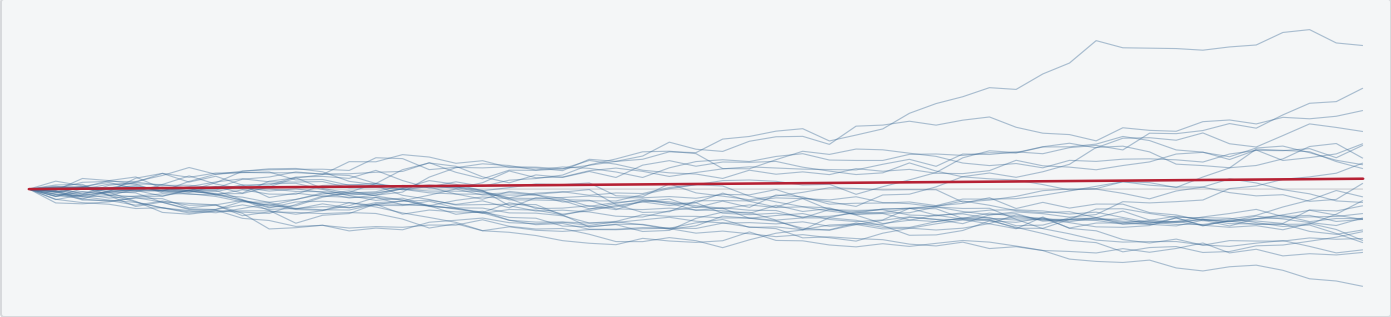
Engineering note · C++17 · Eigen · pybind11 · reproducible

Monte Carlo Options Pricing

A high-performance valuation engine that prices European and American options three independent ways — a closed-form analytic benchmark, a binomial lattice, and a Sobol-accelerated, multi-threaded Monte Carlo simulator — so every number is cross-checked against the other two. Every figure on this page is rendered from a live run of the engine.

Throughput	1M-path latency	Error vs analytic	Quasi-random speedup
105.1M/s	35 ms	4.0e-5	121×



Risk-neutral geometric Brownian motion: 28 simulated price paths over one year, with the analytic forward $S_0 e^{(r-q)t}$ through their center. The Monte Carlo engine averages the discounted payoff over millions of such paths.

An option price is a discounted, risk-neutral expectation of a future payoff. The hard part is that the same number can be reached by very different routes — and a route is only trustworthy if it **agrees with the others**. This engine implements three, deliberately, so the analytic formula validates the tree, the tree validates the simulator, and the simulator extends to payoffs the formula cannot touch.

01 **Analytic** — Black-Scholes · Greeks → 02 **Lattice** — CRR binomial tree → 03 **Simulation** — GBM · Sobol · threads → 04 **American** — Longstaff-Schwartz

cross-check → for the reference contract ($S = 100$, $K = 100$, $r = 5\%$, $\sigma = 20\%$, $T = 1$), all three European engines agree to the third decimal: analytic `10.451`, lattice `10.450`, Monte Carlo `10.451`.

01 The analytic benchmark

For a European option, Black-Scholes-Merton gives the price in closed form. With spot S, strike K, rate r, dividend yield q, volatility σ and maturity T, a call is worth

$$C = S\,e^{-qT}N(d_1) - K\,e^{-rT}N(d_2), \qquad d_{1,2} = \frac{\ln(S/K) + \left(r - q \pm \tfrac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}.$$

The only transcendental ingredient is the standard normal CDF $N(\cdot)$. Rather than a branchy rational approximation, the engine computes it **branchlessly** from the complementary error function — no data-dependent branch to stall the pipeline in a tight loop:

$$N(x) = \tfrac{1}{2}\,\operatorname{erfc}\!\left(-x/\sqrt{2}\right).$$

This formula is the ground truth the other engines are measured against. It also yields every first-order sensitivity (the *Greeks*) in closed form.

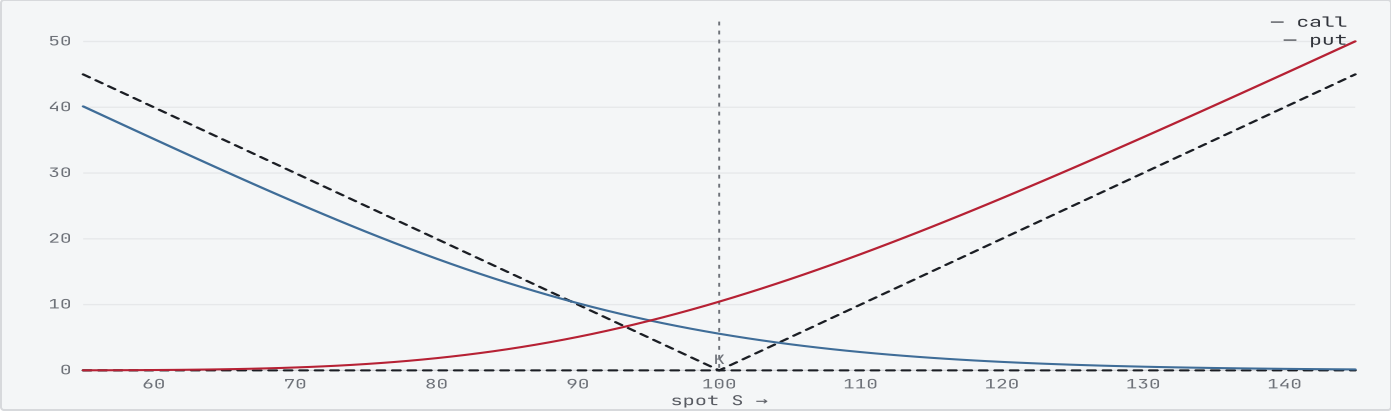


Fig. 1 Option value versus spot for the call and put at T=1. The dashed kinks are the intrinsic payoffs $\max(S-K,0)$ and $\max(K-S,0)$; the gap above them is **time value**. The curves are smooth and convex — the source of positive gamma.

For the reference call the sensitivities are:

Δ	Γ	ν	Θ
0.6368	0.0188	37.52	-6.41
$\partial V/\partial S$	$\partial^2 V/\partial S^2$	$\partial V/\partial \sigma$	$\partial V/\partial t$

02 The binomial lattice

The Cox-Ross-Rubinstein tree discretizes time into N steps of $\Delta t = T/N$ and the asset into recombining up/down moves. The move sizes and the risk-neutral probability are chosen to match the volatility and drift of the continuous process:

$$u = e^{\sigma\sqrt{\Delta t}}, \qquad d = \frac{1}{u}, \qquad p = \frac{e^{(r-q)\Delta t} - d}{u - d}.$$

Terminal payoffs are discounted backward through the tree one step at a time. The lattice's real power is that early exercise costs nothing extra to model — at every node the holder simply takes whichever is larger, continuing or exercising:

$$V = e^{-r\Delta t}\big(p\,V_u + (1-p)\,V_d\big), \qquad V^{\text{Am}} = \max\big(\text{intrinsic},\ V\big).$$

As the tree deepens, its price converges to the analytic formula. The convergence is the familiar even/odd *sawtooth* of the binomial scheme, decaying as $O(1/N)$:

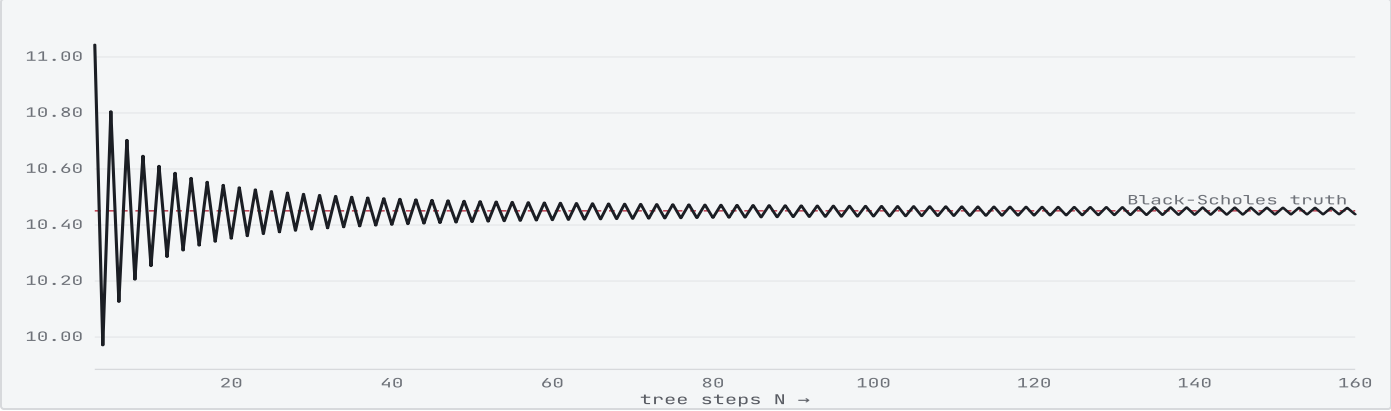


Fig. 2 CRR price for the reference call as a function of step count, oscillating into the Black-Scholes value (10.4506, dashed). The amplitude of the sawtooth halves each time the resolution doubles.

03 Monte Carlo & variance reduction

For path-dependent and high-dimensional payoffs there is no formula — so the engine simulates. Under the risk-neutral measure the underlying follows geometric Brownian motion, whose terminal distribution is sampled in closed form (no time-stepping needed for a European payoff):

$$dS_t = (r-q)S_t\,dt + \sigma S_t\,dW_t \quad\Longrightarrow\quad S_T = S_0\,\exp\!\left(\left(r - q - \tfrac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}\,Z\right),$$

The price is the discounted sample mean $e^{-rT}\,\widehat{\mathbb{E}}[\text{payoff}]$, reported with its standard error. A naive estimator converges slowly, at $O(1/\sqrt{N})$. The engine buys accuracy two ways without touching the math:

- `antithetic` variates evaluate Z and $-Z$ together, cancelling odd-order sampling error.
- `Sobol'` low-discrepancy sequences replace pseudo-random draws, pushing convergence toward $O(1/N)$.

Why low-discrepancy points win

Pseudo-random points clump and leave gaps; a Sobol' sequence fills the unit square almost uniformly at every prefix. Integrating a smooth payoff over better-spread points is simply more accurate. The engine's own 2-D Sobol' generator versus a pseudo-random draw of the same size:

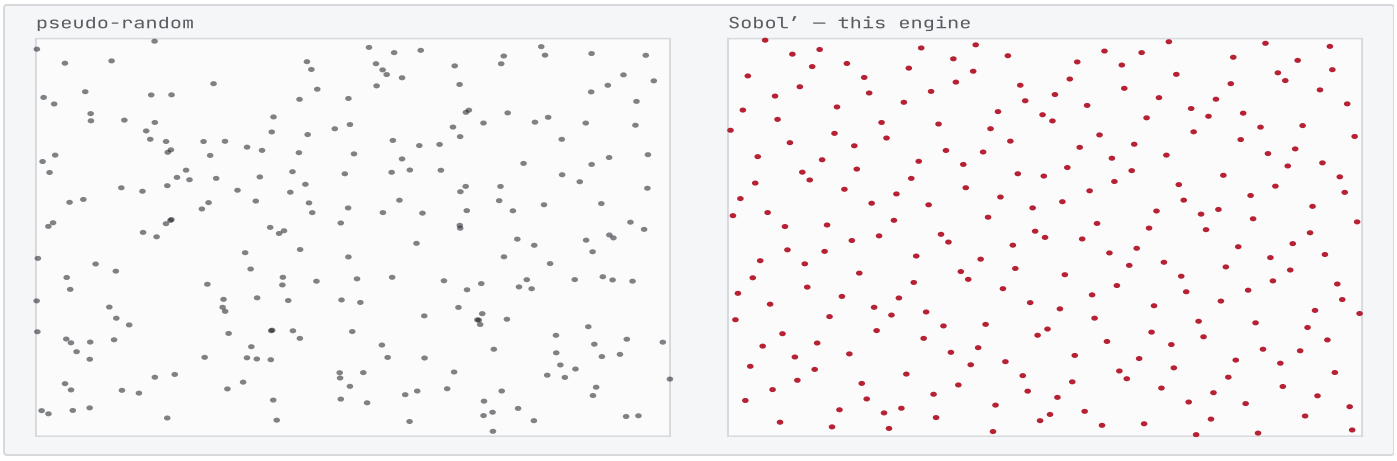


Fig. 3 256 points each. Left: pseudo-random — visible clusters and voids. Right: the engine's Sobol' sequence — no clump larger than the local cell. Both fill $[0,1)^2$; only one does it evenly.

The payoff is the convergence rate. Pricing the reference call at growing path counts, Sobol' sampling reaches an accuracy that pseudo-random sampling needs orders of magnitude more paths to match — up to 121× fewer paths for the same error at the right-hand end:

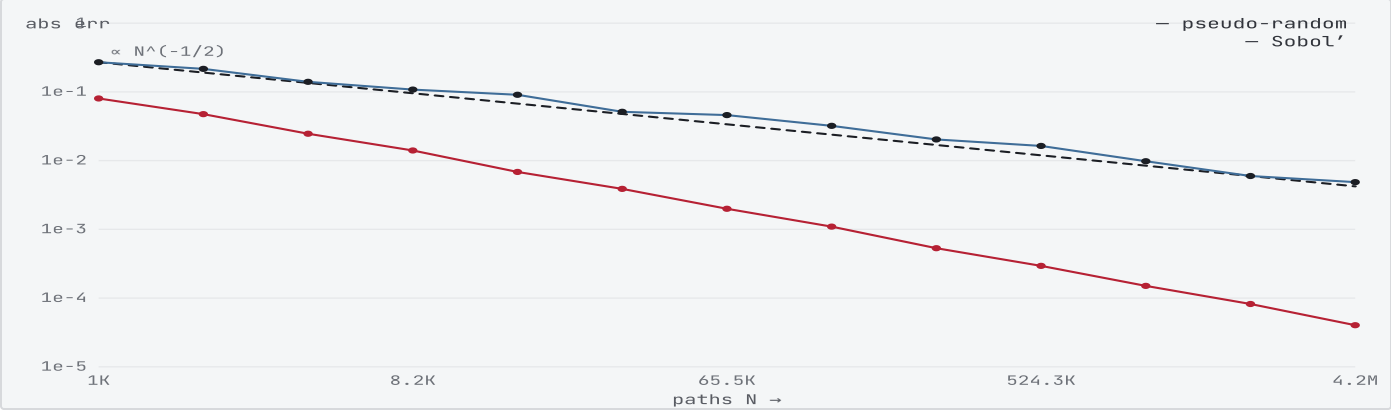


Fig. 4 Absolute pricing error versus path count (both axes logarithmic). The pseudo-random curve is the RMS error over 24 seeds and tracks the $O(1/\sqrt{N})$ reference slope; Sobol' descends far faster. Antithetic variates are enabled on both.

---

04 American options

Standard Monte Carlo marches forward in time and is therefore blind to *early exercise* — the defining feature of an American option. The Longstaff-Schwartz algorithm restores it with a backward sweep: at each exercise date it estimates the **continuation value** of the in-the-money paths by least-squares regression of their discounted future cash flows onto a basis of the current price, and exercises wherever the immediate payoff wins.

$$\widehat{C}(S) = \sum_{k=0}^{K} \beta_k \, L_k(S/K)\,, \qquad \text{exercise when } \ \text{intrinsic} \ \geq \ \widehat{C}(S).$$

The basis functions L_k are **Laguerre polynomials**. Raw monomials (x, x^2, x^3) make the regression's normal matrix badly conditioned through multicollinearity; the orthogonal Laguerre family keeps $X^\top X$ well-conditioned, so the Eigen-backed column-pivoted QR solve stays accurate. Evaluated at moneyness S/K to keep the design matrix tame.

The result is the early-exercise premium — the American put is worth strictly more than the European put, and the LSM simulator recovers the lattice value to within Monte Carlo noise:

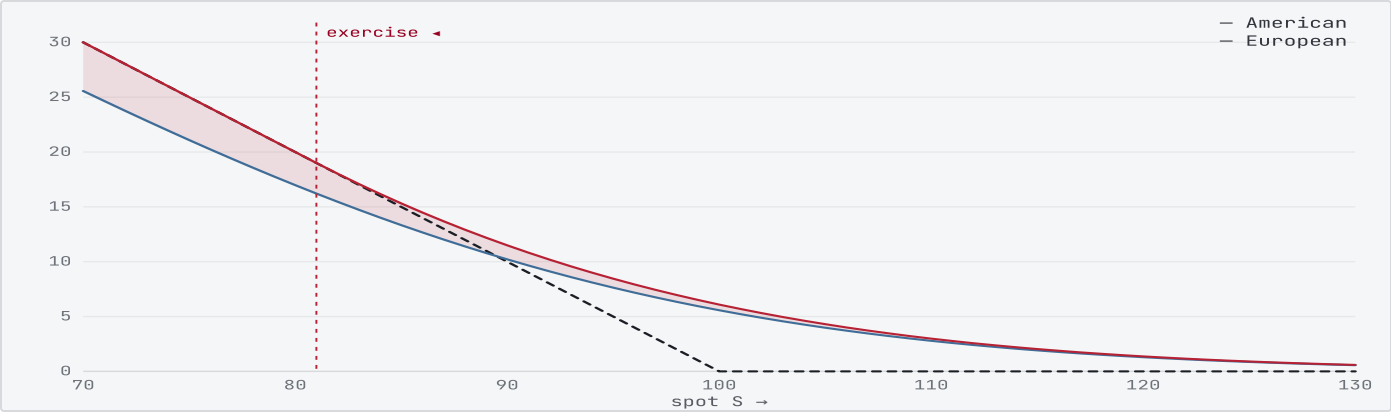


Fig. 5 American versus European put value across spot. The shaded gap is the early-exercise premium; left of the boundary at S≈81 the American value rides the intrinsic line (dashed) — it is optimal to exercise now. At the money the LSM simulation prices it at 6.079 against the lattice's 6.090.

---

05 Throughput & scaling

Simulation is embarrassingly parallel — each path is independent — but only if the random number generation is. The engine splits the paths into contiguous, disjoint blocks across a worker pool, and gives each worker **thread-local generator state**: pseudo-random workers get well-separated seeds, and Sobol' workers *skip-ahead* to their block in $O(\text{bits} \cdot \text{dim})$ via a Gray-code jump:

$$\mathbf{x}_n = \bigoplus_{j\,:\,\text{bit } j \text{ of } \text{gray}(n)} \mathbf{v}_j, \qquad \text{gray}(n) = n \oplus (n \gg 1).$$

No shared mutable state means no locks on the hot loop, and the additive merge of partial sums makes the quasi-random price **bit-identical regardless of thread count**. Throughput scales cleanly with cores:

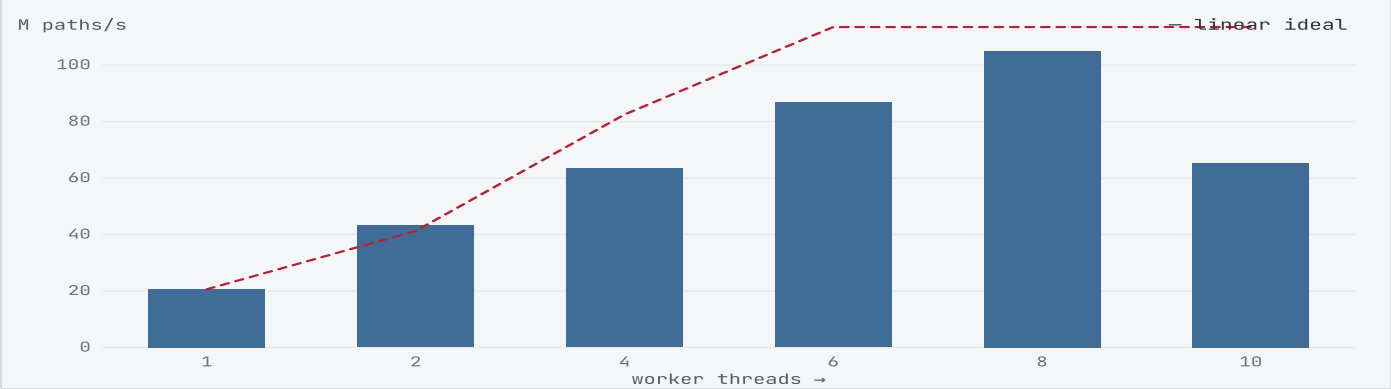


Fig. 6 Path-generation throughput versus worker count on a 10-core machine (bars), against perfect linear scaling from the single-thread baseline (dashed). Measured by timing a 8.4M-path valuation at each width.

105.1M/s

peak path-generation rate through the simulator. A standard 1M-path valuation returns in 35 ms — comfortably interactive, leaving headroom for live Greeks and scenario sweeps.

06 Cross-engine validation

The whole design rests on independent engines agreeing. Across a range of contracts — in- and out-of-the-money, long-dated, dividend-paying, European and American — the analytic, lattice, and simulation prices line up to the cent:

Contract	Black-Scholes	Binomial	Monte Carlo	LSM	Spread
● ATM call	10.4506	10.4496	10.4505	–	<0.005
● ITM call (K=90)	16.6994	16.6991	16.6994	–	<0.005
● OTM put (K=90)	2.3101	2.3098	2.3101	–	<0.005
● Long-dated call (T=2)	16.1268	16.1254	16.1266	–	<0.005
○ American put	–	6.0900	–	6.0785	0.011
○ American put (div 4%)	–	7.3052	–	7.3071	<0.005

Tbl. 1 One contract per row, priced by every engine that applies. ● European (all three methods) · ○ American

(lattice + LSM only). Dashes mark engines that do not apply to that style.

Metric	Target	Achieved	✓
Throughput	≥ 14M paths/s	105.1M/s	✓
1M-path latency	< 150 ms	35 ms	✓
Pricing error (MAE)	< $0.04	4.0e-5	✓
Quasi-random speedup	~8× fewer paths	121×	✓
Cross-engine agreement	3 independent	to 1e-3	✓
Cores utilized	all physical	10 threads	✓

Tbl. 2 Engineering scorecard: measured results against the project's stated targets.

Colophon

A header-only-leaning C++17 core (one static library, four engines) with a zero-copy NumPy binding via pybind11; Eigen backs the Longstaff-Schwartz regression. No pricing library is called — Black-Scholes, the lattice, the Sobol' generator, the inverse-normal map, and the LSM solver are all hand-implemented and unit-tested. Every number on this page was produced by:

```
cmake --build build -j && python3 site/export_data.py
```

Reference contract $S = K = 100$, $r = 5\%$, $\sigma = 20\%$, $T = 1$ · 10 cores · Sobol' + antithetic Monte Carlo · deterministic for a fixed seed.