AI in Quantitative Finance

What MFEs Need to Know Now

Henry Fuheng Wu

Executive Director, ML/AI & Senior Manager, Quant Analytics · Wells Fargo

UIUC MSFE Speaker Series · October 12, 2026 · Host: Prof. Liming Feng

The Road Here

Bloomberg · convertible bond pricing

WorldQuant · Asia-Pacific trade execution

Google · **Uber** · **Oracle** · generative AI

Wells Fargo · ML/AI and quant analytics

> *The tools changed at every step. The questions a good quant asks did not.*

PART I

Where AI Sits in Quantitative Finance

Valuation engines · Macro regime shifts · Bounded monotone residuals · Pricing surrogates · Factor representations

Where AI / ML Enters Quantitative Finance: Risk, Pricing, Trading

Area	Traditional Quant Approach	AI / ML Approach	Examples
Credit Risk	Logistic regression, scorecards	Gradient-boosted trees (XGBoost, LightGBM), neural networks	PD, default prediction, underwriting
Market Risk	VaR, stress testing, sensitivities	ML surrogate models, scenario generation	Fast pricing, nonlinear risk approximation
Pricing	PDE, Monte Carlo, trees	Neural-network approximators	Derivatives/MBS pricing acceleration
Mortgage / Prepayment	Hazard models, CPR/PSA	Gradient-boosted trees, survival ML, neural networks	Refi/prepayment/default prediction
Trading	Factor models, signals	ML alpha models, NLP	Signal generation, execution

Mortgage row, at scale: Sadhwani, Giesecke & Sirignano, "Deep Learning for Mortgage Risk," J. Financial Econometrics 2021 (120M US loans)

Inside Quant Finance — and Beyond It

Area	Traditional Quant Approach	AI / ML Approach	Examples
Portfolio Management	Mean-variance, factor models	ML forecasting + optimization	Return/risk forecasts
Fraud / AML	Rules	ML, anomaly detection, graph ML	Transaction monitoring
Operations	Rules/manual workflow	LLMs, agents, NLP	Documents, research, coding, workflow automation
Model Risk	Statistical validation	AI model validation	Explainability, drift, robustness

> Fraud detection and chatbots are important AI at financial firms, but **not every AI problem in finance is quantitative finance.** The MFE edge is where **finance + mathematics + data + computation** intersect.

Pattern 1: ML Sits *Inside* a Classical Financial Model

A mortgage-backed security still has a discounted-cash-flow structure:

$$P = \mathbb{E}^{\mathbb{Q}}\left[\sum_{t=1}^{T} D(0,t)\, CF_t\right], \qquad CF_t = f(\text{rates, prepayment, default, seasoning, } \ldots)$$

Traditional: parametric prepayment model

$$P(\text{prepay}) = \sigma(\beta_0 + \beta_1 \text{ refi incentive} + \beta_2 \text{ loan age} + \cdots)$$

Today: gradient boosting, survival models, neural networks

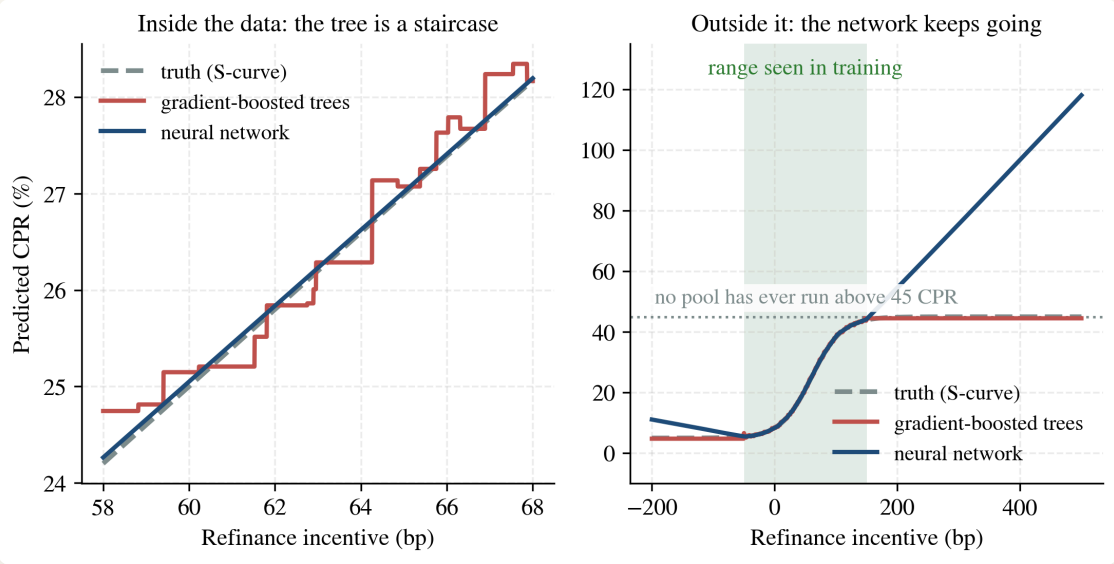
$$\hat{p}_t = f_\theta(X_t)$$

Financial theory → ML behavioral model → pricing engine → risk system → decision

> ML has **not** eliminated the financial model. You still need discount curves, cash-flow engines, OAS, duration, convexity, DV01, scenario analysis, and risk-neutral valuation.

Note: Historical prepayment models estimate physical borrower behavior (measure P). Market-consistent valuation requires calibration and risk adjustments (e.g., OAS); physical and pricing measures need not coincide.

Regime Change: When Macro Shocks Break Models



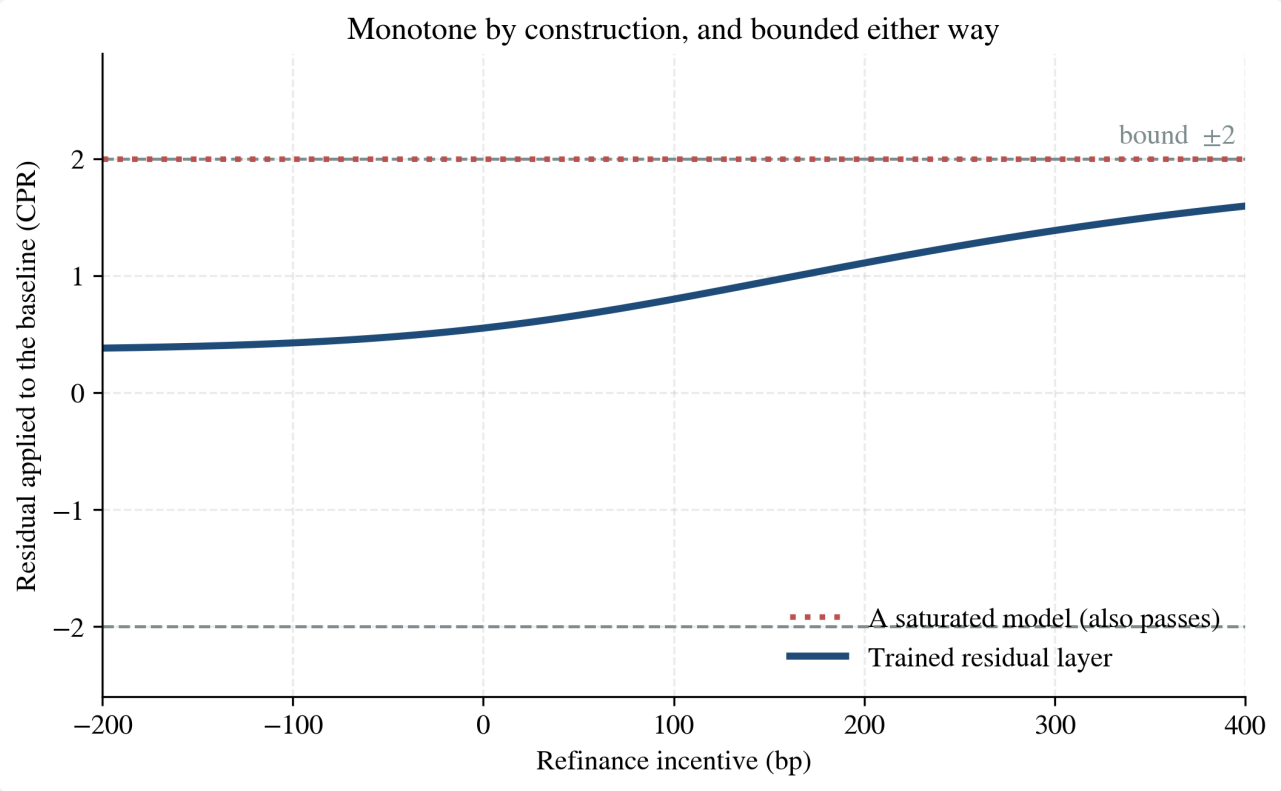
The 2022 Macro Shock

- From March 2022, the Fed ran the **fastest tightening campaign in four decades**; 30-year fixed mortgage rates surged from **about 3.2% to over 7%** (driven by policy rate hikes, 10-year Treasury yields, and mortgage basis spreads).
- **Regime changes like that push models far from the data they were fitted on.**

Two Black-Box Failure Modes

- **Trees (left)**: staircase behavior. Constant leaves yield $\frac{\partial \text{CPR}}{\partial r} = 0$ almost everywhere; breaks hedging Greeks (Δ, Γ).
- **Networks (right)**: smooth extrapolation into unphysical territory (e.g., predicting implausibly high CPRs far outside training scenarios).

Keep the Classical Model in Charge: A Bounded Residual

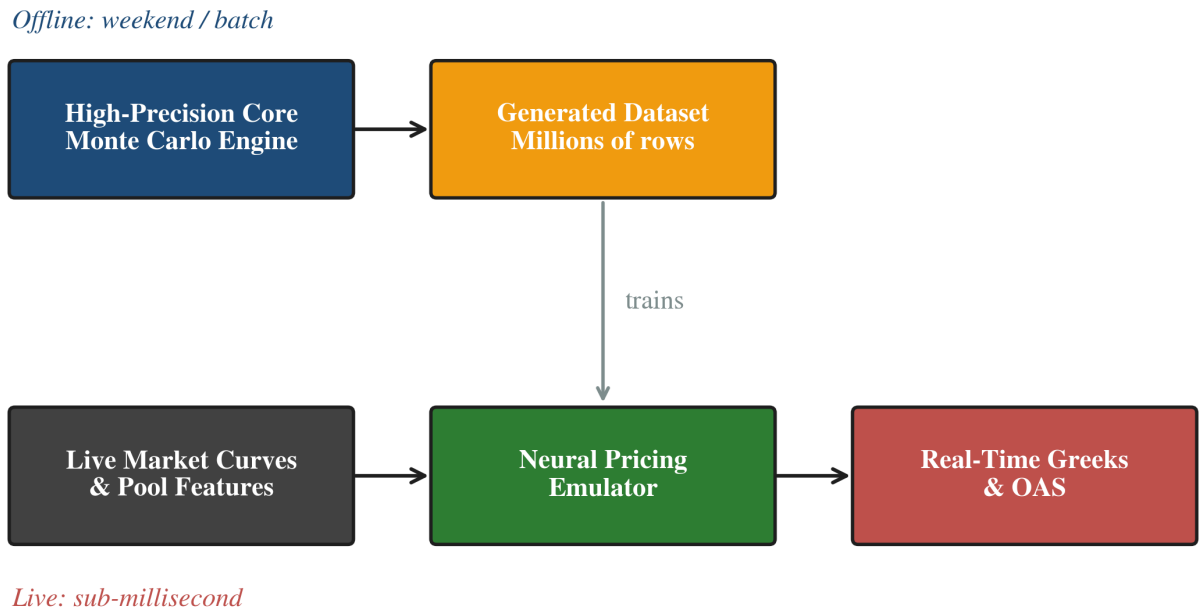


- An unconstrained network can predict that borrowers refinance *less* when rates drop.
- **Hybrid:** classical baseline + bounded monotone residual

$$\mathrm{CPR}(x) = \mathrm{clip}\Big(\mathrm{CPR}_{\mathrm{baseline}}(x) + r_\theta(x),\, 0,\, 1\Big) \quad \text{or} \quad \sigma\big(b(x) + h_\theta(x)\big)$$

- The residual $r_\theta(x)$ is **locally monotone** in refinance incentive by construction (non-negative branch weights) and **strictly bounded** ($|r_\theta| \le \delta$).

Pattern 2: Surrogate Models (Emulators)



If computing

$$V(x) = \mathbb{E}^{\mathbb{Q}}\left[\text{payoff} \mid X_0 = x\right]$$

takes seconds or minutes, and you must evaluate millions of scenarios:

1. Generate training data with the **validated** pricing model.
2. Learn $\hat{V}_\theta(x) \approx V(x)$.
3. Inference might take **milliseconds**.

> *The neural network did not replace financial mathematics. It learned the output surface created by financial mathematics.*

Pattern 3: Human-Designed Factors vs. Learned Representations

Barra: humans define the factors

$$r_i = \beta_{i,1} F_{\text{Value}} + \beta_{i,2} F_{\text{Size}} + \beta_{i,3} F_{\text{Momentum}} + \cdots + \epsilon_i$$

- Value, Size, Momentum, Volatility, Industry...
- Institutions paid vendors such as Barra for a **carefully engineered representation** of the market.

Deep learning: the model learns them

$$z = f_\theta(x)$$

- An embedding is a **learned coordinate system.**
- But the z_i may not be as easy to interpret as a Barra factor.

> **A learned representation does not automatically mean a better financial representation.**
> Human-designed factors → human + machine-learned factors → hybrid financial representations.

PART II

Four Stories from the Field

GPU batch divergence · Evaluating unstructured alpha · Label-free drift · Agent reliability

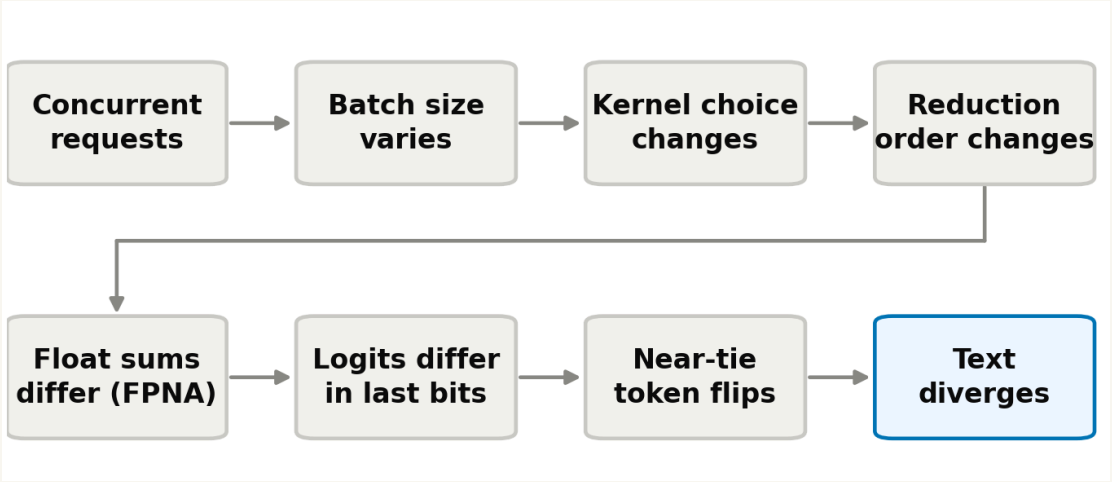
Story 1: A Quick Poll

> Same prompt. Same model.
> Temperature $= 0$. Greedy decoding.
> Ask it 100 times.

Same answer every time?

The Answer: No

`(1e8 + -1e8) + 3.14 = 3.14` `1e8 + (-1e8 + 3.14) = 0` (float32)



Verified in Python: The 3.14 Vanishes

Python Standard (`float64`)

```
$python
Python 3.12.12 | packaged by Anaconda, Inc. | (main, Oct 21 2025, 20:16:04) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> (1e8 + -1e8) + 3.14
3.14
>>> 1e8 + (-1e8 + 3.14)
3.1400000005960464
>>>
```

- First order: `3.14`
- Swapped order: `3.1400000005960464`
- **Myth**: *"Python has infinite precision!"* Only `int` does; `float` is C `double` (53-bit mantissa). Addition is **not associative**!

GPU / ML Standard (`numpy.float32`)

```
>>> import numpy as np
>>> a = np.float32(1e8)
>>> b = np.float32(-1e8)
>>> c = np.float32(3.14)
>>> print((a + b) + c)
3.14
>>> print(a + (b + c))
0.0
>>>
```

- First order: `3.14`
- Swapped order: `0.0` — 3.14 is completely lost in mantissa rounding!
- In `bfloat16` (7-bit mantissa), precision loss is even more severe!

> **(A + B) + C ≠ A + (B + C)**: In GPU Tensor Cores, dynamic batching alters parallel reduction order, flipping logits and diverging text.

Measured on My Laptop: Batch Variance in Action

PyTorch Test (RTX 5070 Ti, bf16)

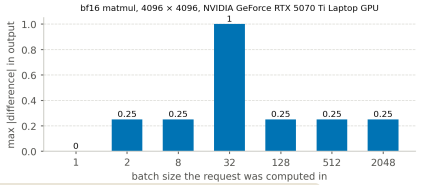
```
W = torch.randn(4096, 4096, device="cuda", dtype=torch.bfloat16)
X = torch.randn(2048, 4096, device="cuda", dtype=torch.bfloat16)
alone = X[:1] @ W  # row 0 alone

for B in (1, 2, 8, 32, 128, 512, 2048):
    in_batch = (X[:B] @ W)[:1]  # row 0 in batch B
    diff = (alone.float() - in_batch.float()).abs().max().item()
    print(f"batch {B:4d}: max diff = {diff:.4g}")

# 50 repeats at fixed batch 64: 1 distinct result!
```

(PyTorch 2.5, CUDA 12.4, RTX 5070 Ti, bf16 GEMM; Qwen2.5-0.5B-Instruct, greedy)

Empirical Findings



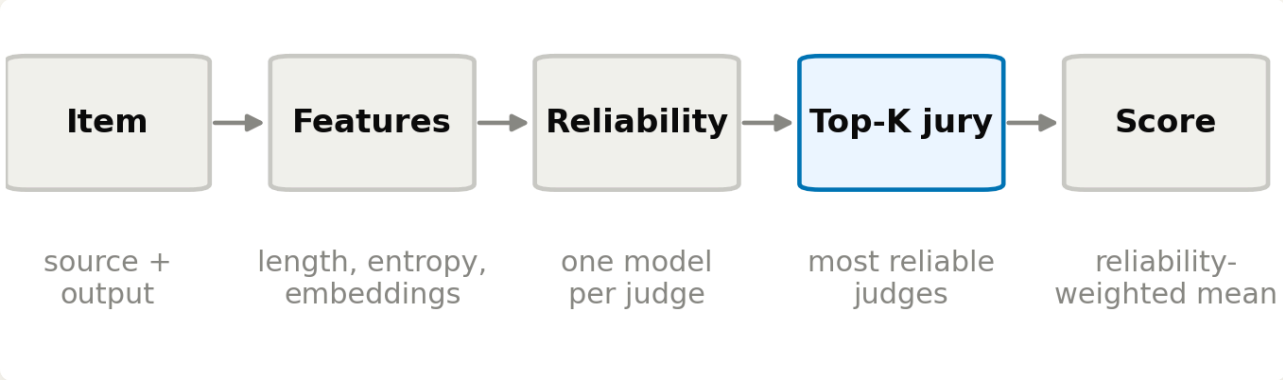
- Same row alone: `diff = 0.000` (identical)
- Same row in batch B: `diff` reaches `0.0078`
- 50 calls at fixed batch: `1 result` (deterministic kernel)
- Real LLM (Qwen2.5, greedy): 8 batch sizes → **8 divergent outputs**!

> In shared serving with dynamic batching: **concurrent traffic changes reduction order, causing greedy decoding to diverge.**

Making It Reproducible

- Remove sampling: temperature 0, greedy, no top-k / top-p
- Deterministic framework settings (PyTorch, cuBLAS)
- **Batch-invariant kernels** — same reduction order at any batch size
- Validate with **repeats**; separate decisions from prose
- Log model version, prompt fingerprint, decoding config

Story 2: Who Judges the Judge?



- **No market ticker**: Pricing models calibrate to quotes; **unstructured alpha** (10-Ks, FOMC, earnings) has no clean ticker at signal time.
- **LLM judge bias**: Scalable signal scoring, but vulnerable to length bias, position bias, and self-preference.
- **Condorcet trap**: Naive static voting **degrades** accuracy (τ: **0.63** $\rightarrow$ **0.58**) when noisy judges dilute the pool.
- **Jury-on-Demand**: Meta-model predicts item-level reliability $\rightarrow$ dynamic weighting achieves $\tau = \mathbf{0.68}$.

Li, Wang et al. (2025), arXiv:2512.01786

Non-linear metric trap (like a digital option payoff): Exact-match thresholding turns smooth token learning ($0.80 \rightarrow 0.95$) into an apparent discontinuous "emergence" jump (p^{10} : $0.11 \rightarrow 0.60$). Schaeffer, Miranda & Koyejo, NeurIPS 2023

Story 3: Seeing Drift Before Labels Arrive

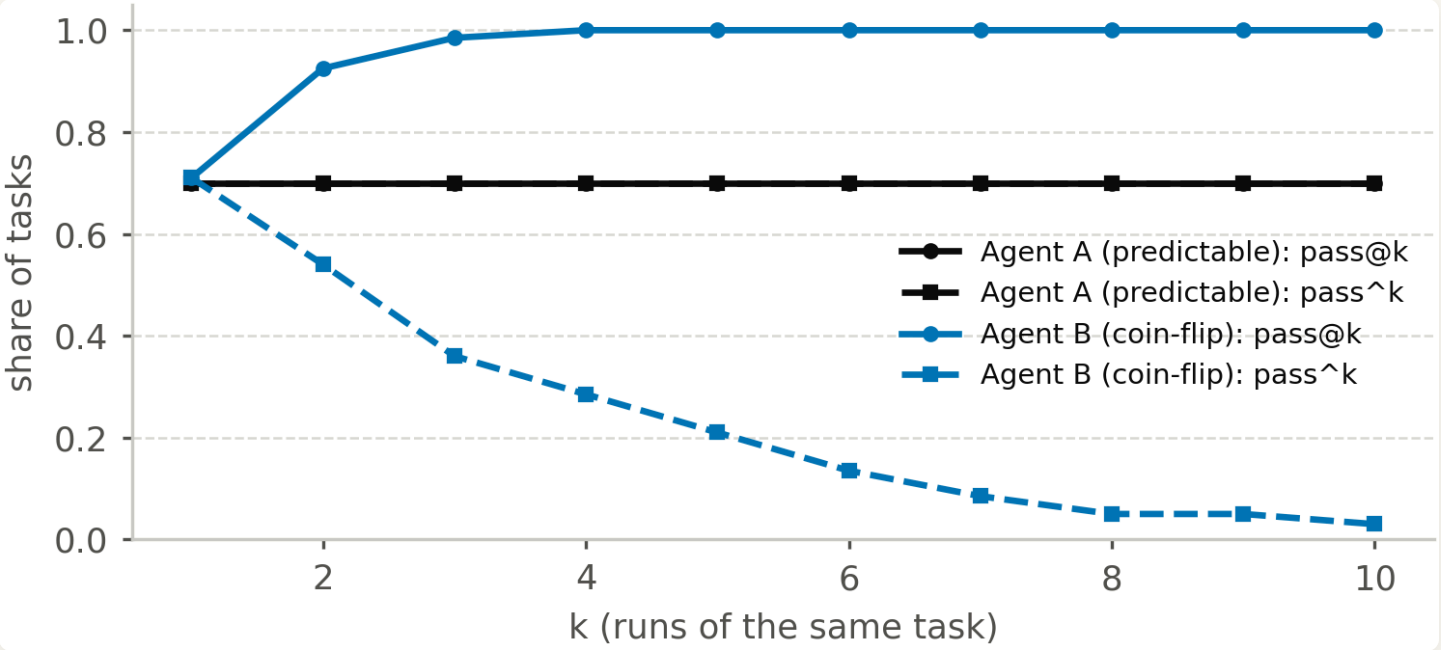


- Fraud labels arrive late — tail of **weeks to months** (card disputes run 120 days).
- Riskiest transactions are **blocked** → no clean label.
- The standard monitor is **PSI** on the score distribution. PSI detects change, not harm.
- **Synthetic stream (controlled benchmark):** PSI never exceeds **0.012** while true model performance drops (AUC-PR 0.42 → 0.26).

Fix: train a proxy for the model's loss; compare a **time-uniform** lower bound on the production high-loss rate to the source rate. Alarm fires **13 days** after shift. Podkopaev & Ramdas, ICLR 2022

Note: Sequential confidence bounds validly monitor the proxy-defined risk; detecting actual performance degradation assumes the proxy-to-loss relationship remains reliable under distribution shift.

Story 4: Agents — Capable Is Not Reliable



Rabanser et al. (2026), Princeton — consistency · robustness · predictability · safety

(Assuming independent Bernoulli trials per attempt for Agent B: $0.70^5 \approx 0.168$)

Two agents. Both 70% accurate.

A: same 70% of tasks, every time

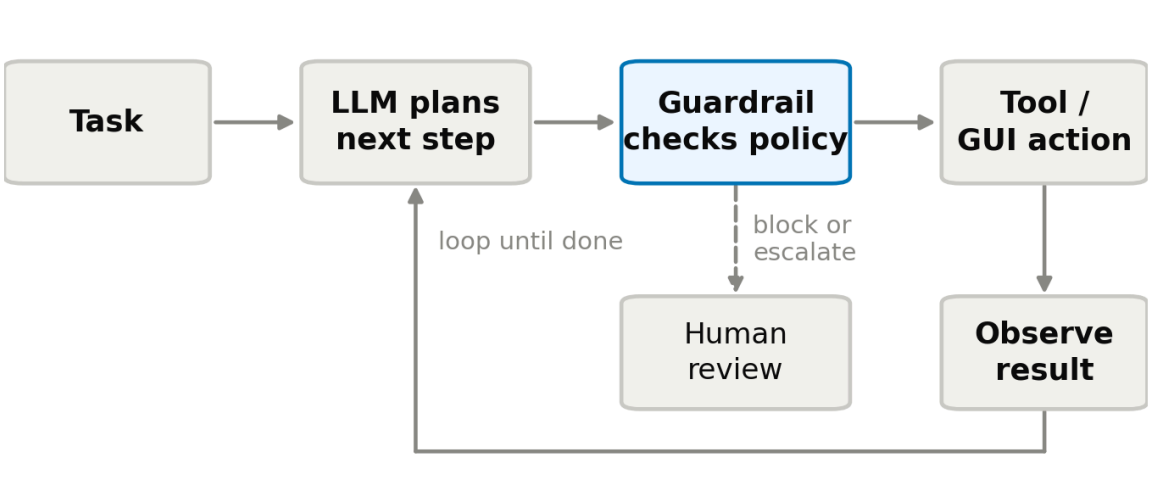
B: any task, 70% of the time

pass@k flatters B

pass^k is what production feels

pass^5: A = **0.70**, B = **0.17**

Guardrails Live Outside the Model



- **AgentPoison:** poison <0.1% of memory → >80% attack success, <1% drop in normal performance
- **ShieldAgent:** turn policy documents into verifiable rules; check every action

Chen et al., NeurIPS 2024 · Chen, Kang & Li, ICML 2025

Production AI Is a Constrained Optimization Problem

Academic models often optimize one metric. Production systems are multi-objective:

$$\max \{\text{accuracy}\} \quad \text{subject to} \quad \text{latency} < 100\ \text{ms}, \quad \text{cost} < \$0.001/\text{req}, \quad \text{memory} < 24\ \text{GB} \quad \text{(Illustrative SLA)}$$

...while also satisfying security, interpretability, reproducibility, and regulatory compliance.

Portfolio construction

$$\min_{w} \; w^T \Sigma w - \lambda \mu^T w$$

Machine learning

$$\min_{\theta} \; L(\theta)$$

Reinforcement learning

$$\max_{\pi} \; \mathbb{E}\Big[\sum_{t} \gamma^t r_t\Big]$$

> Define the state, define the objective, define the constraints, optimize. **Are we optimizing the right thing?**

PART III

What This Means for Your Career

Workflow compression · The junior rung · The 4-layer stack · You already know the math · What to bring

Generative AI Is Compressing the Quant Workflow

Before

research → derive → code → test → debug → document → deploy

Now

Quant → AI coding/research agent → code + tests + documentation → human / model governance

The valuable skill is shifting from

> “Can you write 500 lines of Python?”

toward

> “Can you define the right financial problem, formulate it mathematically, determine whether the result makes sense, and design a reliable system around it?”

The Junior Rung

> We once tested a system that could
> complete about 70% of a development workload.
> The team cheered.
> Soon after: stop hiring junior programmers.

— from my memoir, *Above the Clouds*

- The risk for students is not that experienced quants disappear.
- It is that the **junior rungs**, where experience used to be built, get automated first.
- So don't compete with the machine on what it does well.

Three People, One Mortgage Portfolio

Person A — ML engineer

“I'll train XGBoost.”

Person B — traditional quant

“I'll build a prepayment model and calculate OAS.”

Person C — modern financial engineer

“First I'll define the economic mechanism. I'll construct borrower-level features describing refinancing incentives and loan characteristics, estimate conditional prepayment probabilities, feed those predictions into the cash-flow engine, discount under consistent rate scenarios, calculate OAS and risk sensitivities, backtest, stress it under regime changes, and monitor it after deployment.”

Person C is the career position to target.

How the Strong Quant Has Changed

Twenty years ago

$$\boxed{\text{Finance} + \text{Mathematics} + \text{C++}}$$

Ten years ago

$$\boxed{\text{Finance} + \text{Mathematics} + \text{Python} + \text{Data}}$$

Today

$$\boxed{\text{Finance} + \text{Mathematics} + \text{ML/AI} + \text{Software Systems}}$$

> Your advantage as an MFE is **knowing what the numbers mean.**

A Four-Layer Stack

1. Financial Mathematics

Probability → stochastic processes → derivatives → fixed income → risk

Ito calculus, optimization, Monte Carlo, term-structure models, risk measures

2. Machine Learning

Regression → trees / boosting → deep learning → Transformers / LLMs

Don't start with LLMs and skip statistics.

3. Computing

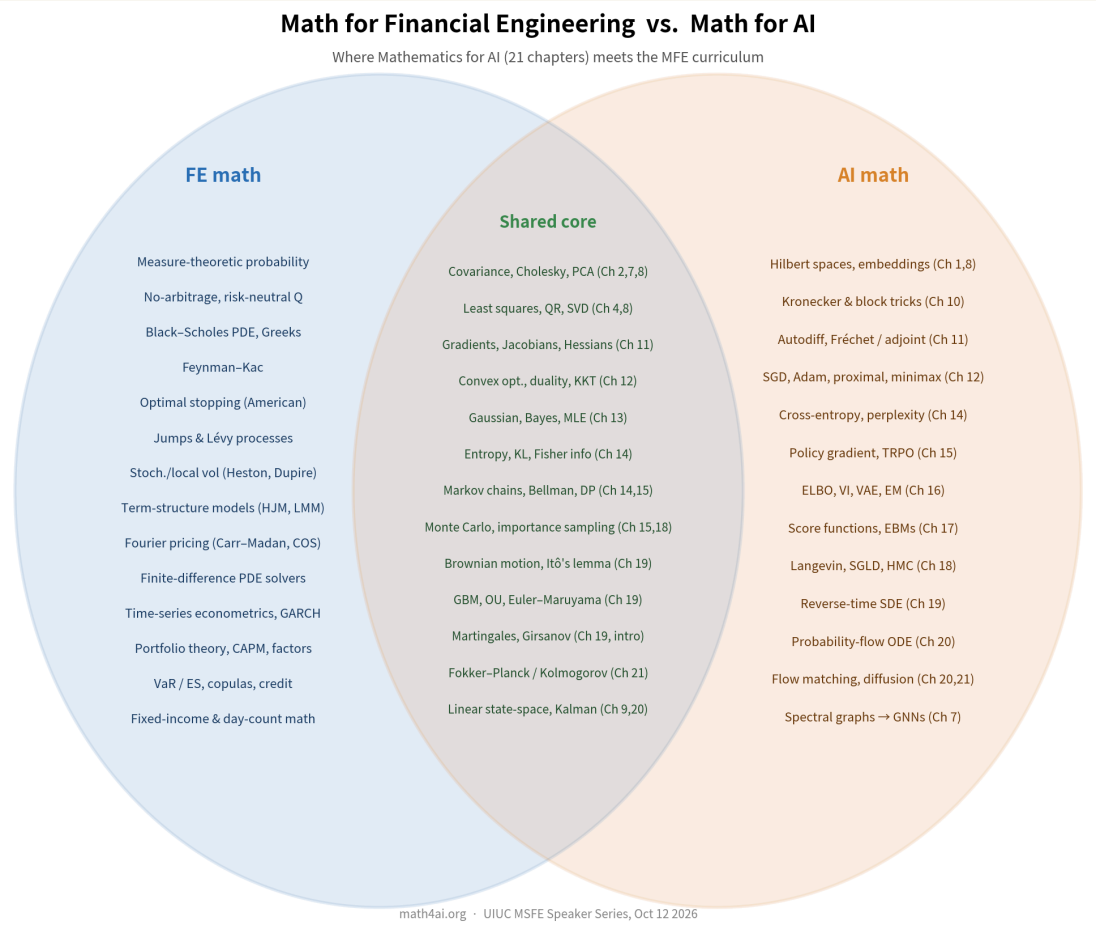
Python + SQL + C++ + Git + Linux + Cloud/GPU

A model that cannot be deployed, monitored, or reproduced has limited value in production.

4. Financial AI Engineering

Finance + Math + ML + Systems — understand all four well enough to **connect them.**

Two Circles, One Foundation: FE Math vs. AI Math



FE prices and hedges under Q · AI fits and samples under P · The shared core is massive.

Shared Mathematical Structures: The Rosetta Stone

You learned in MFE	It is now called in AI	Shared Mathematical Foundation
Binomial / Condorcet	Self-consistency voting	Majority voting error decays exponentially under independent trials
Gaussian sampling & Cholesky	Gaussian reparameterization (VAE)	Drawing latent Gaussian samples: $\mathbf{z} = \boldsymbol{\mu} + L\boldsymbol{\epsilon}$ (coordinate or Cholesky factor)
Yield-curve PCA / SVD	LoRA weight adaptation	Shared low-rank structure: capturing variation / updates in rank $r \ll d$
American options (Optimal stopping)	RL value iteration (Bellman)	Dynamic programming: immediate payoff vs. continuation value
SDEs & Drift modification	Diffusion models (reverse SDE)	Modifying SDE drift: market price of risk ($\mathbb{Q}$) vs. score $\nabla_x \log p_t$
Martingales (Ville's inequality)	Sequential drift testing (SHSD)	Time-uniform confidence sequences for non-asymptotic monitoring

AI did not replace mathematics. AI industrialized mathematics.

What to Bring

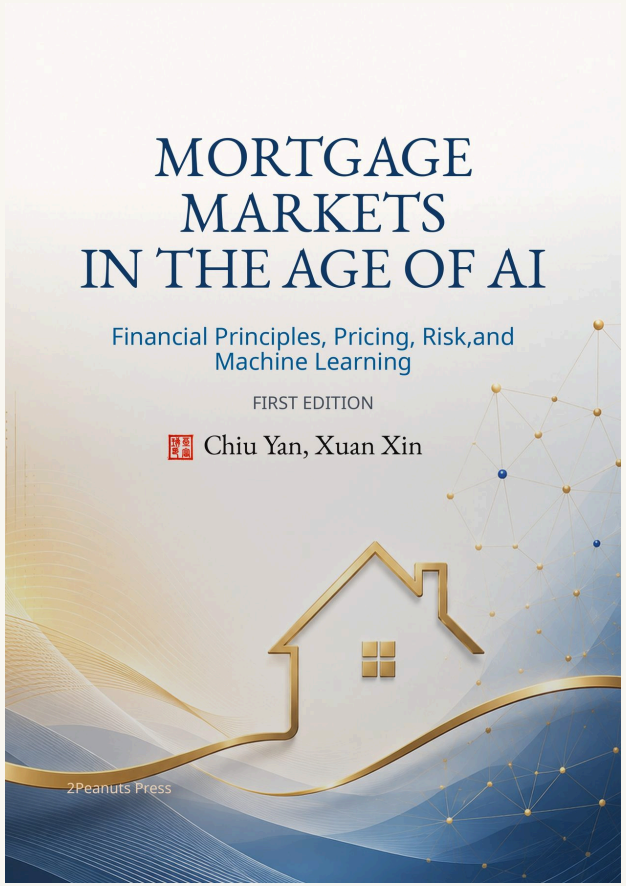
1. **Math you can use under pressure** — What is the state variable? What measure are we under? Does the output make economic sense?
2. **Software you can trust — and reproduce** — one project end to end that someone else can rerun: held-out data, a baseline, error bars, a README.
3. **Measurement as a habit** — never report one run; walk-forward validation, leakage, look-ahead bias.
4. **A domain you know deeply** — AI specialists are becoming common; people who understand AI *and* the financial product are much rarer.
5. **Judgment about risk** — enough finance to know when the model is wrong.

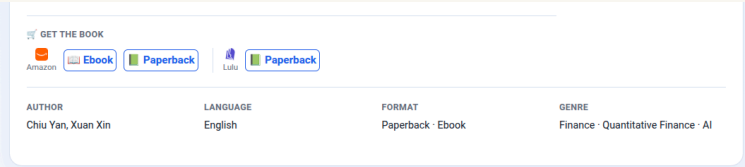
Mathematics tells you why it works.
AI tells you what can be learned from data.
Engineering determines whether it works in the real world.

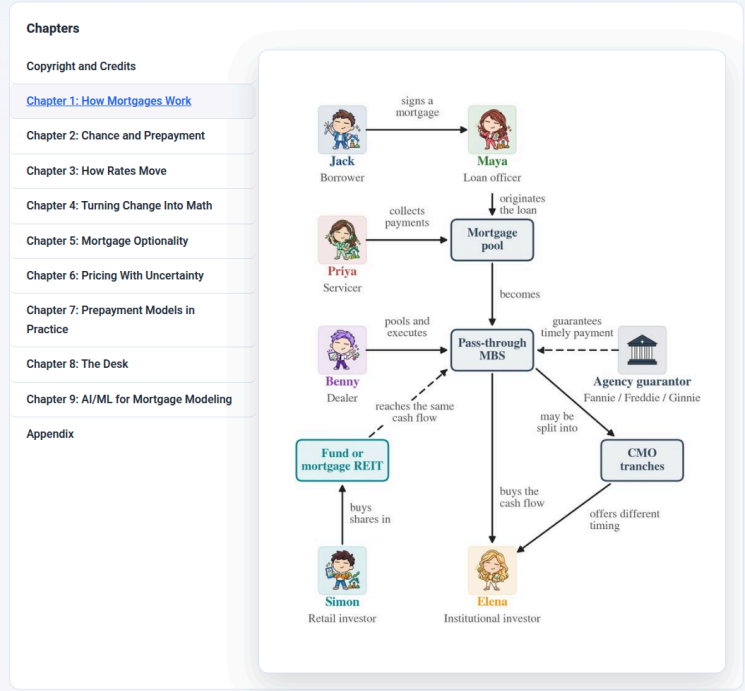
Do not become an "AI person who knows a little finance."
Become a financial engineer who is exceptionally good at using AI.

The field that taught you to price uncertainty is well placed for a world full of uncertain models.

Mortgage Markets in the Age of AI

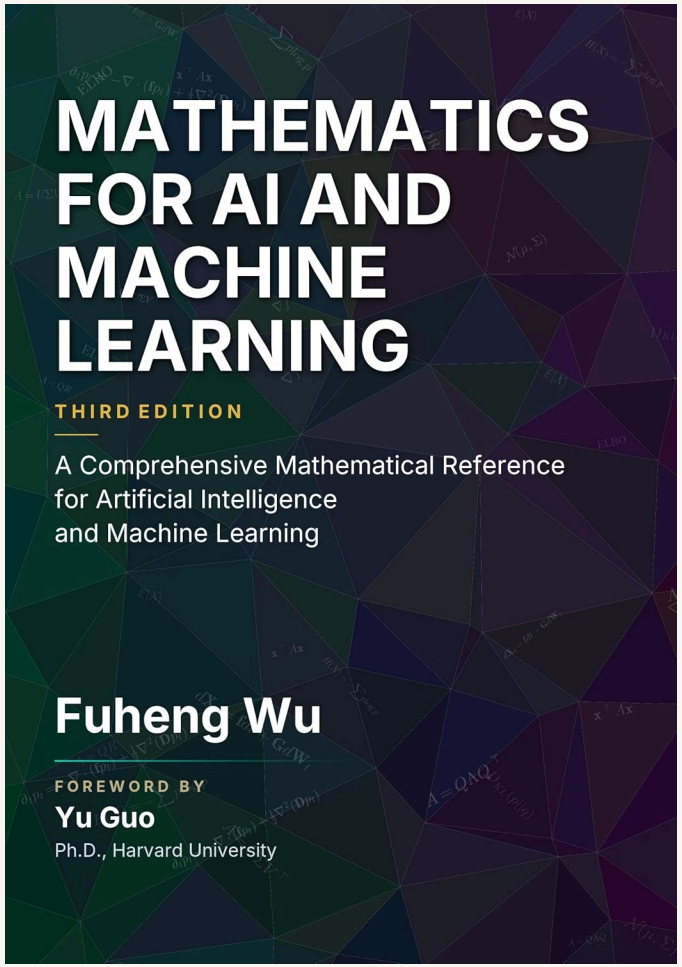






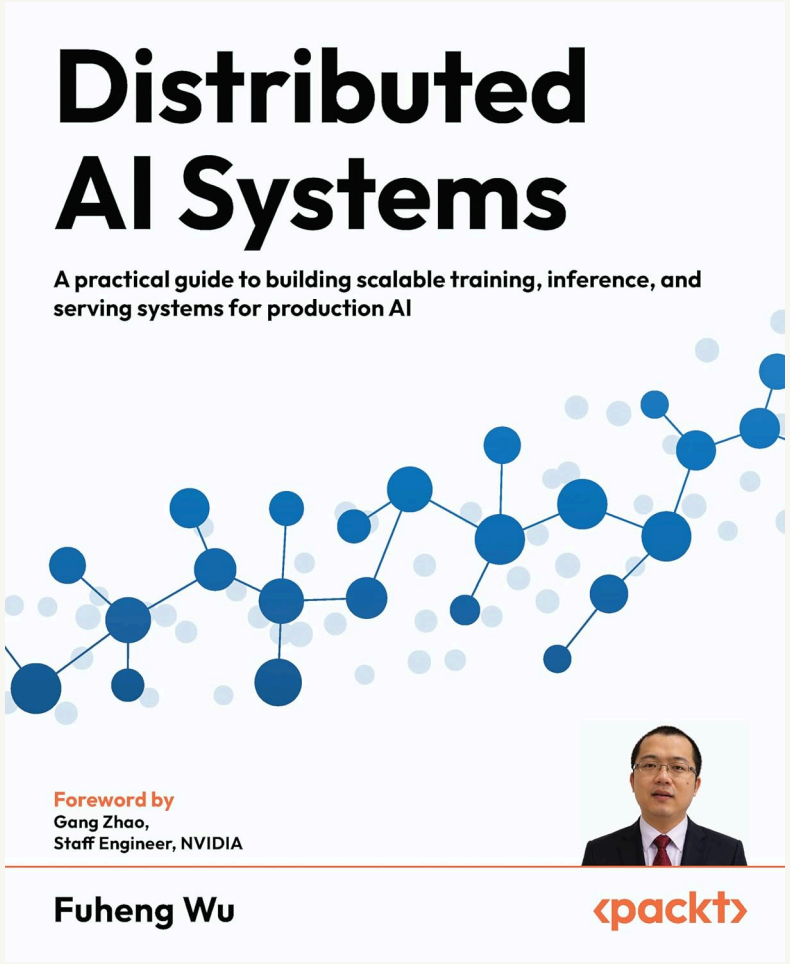
https://mortgage.2peanuts.com/en/

Mathematics for AI and Machine Learning



https://math4ai.org/

Distributed AI Systems



https://distaisys.com/

Thank You

Questions & Discussion?

📘 **Books & Code**

Mortgage Markets in the Age of AI

Mathematics for AI and Machine Learning · *Distributed AI Systems*

`pip install mortgagekit` · https://github.com/xuanxinjushi/mortgage

📬 **Connect**

LinkedIn: in/henrywoo/