Longstaff-Schwartz Least Squares Monte Carlo Method

Quantitative Finance Tutorial

May 26, 2025

Contents

1 Introduction

The Longstaff-Schwartz (2001) Least Squares Monte Carlo (LSM) method provides a practical numerical solution for pricing American-style options. Unlike European options, American options allow for early exercise, complicating their valuation.

2 Theoretical Foundation

2.1 Dynamic Programming Framework

Consider an American option with underlying asset price S_t, payoff function $h(S_t)$, and risk-neutral dynamics given by:

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{1}$$

The value of an American option at time t is the solution to:

$$V_t(S_t) = \max\left(h(S_t), e^{-r\Delta t}\mathbb{E}[V_{t+\Delta t}(S_{t+\Delta t})|S_t]\right) \tag{2}$$

2.2 Conditional Expectation and Regression

To approximate the conditional expectation, Longstaff and Schwartz propose a regression approach using basis functions $\psi_j(S_t)$:

$$\mathbb{E}[V_{t+\Delta t}(S_{t+\Delta t})|S_t] \approx \sum_{j=1}^{M} \alpha_j \psi_j(S_t) \tag{3}$$

Regression determines coefficients α_j by minimizing least squares error.

2.3 Basis Functions

Longstaff-Schwartz commonly uses simple polynomial bases:

$$\psi_j(S) = S^{j-1}, \quad j = 1, 2, ..., M \tag{4}$$

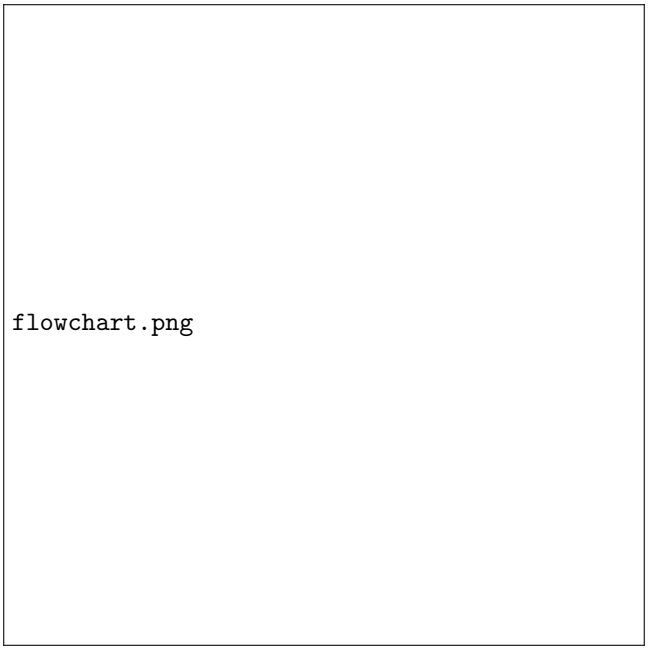
3 Implementation Guide

3.1 Algorithm Steps

The Least Squares Monte Carlo algorithm proceeds as follows:

1. Simulate paths $S_t^{(i)}$ for $i = 1, ..., N$ using Geometric Brownian Motion.
2. Set terminal payoff $V_T^{(i)} = h(S_T^{(i)})$.
3. Iteratively calculate continuation values via regression at each timestep.
4. Determine the optimal exercise boundary.

3.2 Flowchart



4 Python Implementation

Using parameters from Longstaff and Schwartz (2001), Table 1:

```
import numpy as np

# Parameters
S0, K, r, sigma, T = 36, 40, 0.06, 0.2, 1
M, I = 50, 10000
dt = T / M

# Path generation
np.random.seed(0)
S = np.zeros((M + 1, I))
S[0] = S0
for t in range(1, M + 1):
    Z = np.random.standard_normal(I)
    S[t] = S[t-1] * np.exp((r - 0.5 * sigma**2)*dt + sigma*np.sqrt(dt)*Z)

# Payoff calculation
h = np.maximum(K - S, 0)
V = h[-1]
```

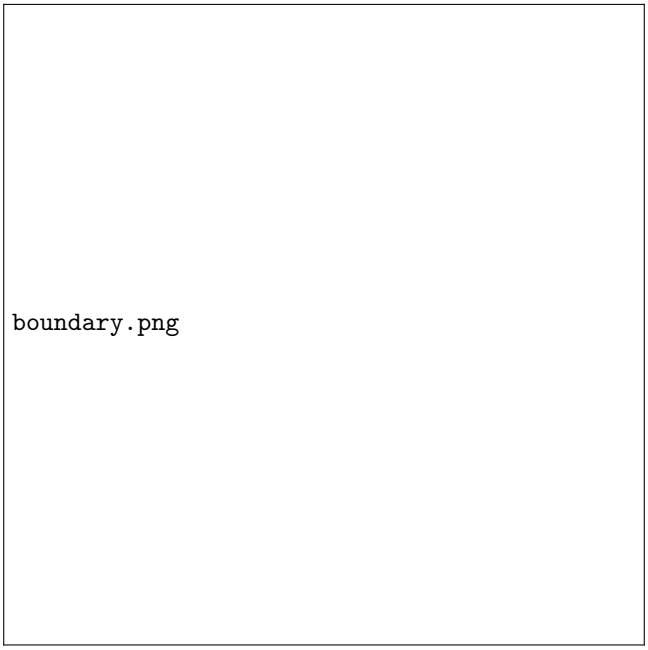
```
# Backward induction
for t in range(M - 1, 0, -1):
    itm = np.where(h[t] > 0)
    reg = np.polyfit(S[t][itm], V[itm]*np.exp(-r*dt), 2)
    continuation = np.polyval(reg, S[t][itm])
    exercise = h[t][itm]
    V[itm] = np.where(exercise > continuation, exercise, V[itm]*np.exp(-r*dt))

# Valuation
price = np.mean(V)*np.exp(-r*dt)
print(f'Option price: {price:.4f}')
```

5 Intermediate Results

Example regression output (coefficients):

$$V(S) \approx \alpha_0 + \alpha_1 S + \alpha_2 S^2 \tag{5}$$

Exercise boundary visualization:



6 Comparison with Alternative Methods

The LSM method is compared with Tsitsiklis-van Roy (TVR) and traditional binomial models.

7 Error Sources and Convergence

Main errors arise from:

- Path discretization
- Basis function selection
- Monte Carlo sampling

Convergence analysis:

8 Conclusion

The LSM method provides an effective solution for American options. Optimal regression choice and sufficient simulation paths yield high accuracy, with convergence guaranteed under proper conditions.