01 / EXECUTIVE SUMMARY

Executive summary

This project implements a compact Monte Carlo derivatives-pricing engine in Python. NumPy generates 10,000 risk-neutral Geometric Brownian Motion paths; terminal payoffs produce European call and put estimates; backward induction with least-squares continuation values extends the engine to American exercise.

The implementation is intentionally educational rather than production-grade. Its value is the complete chain from stochastic model to vectorized simulation, discounted payoff valuation, early-exercise logic, and an interactive Streamlit interface.

10,000	252	$10.28	$6.14
SIMULATED PATHS	TIME STEPS	EUROPEAN CALL	AMERICAN PUT (LSM)

What this demonstrates

- Applied understanding of risk-neutral valuation and lognormal asset-price dynamics.
- Vectorized NumPy implementation with explicit input validation and clear array semantics.
- European pricing through discounted expected payoff and American pricing through Longstaff-Schwartz regression.
- Independent numerical validation against Black-Scholes, put-call parity, and a binomial American put benchmark.

Bottom line

For the reference case S0 = K = $100, r = 5%, volatility = 20%, and T = 1 year, the Monte Carlo European call estimate is within its sampling uncertainty of the Black-Scholes value. The result supports implementation correctness, but it does not remove model risk: GBM assumes constant volatility, continuous trading, and lognormal returns.

Project scope

Component	Implementation	Purpose
Path generator	Exact discretization of GBM in log space	Simulate strictly positive stock prices
European pricer	Discounted mean terminal payoff	Estimate call and put fair values
American pricer	Longstaff-Schwartz, degree-2 polynomial	Approximate the early-exercise decision
Interface	Streamlit + Plotly	Expose parameters and visualize sample paths

02 / FINANCIAL MODEL

Financial model

Geometric Brownian Motion models the stock price as a stochastic process with proportional drift and volatility.

Model equation

$$dS_t = r S_t dt + \sigma S_t dW_t$$

Under the risk-neutral measure, the expected return is the continuously compounded risk-free rate r. This is a valuation assumption, not a forecast that the stock will earn r in the real world.

Exact time-step transition

Because GBM has a closed-form transition, the code does not apply a simple additive Euler step. It samples log returns directly:

$$S_{t+\Delta t} = S_t \exp[(r - 0.5\sigma^2)\Delta t + \sigma\sqrt{\Delta t} Z]$$

where Z ~ N(0,1). The -0.5 sigma^2 correction ensures the simulated level has the intended risk-neutral expectation.

Vectorized implementation

```
dt = T / N

shocks = np.random.standard_normal((N, 10_000))

log_returns = (r - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * shocks

paths = S0 * np.exp(np.cumsum(log_returns, axis=0))
```

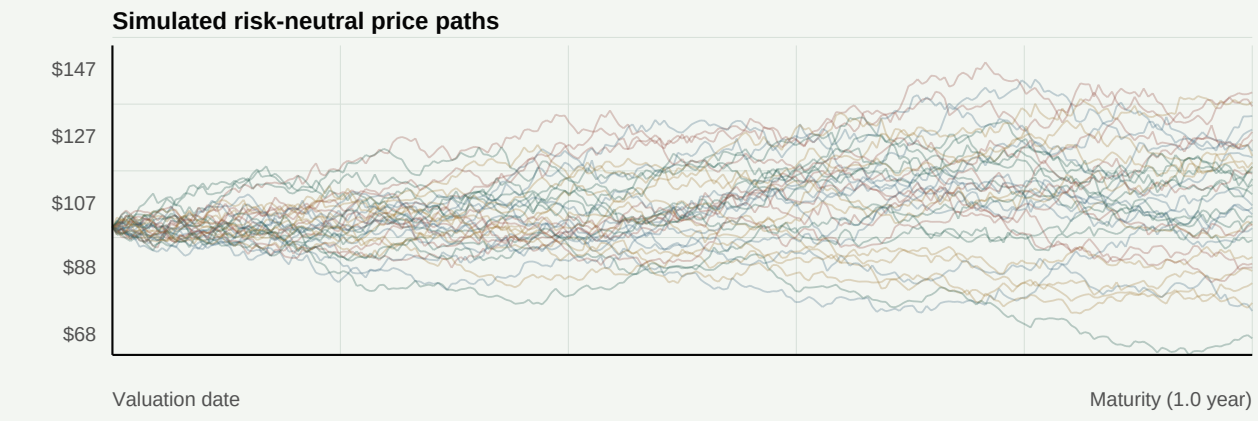

Figure 1. Thirty-five paths from the seeded reference experiment. These paths represent valuation scenarios under the GBM model, not real-world forecasts or confidence bands.

Implementation choices

- The returned array has shape (N, 10,000); rows are future time steps and columns are independent paths.
- The valuation-date price is intentionally excluded. The first row corresponds to Delta t and the last row to maturity.
- Input checks reject non-positive prices or maturities, negative volatility, and non-positive step counts.

03 / EUROPEAN OPTION VALUATION

European option valuation

For a European option, exercise occurs only at maturity. The estimator discounts the cross-sectional mean of terminal payoffs.

Risk-neutral estimators

Call: C0 = exp(-rT) E[max(ST - K, 0)]
Put: P0 = exp(-rT) E[max(K - ST, 0)]

The same simulated terminal prices support both payoffs, which makes the implementation compact and provides a direct put-call-parity diagnostic.

Reference experiment

Input	Value	Interpretation
Spot price S0	$100.00	Current underlying price
Strike K	$100.00	At-the-money strike
Risk-free rate r	5.00%	Continuously compounded annual rate
Volatility sigma	20.00%	Constant annualized volatility
Maturity T	1.00 year	Time to expiration
Simulation	10,000 paths x 252 steps	Seed 20260730 for reproducible analysis

Instrument	Monte Carlo	95% MC interval	Analytical benchmark	Absolute difference
European call	$10.2768	$9.9903 - $10.5633	$10.4506	$0.1738
European put	$5.6982	$5.5266 - $5.8697	$5.5735	$0.1246

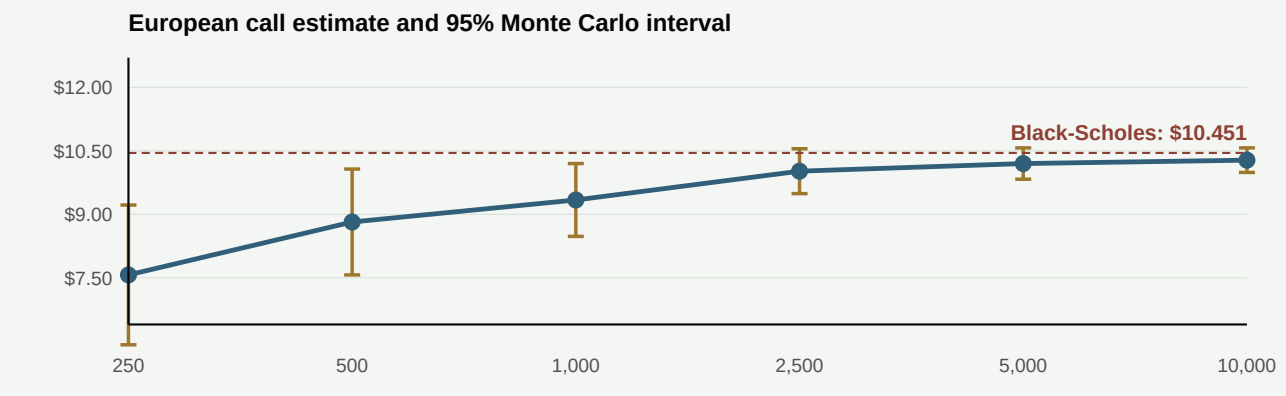

Figure 2. The estimator fluctuates at low path counts and stabilizes as sampling increases. Error bars are approximate 95% intervals based on the sample standard error of discounted payoffs.

Interpretation

The seeded call estimate of $10.2768 is compared with the Black-Scholes value of $10.4506. The analytical benchmark lies inside the reported Monte Carlo interval. This is the expected behavior of an unbiased estimator with finite sampling noise; exact equality is neither required nor expected.

04 / AMERICAN OPTION VALUATION

American option valuation

American options may be exercised at any simulated date. Their value therefore depends on comparing immediate exercise with the value of keeping the option alive.

The project implements the Longstaff-Schwartz least-squares Monte Carlo method.

Backward induction

Step	What the implementation does
1. Initialize	Set pathwise values equal to payoff at maturity.
2. Discount	Move each path's future value back by one simulation interval.
3. Filter	Keep only in-the-money paths, because out-of-the-money exercise is never rational.
4. Regress	Fit a degree-2 polynomial of discounted future cash flow on the current stock price.
5. Decide	Exercise when intrinsic value exceeds fitted continuation value; otherwise continue.
6. Repeat	Work backward to the first simulated date and discount once more to time zero.

Continuation-value approximation

The regression estimates E[discounted future cash flow | current price] using only in-the-money paths. This converts the optimal-stopping problem into a sequence of cross-sectional prediction problems.

$6.136	$6.090	$5.698
LSM AMERICAN PUT	1,500-STEP BINOMIAL	EUROPEAN PUT (MC)

For the reference case, Longstaff-Schwartz estimates the American put at $6.1357, while an independent 1,500-step Cox-Ross-Rubinstein tree gives $6.0899. The LSM value also exceeds the European Monte Carlo put estimate of $5.6982, consistent with the economic value of early-exercise flexibility.

Important caveats

- The polynomial basis is deliberately simple. Scaling, orthogonal basis functions, or alternative state variables may improve numerical conditioning.
- The estimate contains both simulation error and regression/optimal-stopping bias; a single run should not be treated as a precise market quote.
- Exercise is allowed only at the 252 simulated dates, so the implementation approximates continuous American exercise with a Bermudan grid.
- A non-dividend-paying American call should ordinarily have the same value as its European counterpart; this provides an additional diagnostic, not a promise of exact numerical equality.

05 / VALIDATION AND CONTROLS

Validation and controls

A pricing implementation is credible only when it is tested against independent financial identities and numerical benchmarks.

Check	Observed result	Assessment
Black-Scholes call	MC $10.2768 vs analytical $10.4506	Analytical price falls inside 95% MC interval
Black-Scholes put	MC $5.6982 vs analytical $5.5735	Analytical price falls inside 95% MC interval
Put-call parity	Residual $-0.2985	Small finite-sample residual relative to $100 spot
American put benchmark	LSM $6.1357 vs tree $6.0899	Directionally and numerically consistent
Expected terminal level	Sample $104.813 vs theory $105.127	Consistent with E[ST] = S0 exp(rT)
Price positivity	Minimum simulated value $43.314	Exponential construction preserves positivity

What the controls do - and do not - prove

- Benchmark agreement supports the mechanics of path generation, payoff calculation, and discounting for the tested parameter set.
- Put-call parity is not an independent stochastic test when calls and puts reuse the same terminal paths, but it remains a useful end-to-end consistency check.
- A seeded analysis makes the reported numbers reproducible. The application itself currently uses NumPy's global random state, so repeated UI runs naturally vary.
- These checks validate implementation behavior under GBM. They do not validate GBM as a complete empirical description of equity markets.

Reproducibility note

The project function fixes the path count at 10,000 and does not accept a Generator or seed argument. For this report, np.random.seed(20260730) was set immediately before simulation. A production-oriented API should accept an explicit numpy.random.Generator to avoid hidden global state.

Input validation already present

Function	Rejected inputs
simulate_gbm_paths	Non-positive spot, maturity, or step count; negative volatility
price_european_options	Empty/non-2D paths; non-positive strike or maturity
price_american_option_lsm	Fewer than two steps; invalid type; negative polynomial degree

06 / SYSTEM DESIGN

System design

The project separates numerical pricing logic from presentation. The core module can be imported from scripts or tests, while Streamlit provides a lightweight interactive layer.

Layer	Responsibility	Evidence in repository
Numerical core	Simulation, European payoff pricing, LSM backward induction	gbm.py
Example client	Minimal executable usage and console output	example.py
Interactive UI	Inputs, option selection, calculation trigger, path chart	app.py
Dependency declaration	Reproducible package requirements	requirements.txt
Technical guide	Setup, API behavior, and methodology summary	README.md

Data flow

USER INPUTS	→	NUMPY ENGINE	→	VALUATION	→	OUTPUT
S0, K, r, sigma, T, N		10,000 GBM paths		European payoff or LSM exercise		Price + 50-path Plotly chart

Engineering strengths

- NumPy broadcasting and cumulative sums eliminate Python loops from the path-generation hot path.
- The pricing functions use type hints, docstrings, explicit validation, and predictable return types.
- The dashboard does not mix UI logic into the numerical functions, keeping the core reusable.
- The interface explains financial terms and makes clear that the estimate is not investment advice.

Known engineering constraints

Constraint	Practical effect	Recommended improvement
Path count fixed at 10,000	Accuracy/performance trade-off cannot be configured	Expose n_paths as an argument with a safe UI cap
Global random state	Runs cannot be reproduced through the public API	Accept numpy.random.Generator or seed
Full path matrix retained	Memory grows with steps x paths	Use terminal-only mode for European options
No uncertainty shown	A point estimate may imply false precision	Report standard error and confidence interval
Raw polynomial LSM basis	Potential conditioning issues	Scale state variables and test Laguerre basis functions

07 / LIMITATIONS AND ROADMAP

Limitations and roadmap

The project is a transparent educational implementation. It should not be described as an institutional options-pricing library, a trading strategy, or a source of executable market prices.

Model limitations

- Constant volatility excludes the volatility smile, skew, term structure, and volatility clustering observed in markets.
- Continuous lognormal diffusion excludes jumps, default, trading halts, and other discontinuous price behavior.
- The model contains no dividends or foreign yield. Dividend-paying equities require a carry adjustment and can change American-call exercise incentives.
- A single constant risk-free rate ignores the discount curve and collateral/funding conventions used in practice.
- The output is model value under assumptions, not a prediction of realized option profit or loss.

Recommended roadmap

Priority	Enhancement	Why it matters
1	Injectable RNG, configurable path count, and unit tests	Reproducibility and software reliability
2	Confidence intervals and convergence diagnostics in the UI	Makes Monte Carlo uncertainty visible
3	Black-Scholes and put-call-parity regression tests	Protects core pricing behavior from future changes
4	Antithetic variates and control variates	Reduces estimator variance without only adding paths
5	Dividend yield and term-structure inputs	Expands economically relevant use cases
6	Scaled/orthogonal LSM bases with out-of-sample exercise evaluation	Improves early-exercise stability
7	Heston or local-volatility extension	Addresses smile and stochastic-volatility behavior

Honest project claim

I built and validated a vectorized Monte Carlo pricing prototype that connects GBM theory to European and American option valuation. It reproduces standard benchmarks under controlled assumptions and exposes where sampling error, regression approximation, and model risk remain.

Repository guide

Run the command-line example with python3 example.py. Start the interface with python3 -m streamlit run app.py. The implementation and documentation are located in the gbm_price_simulator/ project directory.

Method note

All numerical figures in this case study were generated from the repository's current gbm.py implementation. Black-Scholes values use the closed-form no-dividend formula. The American put benchmark uses a separately implemented 1,500-step Cox-Ross-Rubinstein tree. Reported Monte Carlo intervals are normal approximations based on discounted payoff standard errors. Monetary values are illustrative and denominated in generic dollars.