
Parallel Longstaff–Schwartz Pricing of American Options in Haskell

Project Proposal: lsm-american
COMS 4995 Parallel Functional Programming, Fall 2026

Yilu Yang (yy3626)
Columbia University
yy3626@columbia.edu

1 Background

Options. An *option* is a contract that gives its holder the right, but not the obligation, to buy or sell an underlying asset at a fixed *strike price* K on a future date; this proposal considers options on a single stock, so the asset is one share of a stock. A *call* is the right to buy, a *put* the right to sell. Since the holder exercises only when it is profitable, the *payoff* at exercise is $g(S) = \max(S - K, 0)$ for a call and $g(S) = \max(K - S, 0)$ for a put, where S is the stock price at exercise. The price of an option is the expected value of its discounted payoff under the risk-neutral measure, that is, with the expected growth rate of the stock set equal to the risk-free interest rate r , which is also the discount rate. The standard model of the stock price is geometric Brownian motion, under which the stock's logarithmic return over each period is an independent normal random variable; its standard deviation per unit time is the *volatility* σ , taken as constant. The stock is taken to pay no dividends.

European options. A European option can be exercised only on its final date, the *maturity* T . Under geometric Brownian motion, its price has a closed form, the Black–Scholes formula [1].

American options and early exercise. An American option can be exercised at any time up to T . Its holder faces a decision at every time: exercise now and receive the immediate payoff $g(S)$, or continue to hold. The value of holding, the *continuation value*, depends on all later decisions, so the option price is the expected discounted payoff under the optimal exercise strategy, the rule that makes these decisions. For a put, the optimal strategy is a threshold rule: at each time there is a critical stock price below which immediate exercise is optimal, because the strike is received earlier and the remaining downside is bounded. The threshold varies with time and has no closed form, and neither does the option price. The American put is the standard test problem for early exercise, including in Longstaff and Schwartz [3]. For a call, early exercise is never optimal, since the strike is paid rather than received and the payoff is unbounded above; the American call is therefore worth exactly the European call and needs no new method.

Least-squares Monte Carlo. For an option on a single stock with constant volatility and interest rate, the stock price is the only *state variable*, that is, the only quantity whose current value the option price depends on. The price can then be computed by backward induction, working from maturity back to today, on a grid of stock prices. One such method is a *lattice method*, which restricts the stock price to a discrete set of values at a discrete set of times, such as the binomial tree of Cox, Ross, and Rubinstein [2]; another is *finite differences*, which discretize the partial differential equation that the option price satisfies. The grid grows exponentially with the number of state variables, so for options with several of them, for instance on several assets or with a stochastic volatility or interest rate, the price is instead estimated by Monte Carlo simulation. Many sample paths are generated and their discounted payoffs averaged, at a cost that grows only linearly with the number of state variables. Averaging the payoffs of simulated paths requires the payoff of each path to be known.

For an American option, the payoff of a path depends on the exercise strategy, which at every time compares the immediate payoff with the continuation value. Longstaff and Schwartz [3] estimate the continuation value by regression. At each date of the simulation, the realized continuation payoffs of the paths whose immediate payoff is positive, that is, the payoffs those paths receive later, discounted to that date, are regressed by least squares on a low-order polynomial in the current stock price; the fitted value is then compared with the immediate payoff. Their least-squares Monte Carlo method, LSM, is the standard method for American options whenever lattice and finite-difference methods are impractical.

Proposed work. This project will implement LSM for the American put on a single stock in Haskell, first sequentially and then in parallel, and measure how the speedup scales with the number of cores. The single-asset case has a reference solution, the binomial tree, so correctness can be verified. The same implementation extends to several stocks, the case in which LSM is the standard method. The paths then become multi-dimensional, and the regression polynomial needs more terms, its basis functions. The American call serves as a test case, since its price is given by the Black–Scholes formula.

2 Algorithm

Inputs: today's stock price S_0 , strike K , risk-free interest rate r , volatility σ , maturity T , M exercise dates, equally spaced $\Delta t = T/M$ apart, and the number of simulated paths N . The algorithm restricts exercise to these dates, so it prices a *Bermudan* option, which can be exercised only on a fixed set of dates. Its price converges to the American price as M grows; Longstaff and Schwartz use 50 dates per year. A path is *in the money* at a date if its immediate payoff $g(S)$ is positive.

1. **Forward simulation.** Generate N sample paths of the stock price over the M dates, where $S_{t,i}$ denotes the price of path i at date t :

$$S_{t+1,i} = S_{t,i} \exp \left((r - \frac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t} Z \right),$$

with $S_{0,i} = S_0$ and Z a standard normal random variable drawn independently for every path and date, and store the prices as an $M \times N$ array.

2. **Backward induction.** Proceed from the last date to the first. Maintain for each path i a value V_i : the payoff that path receives under the decisions made so far, discounted to the current date. At maturity, $V_i = g(S_{M,i})$. Then for $t = M - 1$ down to 1:
 - (a) *Discount*: multiply every V_i by $e^{-r\Delta t}$, moving the value one date earlier.
 - (b) *Regress*: over the in-the-money paths, fit V_i by least squares as a linear combination of the basis functions, here $1, x, x^2$ with $x = S_{t,i}/K$. The three coefficients come from the *normal equations*, the linear system obtained by setting the derivatives of the squared error with respect to the coefficients to zero, here 3×3 ; its entries are the sums over the in-the-money paths of x^k for $k = 0, \dots, 4$ and of $x^k V_i$ for $k = 0, 1, 2$, 8 sums in all. The fitted value is the estimate of the continuation value.
 - (c) *Decide*: for each in-the-money path, if the immediate payoff $g(S_{t,i})$ exceeds the estimated continuation value, the path is exercised: set V_i to the immediate payoff; otherwise keep V_i .
3. **Price.** Discount once more and average: the price is $e^{-r\Delta t}$ times the mean of the V_i , reported with its standard error.

The cost is $O(NM)$. Per path and date, the forward simulation performs a large amount of arithmetic over little memory traffic: a normal random variable by the Box–Muller transform, which turns two uniform random numbers into two normal ones, and one exponential, against one price read and one written. The backward induction performs little arithmetic over more traffic: a few multiplications and additions against reads of the price and of V_i and a write of V_i . The *arithmetic intensity*, operations per byte of memory traffic, is therefore high in the forward simulation and low in the backward induction.

This is more than an embarrassingly parallel Monte Carlo, that is, one whose work units are independent, and more than the k-means loop, whose iteration has the shape of one date of the backward induction: the two phases differ in arithmetic intensity, and the subset of paths that each date

regresses over is data-dependent. The backward induction is a chain of dependent dates: the regression at date t needs all N values from date $t + 1$. The work at each date is a parallel reduction (the sums of the normal equations, with the discount folded into the sums over V_i), a serial 3×3 solve, and a parallel map (the discount and the exercise decision). The program therefore alternates parallel and serial sections M times. The serial fraction, the share of the running time that cannot be parallelized, which Amdahl’s law relates to the speedup, is set by N : the serial work of a date is fixed, while its parallel work grows with the number of paths.

3 Parallelization Plan

1. **Sequential baseline.** The project starts from a sequential implementation and improves it before adding parallelism. The paths and the values are stored in strict, unboxed arrays, and the inner loops use strict accumulators (bang patterns and `foldl`’), so that no thunks accumulate. The random number generator is splittable, so that every chunk of paths (a block of consecutive paths processed as one unit) has its own stream, and the partial sums of the chunks are combined in a fixed order, so that the result is reproducible for any number of cores.
2. **Forward simulation.** The forward simulation partitions the paths into chunks and evaluates the chunks in parallel with Strategies [4]. It is independent per path and has a high arithmetic intensity, so near-linear speedup is expected.
3. **Backward induction.** At each date, the backward induction
 - (a) computes the 8 sums of the normal equations as a parallel map-reduce over chunks, each chunk accumulating its partial sums strictly before the partial sums are combined;
 - (b) solves the 3×3 system serially;
 - (c) applies the discount and the exercise decision as a parallel map over chunks.
 It uses Strategies first, then the Par monad for comparison.
4. **Design experiments.**
 - (a) Granularity: the number of chunks against speedup.
 - (b) Fusion: the exercise decision of date t and the sums of the normal equations of date $t - 1$ in one sweep, which saves one sweep over the arrays per date.
 - (c) Garbage collector: its settings.
 - (d) Representation: boxed versus unboxed arrays.
5. **Metrics.** Speedup from 1 to 8 cores, the parallel fraction estimated from Amdahl’s law, and ThreadScope timelines of the backward induction, which show each core’s activity over time; there the serial solve at every date appears as a serial gap between parallel sections.

4 Correctness Tests

- A European put priced from the same simulated paths (no early exercise) must match the Black–Scholes formula within three standard errors for every N in the test range, and the deviation from the formula across seeds must shrink as $1/\sqrt{N}$.
- The American call, priced with $g(S) = \max(S - K, 0)$, must match the Black–Scholes call price within three standard errors (see Background). This tests the regression and the exercise decision, since exercise before maturity lowers the call price below the formula.
- The American put is compared with the binomial tree of Cox, Ross, and Rubinstein [2] with 2000 steps and exercise restricted to the same M dates, which prices the same Bermudan option, and with Table 1 of Longstaff and Schwartz [3]: $K = 40$, $r = 0.06$, $\sigma \in \{0.2, 0.4\}$, $T \in \{1, 2\}$, $S_0 = 36, 38, \dots, 44$, and 50 exercise dates per year. It must agree with both to within the difference that Table 1 reports between its simulated and finite-difference values.
- For the same seed and chunk count, the parallel program must give exactly the sequential price on any number of cores.

5 Input Data

All input is synthetic: the inputs of the Algorithm section and a seed. N ranges from 10^4 (for tests) to 10^6 (for scaling), and M from 50 to 100, which gives problem sizes from small to large.

6 Scope

Minimum deliverable: a single-asset American put, sequential and parallelized with Strategies, the correctness tests and the metrics above. **Stretch goals**, only once the minimum is complete:

- the comparison with the Par monad;
- design experiments (b) to (d);
- a Repa version of the forward simulation and of the sweeps of the backward induction, since both are data-parallel operations over large unboxed arrays;
- a two-asset Bermudan max-call, whose payoff depends on the higher of two stock prices; it needs correlated paths by Cholesky factorization and more basis functions, which raises the arithmetic intensity.

Accelerate is not planned, since the metric is scaling across CPU cores.

References

- [1] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.
- [2] John C. Cox, Stephen A. Ross, and Mark Rubinstein. Option pricing: A simplified approach. *Journal of Financial Economics*, 7(3):229–263, 1979.
- [3] Francis A. Longstaff and Eduardo S. Schwartz. Valuing American options by simulation: A simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147, 2001.
- [4] Simon Marlow. *Parallel and Concurrent Programming in Haskell*. O’Reilly Media, Sebastopol, CA, 2013.