QuantLab

Computational Finance & Scientific Machine Learning Workstation

A Step-by-Step Technical and Research Development Report

Rancy Chepchirchir

2 September 2026

Abstract

QuantLab is an interactive computational-finance research workstation developed to compare analytical pricing, lattice methods, finite-difference PDE solvers, Monte Carlo simulation, implied-volatility models, and physics-informed neural networks within a common experimental environment. The project began as an option-pricing comparison application and evolved into a broader research platform with four principal modules: Pricing Lab, Volatility Lab, Surface Atlas, and Research Lab. This report records the project chronologically, explains the mathematical methods, documents key implementation and deployment decisions, describes the American Option Surface Atlas and PINN convergence experiments in detail, and preserves the scientific cautions needed for future work. The report is intended both as project documentation and as a technical notebook that can support future thesis, paper, portfolio, and research-development work.

Contents

1 Project Vision

QuantLab was built around a simple but increasingly important question: how should classical numerical finance and modern scientific machine learning be compared in a way that is computationally reproducible, visually interpretable, and scientifically honest?

Rather than building a collection of unrelated calculators, the project uses shared contract assumptions and common reference grids so that methods can be compared directly. Its current public identity is:

> **QuantLab – Computational Finance & Scientific Machine Learning Workstation.** An interactive research platform for option pricing, volatility modelling and scientific machine learning, combining analytical methods, numerical PDE solvers, Monte Carlo simulation and physics-informed neural networks.

The current application structure is:

Module	Route
Pricing Lab	`/`
Volatility Lab	`/volatility-lab`
Surface Atlas	`/surface-atlas`
Research Lab	`/research-lab`

The old concept of a separate "Overview" page was removed because the root route already functioned as the Pricing Lab. This simplified the information architecture and made the research modules first-class components of the application.

2 System Architecture

2.1 Frontend

The frontend is built with Next.js, React, TypeScript and Tailwind CSS. Recharts is used for two-dimensional quantitative graphics, while Plotly is used for interactive three-dimensional surfaces.

The visual design evolved toward a dark scientific-workstation aesthetic: dense analytical grids, navy/purple/cyan accents, compact metric cards, and research-oriented surface plots rather than a conventional financial dashboard.

2.2 Backend

The backend is implemented in FastAPI and relies on NumPy and SciPy for numerical methods. Pydantic is used for typed API models. HTTPX/Requests support external data-provider abstractions where required.

2.3 Deployment and CI

The backend is deployed on Railway and the frontend on Vercel. GitHub Actions provides continuous integration. A key deployment constraint is that PyTorch is not part of the production backend requirements.

2.4 Offline scientific-ML architecture

PINNs are trained offline. The trained experiment outputs are exported as JSON artifacts and consumed by the production API using lightweight numerical interpolation. This prevents an API request from initiating expensive retraining and keeps production deployment independent of PyTorch.

The architecture can be summarized as follows:

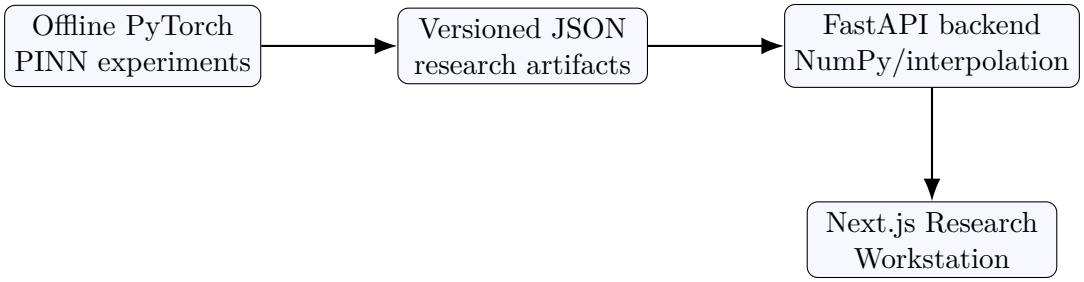

Figure 1: Production architecture for PINN research outputs. Training is deliberately separated from runtime serving.

3 Version 1: Pricing Foundations

3.1 Shared pricing experiment

The first version established a common pricing input model containing spot price, strike, risk-free rate, volatility, maturity, dividend yield and option type. Multiple pricing engines are evaluated under the same market assumptions.

The principal methods were:

- Black–Scholes analytical pricing;
- Cox–Ross–Rubinstein binomial pricing;
- Crank–Nicolson finite differences;
- Monte Carlo simulation;
- Black–Scholes Greeks;
- spot sweeps and sensitivity plots;
- numerical convergence experiments;
- accuracy-versus-runtime comparisons.

3.2 Black–Scholes benchmark

For a European call with continuous dividend yield q,

$$C = S_0 e^{-qT} N(d_1) - K e^{-rT} N(d_2),$$

where

$$d_1 = \frac{\ln(S_0/K) + (r - q + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}, \qquad d_2 = d_1 - \sigma\sqrt{T}.$$

Where an analytical solution exists, it provides a convenient benchmark for lattice, finite-difference and Monte Carlo convergence.

3.3 CRR lattice

The CRR tree approximates the underlying diffusion using

$$u = e^{\sigma\sqrt{\Delta t}}, \qquad d = u^{-1},$$

with risk-neutral probability

$$p = \frac{e^{(r-q)\Delta t} - d}{u - d}.$$

For American options, backward induction applies the exercise projection

$$V = \max\{V_{\text{continuation}}, \Phi(S)\}$$

at each node.

3.4 Finite differences

The Black–Scholes PDE is

$$\frac{\partial V}{\partial t} + (r-q)S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rV = 0.$$

Crank–Nicolson averages the explicit and implicit time discretizations. During early development, the comparison endpoint was improved by using an efficient tridiagonal solution structure instead of a computationally expensive dense approach.

3.5 Monte Carlo

Risk-neutral terminal prices are sampled from

$$S_T = S_0 \exp\left[\left(r - q - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}Z\right], \qquad Z \sim N(0,1).$$

The discounted payoff estimator is

$$\hat{V} = e^{-rT}\frac{1}{M}\sum_{m=1}^{M}\Phi(S_T^{(m)}).$$

The Pricing Lab reports both approximation error and Monte Carlo uncertainty, making stochastic sampling visibly different from deterministic discretization error.

3.6 Convergence Lab

The frontend includes convergence plots for lattice refinement and Monte Carlo sampling. This was an early step toward the central QuantLab idea: methods should be compared through convergence and computational cost, not only through final prices.

4 Version 2: Implied Volatility and Surface Modelling

V2 extended QuantLab from option-value computation to market-implied volatility structure.

4.1 Implied-volatility inversion

Given a market price P_{mkt}, implied volatility solves

$$P_{\text{model}}(\sigma_{\text{impl}}) - P_{\text{mkt}} = 0.$$

A robust bisection procedure was implemented. For American options, CRR pricing can be used inside the inversion loop when exercise optionality invalidates a direct European Black–Scholes inversion.

4.2 Surface modelling

The volatility module grew to include interpolation, raw SVI and SSVI representations, model comparisons, calibration and arbitrage diagnostics.

A generic SVI slice can be written as

$$w(k) = a + b\left\{\rho(k-m) + \sqrt{(k-m)^2 + \sigma^2}\right\},$$

where k is log-moneyness and w total implied variance.

4.3 Scientific cautions

Several correctness decisions were made during this phase:

- butterfly tests must not average call and put prices;
- the option type used in market butterfly diagnostics must be internally consistent;
- SVI/SSVI surfaces in the implemented workflow are European-call constructions;
- fixed-strike total-variance calendar checks are diagnostics rather than universal proofs of static-arbitrage freedom.

These distinctions should remain visible in future documentation and UI labels.

5 Version 3: Research Workstation

The third phase changed QuantLab from a conventional quantitative-finance application into a more explicit research environment.

Major additions included a shared shell, research navigation, dense KPI strips, compact market controls, 3D volatility surfaces, SSVI total-variance visualization and, most importantly, the American Option Surface Atlas.

6 American Option Surface Atlas

6.1 Why American puts?

The current projected Crank–Nicolson implementation is designed for American puts. This makes the put the natural instrument for studying the obstacle problem, the stopping region and the numerical free boundary.

6.2 Variational inequality

For payoff $\Phi(S)$, the American pricing problem is

$$\max\left\{\frac{\partial V}{\partial t}+(r-q)SV_S+\frac{1}{2}\sigma^2S^2V_{SS}-rV,\Phi(S)-V\right\}=0,\qquad V(T,S)=\Phi(S).$$

For a put,

$$\Phi(S)=\max(K-S,0).$$

The Surface Atlas uses time-to-maturity

$$\tau=T-t,$$

so plots run naturally from maturity backward into longer time remaining.

6.3 Projected Crank–Nicolson

A numerical CN candidate is projected onto the obstacle:

$$V_i^n=\max\left(V_i^{n,\mathrm{CN}},\Phi(S_i)\right).$$

This produces a practical numerical reference surface. It must not be called an exact solution.

6.4 Common comparison grid

CRR produces scalar prices at requested state points while CN naturally produces a rectangular space-time grid. To compare the two methods, CRR values are evaluated on the same common grid used for the research atlas.

The signed error surface is

$$E_{\mathrm{CRR}}(S,\tau)=V_{\mathrm{CRR}}(S,\tau)-V_{\mathrm{CN}}(S,\tau).$$

6.5 Exercise gap and free-boundary estimate

Define

$$G(S,\tau)=V(S,\tau)-\max(K-S,0).$$

This is an exercise/continuation gap, not a discretization error.

A numerical boundary estimate can be written

$$S^*(\tau)\approx\sup\{S_i<K:G(S_i,\tau)\le\epsilon\}.$$

The resulting boundary is a diagnostic extracted from the discretized reference surface.

6.6 Poster-style analytical matrix

The atlas design intentionally uses small multiples because they support method-to-method comparison better than a single oversized 3D plot. The early matrix included CRR, CN, signed error, exercise gap and free-boundary visualizations. This structure later became the basis for PINN comparison panels.

7 Physics-Informed Neural Network Experiment

7.1 Model formulation

The PINN approximates the option value $V_\theta(S,t)$ with a Tanh network. Rather than treating the American obstacle as an ordinary equality PDE, the experiment incorporates complementarity through a smoothed Fischer–Burmeister function

$$\phi_\epsilon(a,b) = \sqrt{a^2 + b^2 + \epsilon^2} - a - b,$$

where

$$a = V - \Phi, \qquad b = -\mathcal{L}V.$$

Here $\mathcal{L}$ denotes the Black–Scholes differential operator including discounting.

7.2 Training configuration

The existing V2 experiment uses a representative configuration of 4000 epochs, 3000 interior points, 1000 terminal samples, 1000 boundary samples, learning rate 10^{-3}, complementarity weight 10, terminal/boundary weights 5 and seed 42.

The base contract family in the stored experiment is

$$K = 100, \quad r = 0.05, \quad \sigma = 0.20, \quad T = 1, \quad q = 0.$$

7.3 Coordinate conversion

The PINN was trained in calendar time t, whereas Surface Atlas visualizes $\tau = T - t$. This transformation is required before interpolation and comparison. Forgetting it would silently misalign surfaces while still producing visually plausible plots.

7.4 Offline artifacts

Two important stored artifacts are:

```
backend/experiments/results/american_pinn_v2_surface.json
backend/experiments/results/american_pinn_convergence_atlas.json
```

The convergence experiment captures snapshots at 500, 1000, 2000 and 4000 epochs from a single training trajectory.

8 PINN Error Topography and Learning Dynamics

8.1 Reference error

PINN error is measured against projected CN:

$$E_{\text{PINN}}(S,\tau) = V_{\text{PINN}}(S,\tau) - V_{\text{CN}}(S,\tau).$$

Absolute error is

$$|E_{\text{PINN}}(S,\tau)|.$$

The language “exact error” is intentionally avoided because CN is a numerical reference.

8.2 Improvement surface

To visualize where training made progress,

$$I_{500\to4000}(S,\tau) = |E_{500}(S,\tau)| - |E_{4000}(S,\tau)|.$$

Positive regions indicate a reduction in absolute error between early and late training.

8.3 Boundary-distance profile

A grid-aware distance is used:

$$\Delta S = \text{median}(S_{i+1} - S_i), \qquad d_h = \frac{|S - S^*(\tau)|}{\Delta S}.$$

The error field is summarized in bands such as $< 1\Delta S$, $1 - 2\Delta S$, $2 - 4\Delta S$, $4 - 8\Delta S$, and $8 + \Delta S$.

The observed training trajectory supports a descriptive picture of broad-domain error gradually collapsing into more localized residual structure. This is informative evidence but should not be overstated as a formal causal result.

9 Linear and Logarithmic Error Views

The PINN Error-Evolution Atlas now supports two complementary display modes.

9.1 Linear scale

$$z = |E|.$$

A pooled 97.5th-percentile upper display limit is used across all four checkpoints. This prevents a few large early errors from visually flattening the converged surface while preserving the true uncapped maximum as a reported metric.

9.2 Logarithmic scale

$$z = \log_{10}(\max(|E|, 10^{-6})).$$

The log view exposes low-magnitude residual geometry that cannot be read easily on a linear scale. Importantly, MAE, RMSE and maximum-error cards remain in the original price-error units. Only the display coordinate is transformed.

The free-boundary ridge must be transformed consistently into the same z coordinate in log mode.

10 Frontend WebGL Failure and Resolution

10.1 Observed failure

After several new 3D PINN panels were mounted simultaneously, the top CRR panel became a large white region. Browser logs reported a Plotly/WebGL shader compilation failure.

Because the same generic surface component worked for CN and PINN, the initial possibility of corrupt CRR data was tested directly by removing the lower multi-surface error-evolution component. The CRR surface immediately rendered again.

10.2 Diagnosis

The failure was caused by browser WebGL context/resource pressure rather than pricing data or Plotly trace configuration.

10.3 Architectural fix

A reusable lazy section was introduced using `IntersectionObserver`. Lower 3D components mount when approaching the viewport and may unmount when hidden. This releases WebGL contexts and allows the primary method atlas to stay eager.

This fix is more robust than modifying arbitrary plot parameters because it addresses the actual resource lifecycle.

11 Navigation and Portfolio Integration

The V3 sidebar was simplified so that the main route is explicitly Pricing Lab. Surface Atlas was added as a first-class navigation destination and promoted through a CTA on the main page.

The portfolio-ready information architecture is therefore:

```
Pricing Lab /
Volatility Lab /volatility-lab
Surface Atlas /surface-atlas
Research Lab /research-lab
```

A concise portfolio description is:

> **QuantLab – Computational Finance & Scientific Machine Learning Workstation.** An interactive research platform for option pricing, volatility modelling and scientific machine learning, combining analytical methods, numerical PDE solvers, Monte Carlo simulation and physics-informed neural networks.

The recommended project status is "Active research project" rather than "finished", because the existing application already tells a coherent technical story while leaving room for further research.

12 Testing, CI and Repository Discipline

Backend tests were run through Pytest with the backend package on `PYTHONPATH`. Tests associated with the new Surface Atlas initially exposed stale assumptions about grid dimensions and an obsolete surface-field name. These were corrected to reflect the actual common comparison grid and current dataclass/API fields.

Generated experiment PNGs are intentionally ignored, while compact JSON artifacts needed by the runtime architecture are versioned. This separates reproducible research data from disposable render outputs.

A meaningful repository checkpoint was committed under the message:

```
feat: add American option surface atlas and PINN boundary diagnostics
```

and later frontend integration was committed as:

```
feat: integrate Surface Atlas into QuantLab workstation
```

13 Scientific Interpretation: What QuantLab Can and Cannot Claim

The project now supports several strong statements:

- numerical and learned American-option surfaces can be compared on a common state-time grid;
- PINN approximation error can be studied spatially rather than reduced to a single scalar metric;
- the training trajectory can be inspected at multiple checkpoints;
- residual error becomes substantially smaller and more spatially localized across the stored trajectory;
- meaningful residual structure appears near the numerically estimated stopping boundary in the converged regime.

However, the following claims should be avoided without additional experiments:

- projected CN is an exact American-option solution;
- all residual error near the stopping region is caused by free-boundary learning difficulty;
- a single Pearson distance-error correlation proves boundary concentration;
- the current one-parameter PINN generalizes as an operator across contract families;
- the current results establish superiority of PINNs over numerical methods.

14 Future Research Roadmap

14.1 Near-term: improve the existing experiment

The most natural future steps are not additional dashboard features but improvements in research validity:

1. Promote the strongest converged PINN checkpoint for default comparison while preserving early checkpoints for learning-dynamics analysis.
2. Increase projected-CN resolution and conduct grid-refinement studies.
3. Quantify sensitivity of the extracted free boundary to grid spacing and gap threshold.
4. Separate terminal/spatial-boundary residuals from stopping-boundary residuals if a publication-level attribution is required.

14.2 Numerical-method extensions

Potential baselines include trinomial lattices, explicit/implicit finite differences, adaptive meshes, Richardson extrapolation and Longstaff–Schwartz Monte Carlo.

The purpose should remain a structured comparison of approximation behaviour rather than a feature checklist.

14.3 Scientific-ML extensions

A major research direction is a parameter-conditioned approximation

$$V_\theta(S, K, r, q, \sigma, \tau),$$

which moves the problem toward learning an option-pricing operator instead of fitting a single PDE instance. DeepONet or related operator-learning architectures become more meaningful in that setting.

Uncertainty-aware extensions could include ensembles, Bayesian approximations, conformal calibration or explicit uncertainty over learned-model discrepancy.

14.4 Volatility extensions

Possible future work includes SABR, Heston calibration, local volatility, temporal SVI/SSVI stability, calibration uncertainty and more systematic surface-arbitrage stress tests.

14.5 Surface Atlas as flagship interface

Potential research-interface improvements include maturity slices, method-to-method boundary comparisons, parameter-sweep atlases, training animations, click-to-expand small multiples and an export-ready figure mode for papers and presentations.

Because Plotly 3D surfaces consume WebGL contexts, future visual expansion should preserve lazy mounting rather than rendering every possible surface simultaneously.

15 Recommended Documentation Structure

QuantLab should maintain three complementary documentation layers:

1. a concise repository `README.md`;
2. a living research journal recording dated decisions and future ideas;
3. this formal LaTeX/PDF report, which can grow into thesis or paper documentation.

A useful habit is to add a dated journal entry after each major experiment, especially when a scientific interpretation or architecture decision changes.

16 Current Checkpoint: 2 September 2026

At this checkpoint:

- QuantLab is portfolio-ready.
- The root page is the canonical Pricing Lab.
- Surface Atlas is a first-class module and is linked from the main page.
- American put surfaces compare CRR, projected CN and PINN outputs.
- PINN convergence checkpoints at 500, 1000, 2000 and 4000 epochs are available.
- Linear and log error-evolution views are implemented.
- The free-boundary ridge is displayed consistently in both scales.

- the WebGL multi-surface failure has been fixed using lazy plot mounting.
- frontend integration changes have been committed and pushed.

The next practical task is portfolio integration: use a strong Surface Atlas image as the visual anchor, provide Live Demo and GitHub links, summarize the four research modules, and present QuantLab as active computational-finance research software.

17 Closing Note to Future Me

QuantLab became interesting when it stopped being merely an option-pricing interface and started asking where approximations fail, how error changes during training, and what the geometry of numerical and learned solutions reveals. Preserve that direction.

When considering a new feature, ask three questions:

1. What research question does this answer?
2. What is the comparison or benchmark that makes the result meaningful?
3. Can the visualization distinguish a scientific phenomenon from an implementation artifact?

If those questions do not yet have good answers, the feature can wait.