stress testing.

4. Quasi-Monte Carlo: Deterministic "Random" Numbers

Idea: Replace random samples with low-discrepancy sequences that fill the space more evenly.

Random (MC)	Low-discrepancy (QMC)
Clumpy, gaps	Even coverage
Error $O(1/\sqrt{M})$	Error $O((\log M)^d/M)$

Common sequences:

- Halton sequence
- Sobol sequence (most popular in finance/economics)
- Niederreiter sequence

When to use:

- Works best for smooth integrands in moderate dimensions ($d \lesssim 20$).
- Very popular in option pricing, portfolio optimization.
- Easy to implement: just replace `rand()` with `sobol()`.