Maths, Methods & Machines: Modern Market and Credit Risk

Forde Smith

6 January 2026

Foreword

I began compiling these notes while architecting the **Vannarho Risk Engine**—our open, programmable framework for intraday credit and market risk analytics. Pulling regulatory statutes, numerical methods, and compute-graph blueprints into one place helped me chart the engine's development roadmap and validate design choices against first-principles mathematics. Our platform is grounded in the QuantLib and Open Source Risk Engine repos; without this, our platform would not have been possible, so all credit goes to the teams working on those projects.

Over time the document grew beyond an internal white-paper into the volume you are reading now. If you are building, validating, or supervising modern risk systems, I hope the cross-links between theory, code, and regulation prove as useful to you as they have to the Vannarho team.

Because the scope is intentionally wide, the exposition remains necessarily high-level and often skims technical detail—think of it as a roadmap rather than a step-by-step build guide. Throughout the chapters you will see small call-outs indicating whether and how a feature has already been implemented in the Vannarho Risk Engine. Wherever no such note appears, consider that capability still on the roadmap. These call outs are quite high level and I plan to strengthen them.

While these notes were written first and foremost for the engineering team building the **Vannarho Risk Engine**, the breadth of topics may also be useful to:

- quantitative risk engineers and software architects integrating Basel frameworks;
- model–validation and model–risk teams seeking concise cross-references;
- regulatory-policy analysts who want computational context behind headline ratios;
- postgraduate students and researchers exploring practical implementation paths.

Any forward-looking statements regarding the future, features or timelines reflect *my own* views. This is not an academic paper! For feedback or collaboration, feel free to reach me at `forde@vannarho.com`.

Forde Smith

Contents

Version History

- v2 — 6 January 2026
- v1 — 19 June 2025

Document Overview

Part	Theme	Goal
I	Mathematical Foundations	Supply the quantitative "grammar" for later methods
II	Regulation & Economics	Translate Basel IV into computational specifications
III	Classical Methods	Revisit Monte–Carlo, PDE/FDM, copulas, credit portfolio theory
IV	Automatic Differentiation & Compute Graphs	Formalise adjoint methods, AAD on CPU & GPU
V	Machine Learning for Risk	Systematically benchmark ML vs conventional models
VI	Causal & Network Models	Extend to Bayesian, graphical & contagion frameworks
VII	Live–Risk Architecture	Bring it all together in an intraday capital & XVA stack
VIII	Frontier Topics & Outlook	Quantum, Neuromorphic, Explainable AI, regulatory horizon

Part I

Mathematical Foundations

Before we can explore the sophisticated models and trading strategies that make up modern quantitative finance, we need a firm grasp of the building blocks that support them. Part I—Foundations lays that groundwork. In the chapters ahead, we revisit fundamental ideas about how money grows over time, how risk and reward are measured, and how data can be organised so a computer can make sense of it. We will also meet the key players in today's markets and see how their actions shape prices and opportunities. By the end of this section you should feel comfortable with the language, mathematics and practical tools that will appear throughout the rest of this book (if you were not already).

1 Stochastic Calculus Refresher

Overview

Trading desks need risk software that can churn through thousands of price paths, produce Greeks, and satisfy regulators each night. To work, that software must stand on a clear picture of random price moves. The next pages revisit that picture.

We begin with Brownian motion, the basic random walk behind most models, then bolt on jump terms for sudden market shocks. Next we show how to shift from the real-world view of risk to the pricing view by tweaking the underlying probabilities. A short stop on directional derivatives sets up the automatic differentiation engine that later delivers fast Greeks. We then look at variance-breakdown tools that tell you whether market data or model choices drive P&L. Finally, we sketch the practical rules—time-step accuracy, variance reduction tricks, and GPU-friendly data layouts—that keep massive simulations both stable and lightning-quick.

Stochastic Calculus Refresher

Modern risk engines must turn out sub-millisecond Monte-Carlo Greeks, pathwise sensitivities for thousands of books, and a nightly, regulator-facing VaR back-test . These goals rest on one requirement above all others: a sound mathematical model of random motion. The

next pages recap the key ideas of stochastic calculus in a tight—but still practical—format. Along the way we flag every assumption that will later affect numerical stability and GPU throughput.

Brownian motion—the workhorse martingale On a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P})$ that meets the usual conditions , a *standard Brownian motion* W_t is a centred Gaussian process with independent increments and variance t. Its quadratic variation is

$$\langle W\rangle_t = \lim_{n\to\infty} \sum_{k=0}^{n-1} \Big(W_{t_{k+1}} - W_{t_k}\Big)^2 = t \quad \text{a.s.}$$

That single fact underpins most diffusion models in internal-model approvals . The Itô integral $\int_0^T \sigma_s \, \mathrm{d}W_s$ is the L^2 limit of simple processes . In practice we simulate it with steps $\Delta W_k \sim \mathcal{N}(0, \Delta t_k)$; their variance sets the benchmark for every later variance-reduction trick .

Beyond diffusion: Poisson jumps and Lévy flights Markets jump. Limit-order books gap. Trading can halt. A continuous martingale alone cannot match these realities . We add a *compound Poisson process*

$$J_t = \sum_{k=1}^{N_t} Y_k, \quad N_t \sim \text{Poisson}(\lambda t),$$

or move to the full Lévy–Khintchine class, which includes the heavy-tailed Lévy flights now common on energy and crypto desks . Our general model reads

$$\mathrm{d}X_t = \mu_t \, \mathrm{d}t + \sigma_t \, \mathrm{d}W_t + \int_{\mathbb{R}\setminus\{0\}} \gamma(t,z)\, \tilde{N}(\mathrm{d}t, \mathrm{d}z).$$

This form will drive the later Expected Shortfall proofs under wrong-way risk .

Radon–Nikodym densities and the Girsanov pivot Risk-neutral pricing needs a measure change from $\mathbb{P}$ to $\mathbb{Q}$. If an adapted process θ_t satisfies Novikov's condition, the density

$$\left.\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}\right|_{\mathcal{F}_t} = \exp\Big(-\tfrac{1}{2}\int_0^t \theta_s^2 \, \mathrm{d}s - \int_0^t \theta_s \, \mathrm{d}W_s\Big)$$

makes $W_t^{\mathbb{Q}} = W_t + \int_0^t \theta_s \, \mathrm{d}s$ a Brownian motion under $\mathbb{Q}$. The log-density is the likelihood ratio that underpins Girsanov; the Fenchel–Legendre transform instead yields the cumulant rate function used in large-deviation theory, not the Radon–Nikodym density itself. The same duality later links coherent risk measures to portfolio optimisers.

Gateaux differentials and adjoint calculus For a functional $F : \mathcal{X} \to \mathbb{R}$, the Gateaux derivative in direction h is

$$\delta F(x; h) = \frac{\partial}{\partial \varepsilon} F(x + \varepsilon h)\bigg|_{\varepsilon=0}.$$

This linear directional view is the theory behind adjoint-algorithmic differentiation (AAD). Part III turns it into a GPU-resident reverse-mode engine with tape sizes that follow exactly from the above expansion .

Sobol indices and model risk If forty calibration parameters feed an overnight-indexed-swap desk, P&L variance comes from more than market noise. Sobol total-order indices split the output variance,

$$\mathbb{V}[Y] = \sum_{i=1}^{d} V_i + \sum_{i<j} V_{ij} + \cdots,$$

where $V_S = \mathbb{V}\Big[\mathbb{E}[Y \mid X_S]\Big]$. The same Monte-Carlo machinery—run with common random numbers—delivers unbiased estimates . The stochastic-calculus base we have just set guarantees the advertised $\mathcal{O}(N^{-1/2})$ convergence as long as the time-step bias stays below the sampling noise.

Discretisation, variance reduction, and strong order Euler–Maruyama has strong order 1/2: pathwise error scales like $\Delta t^{1/2}$. Longstaff–Schwartz, multilevel Monte-Carlo, and stratified sampling pay off only after that leading bias is under control . Ignore this, and Itô's "$\mathbb{E}[\int] = \int \mathbb{E}[]$" fails, inflating variance instead of cutting it . Numerical stability starts with calculus, not with GPU occupancy.

Tensor operators and memory coalescing Store a cross-gamma cube as a fourth-order tensor $\Gamma_{ij,kl}$ and you need memory-coalesced access to keep the GPU warps busy . Kronecker products cover the separable case; full tensors cover the $n = 10^4$ portfolio. We use column-major order, matching `cublasLt`; stray from that and peak bandwidth drops by up to 80 % .

```
// GPU-friendly Euler step
__global__ void euler_step(double* __restrict x,
                           const double* __restrict drift,
                           const double* __restrict vol,
                           const double* __restrict z,
                           double dt, int n)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;
    if (i < n){
        x[i] += drift[i]*dt + vol[i]*sqrt(dt)*z[i];
    }
}
```

From Itô to Greeks. Itô's lemma

$$\mathrm{d}f(t, X_t) = \partial_t f \,\mathrm{d}t + \nabla_x f^{\mathsf{T}} \sigma \,\mathrm{d}W_t + \Big(\mu^{\mathsf{T}} \nabla_x f + \tfrac{1}{2}\mathrm{tr}\big(\sigma\sigma^{\mathsf{T}} \nabla_x^2 f\big)\Big)\mathrm{d}t$$

yields closed-form Greeks that anchor pathwise estimators. Chapter 7 combines them with the adjoint engine to deliver full-surface Delta, Vega, and cross-gamma—even for jump-to-default names .

Looking Ahead

Now that we have refreshed the tools of stochastic calculus—those rules for handling random motion and time-dependent uncertainty—it is worth stepping back to examine the mathematical bedrock that quietly supports every derivative and expectation we just manipulated. Stochastic calculus assumes we can assign "sizes" to random events and then change our point of view without breaking the logic. Measure theory provides the language for talking about those sizes, while the Radon–Nikodym idea tells us how to shift from one viewpoint to another in a controlled way. By revisiting these foundations, we equip ourselves to handle more sophisticated quantitative models with confidence.

2 Measure Theory & Radon–Nikodym for Quants

Overview

Before any coding tricks or GPU runs, we need a rock-solid way to talk about chance. Measure theory gives us that language. It treats "how likely" as a precise object rather than a hand-wave, letting us define every random path, jump, or shock we will later simulate. Once the rules for adding up probabilities are fixed, we can turn pricing formulas into well-behaved integrals and be sure our time steps shrink safely to zero.

The chapter then shows how to switch viewpoints—from the real-world measure to the risk-neutral one—using a likelihood ratio that keeps martingale pricing honest. It also links classical calculus ideas to modern sensitivity tools: the Gateaux derivative is the mathematical ancestor of the adjoint code that powers fast Greeks. Finally, it sketches how tensors speed up high-dimensional tasks, how Sobol indices diagnose fragile models, and why understanding Brownian motion and jump processes is essential for building stable, variance-controlled simulations.

Measure Theory & Radon–Nikodym for Quants

Before you fire up a Monte-Carlo engine or launch an adjoint run on a GPU, you need the language of modern probability. That language is measure theory. Regulators read it in back-testing reports, and counterparties expect it in exposure grids (Föllmer & Schied, 2011). We start with a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where $\mathcal{F}$ is the completed Borel σ-algebra and $\mathbb{P}$ carries unit mass (Klenke, 2014). Every random object we will meet—Brownian motion, compound Poisson jumps, or more exotic Lévy flights—must be $\mathcal{F}$-measurable (Shreve, 2004). Once this scaffold is set, the usual processes follow almost automatically (Applebaum, 2009).

From σ-additivity to stochastic integrals

Each pricing formula is an integral, usually disguised as an expectation (Baxter & Rennie, 1996; Björk, 2009). The dominated- and monotone-convergence theorems guarantee that discrete sums converge as the time step $\Delta t \to 0$ (Royden & Fitzpatrick, 2010). These results form the first line of defence against the variance explosions that appear when we discretise SDEs. A sound measure-theoretic base keeps numerical schemes stable; no clever low-discrepancy grid can rescue a model built on a leaky foundation (Glasserman, 2004; Owen, 1998).

Radon–Nikodym derivatives and the alchemy of measure change

Let $\mathbb{Q}$ be a risk-neutral measure that is absolutely continuous with respect to $\mathbb{P}$ (Harrison & Kreps, 1979; Delbaen & Schachermayer, 2006). The Radon–Nikodym derivative

$$Z \;\coloneqq\; \frac{d\mathbb{Q}}{d\mathbb{P}}$$

turns discounted prices into martingales by Girsanov's theorem (Girsanov, 1960; Kallenberg, 2002; Liptser & Shiryaev, 2001):

$$\mathbb{E}_{\mathbb{Q}}[X] \;=\; \mathbb{E}_{\mathbb{P}}[Z\,X], \qquad Z_t = \exp\!\Big\{-\tfrac{1}{2}\!\int_0^t \theta_s^2\,ds - \int_0^t \theta_s\,dW_s^{\mathbb{P}}\Big\}. \tag{1}$$

Desk traders call Z a *likelihood ratio*. It is positive and forms a $\mathbb{P}$-martingale, so $\mathbb{Q}$ is a true probability measure. Later, we will use (1) to build Monte-Carlo Greeks whose variance stays tame even in the wings (Hull, 2018).

Gateaux derivatives and the adjoint lineage

Risk engines live on sensitivities (Giles & Glasserman, 2006). The reverse-mode AD routines that run on GPU clusters trace back to the Gateaux derivative. For a functional $F\colon L^2(\Omega) \to \mathbb{R}$,

$$DF(X;h) \;=\; \lim_{\varepsilon\to 0} \frac{F(X+\varepsilon h) - F(X)}{\varepsilon}, \qquad h \in L^2(\Omega).$$

If F is an expectation, this reduces to linearity of the integral. For risk measures such as Expected Shortfall, the derivative yields the dual form used in FRTB capital optimisation (Acerbi & Tasche, 2002; Rockafellar & Uryasev, 2002). Thus, convex duality is not ivory-tower geometry; it is the link between risk numbers and optimisation code (Boyd & Vandenberghe, 2004; Griewank & Walther, 2008).

Tensor operators and high-dimensional quadrature

Stress tests create scenario cubes so large that plain Kronecker products fail (Bellman, 1957). Tensor operators $\mathcal{T} : L^2(\Omega)^{\otimes k} \to L^2(\Omega)$ give compact notation and a memory layout that maps to GPU tensor cores. Aligning the unfolding of $\mathcal{T}$ with the hardware stride yields coalesced loads and fast reductions (Higham, 2008; Papyan, 2019; NVIDIA, 2022).

```
__global__ void tensor_contract(
    const double* __restrict__ X,
    const double* __restrict__ W,
    double*       Y,
    int           n_paths, int n_dims)
{
    extern __shared__ double sdata[];
    int idx = threadIdx.x + blockIdx.x * blockDim.x;
    if (idx < n_paths * n_dims)
        sdata[threadIdx.x] = X[idx] * W[idx];
    // block-wide reduction ...
}
```

Sobol indices and model-form uncertainty

Good hardware is useless if the model itself is brittle. Sobol sensitivity indices split the output variance into main and interaction effects (Sobol', 2001). The method relies on orthogonal projections in $L^2(\Omega)$ and therefore on the measure-theoretic base just built (Saltelli et al., 2002). Precise indices need low-variance estimators; again, variance reduction and discretisation analysis go hand-in-hand (Owen, 2013).

Brownian motion, jumps, and Lévy flights revisited

Brownian motion, Poisson jumps, and Lévy flights are measures on path space endowed with the Skorokhod topology (Billingsley, 1999; Jacod & Shiryaev, 2003). From them we build Itô and Stratonovich integrals, the workhorses of later Monte-Carlo Greeks (Protter, 2005). Proving that Brownian quadratic variation equals t is a convergence statement in probability (Karatzas & Shreve, 1991). Showing that Lévy flights have infinite variance explains why Gaussian VaR often fails (Cont & Tankov, 2004).

From theory to practice

A production risk call may evaluate (1) millions of times on stratified Sobol grids, push Gateaux-rooted adjoints through tensor cores, and sample Lévy paths with variance control—all within a millisecond SLA (Anderson et al., 2017; Giles, 2015). Such speed is possible only because the mathematics guarantees stability, integrability, and differentiability at every step.

Implementation in VRE

The VRE engine embeds these concepts in its scenario generation and cross-asset models. Measure changes are handled via Radon–Nikodym densities when moving from market to pricing views, while adjoint routines compute Greeks alongside Monte-Carlo paths. Sobol-based variance analysis and GPU kernels follow the tensor layouts introduced here.

Summary

Measure theory underpins everything that follows. Radon–Nikodym derivatives turn heuristic measure shifts into firm density changes. Gateaux derivatives foreshadow the adjoint engines that power today's risk systems. Tensor operators and Sobol indices carry abstract integrals into scalable code. Brownian motion, jumps, and Lévy flights provide the dynamics, while variance-aware discretisation keeps numbers honest. With these tools, the rest of the book stands on solid ground.

Looking Ahead

Having just grounded ourselves in measure theory and the Radon–Nikodym idea, we now understand how continuous probabilities are stitched together and how one distribution can be re-expressed through another. The next step is turning that insight into fast, concrete calculations. Computers, and especially modern GPUs, are happiest when everything is phrased as arrays of numbers that can be shuffled, stacked, and multiplied in parallel. This is exactly the world of linear algebra and tensorisation. By translating our probabilistic concepts into matrices and higher-dimensional blocks, we unlock the hardware's ability to crunch vast models in real time—bridging theory with high-speed practice.

3 Linear Algebra & Tensorisation for GPUs

Overview

Modern risk systems churn through millions of simulated market paths every second. To stay both fast and accurate, they translate complex maths into large blocks of numbers—tensors—that GPUs can process in parallel. The section you are about to read explains how this translation works.

First, it shows how paths, jumps, and other random movements are packed into memory so that neighbouring GPU threads always read neighbouring data, avoiding slowdowns. Next, it introduces a flexible "tensor operator" that generalises the usual Kronecker product without exploding the data size; this operator underpins tasks such as covariance updates, regression steps, and measure changes used in pricing. The text then walks through kernel design, error control, and automatic differentiation, highlighting why storage order and block size matter for both speed and numerical stability. Finally, it connects these ideas to risk metrics like Value-at-Risk and sensitivity analysis, demonstrating how a well-structured tensor pipeline keeps large-scale Monte Carlo engines reliable and lightning-fast.

Linear Algebra and Tensorisation on GPUs

A modern risk engine must handle large linear-algebra workloads in real time while respecting the fine structure of Brownian motion, compound-Poisson jumps, and other Lévy flights that drive today's credit and market portfolios (Applebaum, 2009; Glasserman, 2004; Kirk & Warden, 2016). Analytically, these processes connect through the Radon–Nikodym derivative

$$\Lambda_T = \frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}} = \exp\Big(-\tfrac{1}{2}\int_0^T \|\theta_s\|^2\,\mathrm{d}s - \int_0^T \theta_s^\top \mathrm{d}W_s\Big),$$

which converts the real-world drift to the risk-neutral drift needed for pricing PDEs and Monte-Carlo kernels (Girsanov, 1960; Øksendal, 2003; Shreve, 2004). Numerically, each stochastic path, jump time, and Lévy increment must land in a tensor layout that plays well with GPU thread warps. Poor alignment produces bank conflicts and cache thrashing, which in turn cause mispriced XVA desks (NVIDIA, 2021).

Tensor operators: a GPU-first Kronecker generalisation A classic Kronecker product $\mathbf{A} \otimes \mathbf{B}$ quickly explodes to size $mn \times pq$ and becomes intractable once path stacking enters. We therefore use the tensor operator

$$\mathcal{T} :\ \mathbb{R}^{I_1\times\cdots\times I_d} \times \mathbb{R}^{J_1\times\cdots\times J_d} \longrightarrow \mathbb{R}^{I_1J_1\times\cdots\times I_dJ_d},$$

defined by

$$\Big[\mathcal{T}(\mathbf{A},\mathbf{B})\Big]_{(i_1,j_1),\ldots,(i_d,j_d)} = A_{i_1,\ldots,i_d}\,B_{j_1,\ldots,j_d}.$$

The operator supports partial contractions and adopts striding rules that map cleanly to CUDA block-row major order (Kirk & Warden, 2016; Kolda & Bader, 2009). Adding the path index makes the same contraction the workhorse for covariance updates and Longstaff–Schwartz regression in exposure simulation (Hull, 2018).

CPU vs GPU execution lanes Not every workload belongs on a GPU. Small grids, calibration loops, and debugging runs benefit from CPU SIMD (Auto/NEON/AVX2/AVX512) with tight cache reuse and no launch overhead. Large batched tensor ops, Monte-Carlo path blocks, and adjoint sweeps with wide shocks belong on GPU, where block-major storage and coalesced loads dominate the cost. VRE exposes target switches (Auto/Off/Strict) for CPU vs GPU, SIMD, and compiled-kernel choices so teams can pin critical runs or let the engine pick the fastest lane at runtime.

Conditional expectation on device (WIP) CE regressions currently run on CPU for auditability and small-problem efficiency. Phase E design pushes CE onto GPU for large path×time blocks: forward-only AAD on device, tiled regressions, and host/device caches to reuse basis functions. A config switch (Auto/Off/Strict) will gate GPU CE; default keeps CPU as the reference with GPU CE compared for accuracy before promotion.

Memory-coalesced kernels GPUs reach peak bandwidth only when neighbouring threads read neighbouring data (NVIDIA, 2021). For a tensor $\mathcal{X} \in \mathbb{R}^{N\times P\times d}$ with d factors, we store (n, p, d) in Morton (Z-curve) order so that nearby paths sit in adjacent cache lines. The GEMM-like kernel below sustains more than 0.9 of roof-line bandwidth on an A100 (NVIDIA, 2020) and drives Euler–Maruyama stepping under the $\mathbb{P}$ measure (Kloeden & Platen, 1992):

```
__global__ void tensor_gemm(
        const double* __restrict__ X,
        const double* __restrict__ A,
        double* __restrict__ Y,
        int N, int P, int d) {

    int n    = (blockIdx.x << 5) + threadIdx.x;
    int p    = (blockIdx.y << 5) + threadIdx.y;

    __shared__ double As[32][33];        // +1 column avoids bank conflicts
    double y_local = 0.0;

    for (int k = 0; k < d; k += 32) {
        As[threadIdx.y][threadIdx.x] =
            A[(k + threadIdx.y) * d + (k + threadIdx.x)];
        __syncthreads();

        #pragma unroll
        for (int t = 0; t < 32; ++t) {
            double x_val = X[((k + t) * P + p) * N + n];
            y_local += x_val * As[threadIdx.y][t];
        }
        __syncthreads();
    }
    Y[p * N + n] = y_local;
}
```

A small change in storage order or block size would break the drift-neutral update $X_{t+\Delta t} = X_t + \mu\,\Delta t + \sigma\sqrt{\Delta t}\,Z$ (Øksendal, 2003).

Convex duality and risk-measure algebra Value-at-Risk and Expected Shortfall admit convex dual forms solved by conjugate-gradient or primal–dual Newton methods on exactly these tensors (Föllmer & Schied, 2016; Rockafellar, 1970). The dual ascent needs repeated products $K\mathbf{v}$, where K bundles likelihood ratios across paths. A GPU tensor pipeline removes the traditional $\mathcal{O}(N^2)$ PCIe bottleneck (NVIDIA, 2020).

Sobol indices and discretisation error Model uncertainty enters through drift approximation, truncated Lévy tails, and antithetic pairing (Glasserman, 2004). Sobol indices

$$S_i = \frac{\operatorname{Var}\big(\mathbb{E}[Y \mid X_i]\big)}{\operatorname{Var}(Y)}$$

separate factor contributions provided the Monte-Carlo tensor keeps the X_i orthogonal (Sobol, 1993; Saltelli et al., 2008). Keeping each factor in its own memory slice lets us

compute these indices in a single pass and monitor the $\mathcal{O}(\Delta t^{1/2})$ discretisation error before adding control variates (Kloeden & Platen, 1992).

Adjoint calculus and Gateaux derivatives Automatic differentiation for Greeks amounts to pushing a Gateaux derivative through kernel graphs (Baydin et al., 2018; Giles, 2008). The adjoint of the tensor GEMM reverses index order while reusing the same tiles. Formally,

$$\mathrm{d}\mathcal{F}(\mathbf{X})[\mathbf{H}] = \lim_{\varepsilon \to 0} \frac{\mathcal{F}(\mathbf{X} + \varepsilon \mathbf{H}) - \mathcal{F}(\mathbf{X})}{\varepsilon},$$

which guarantees that no silent transposition error pollutes delta–gamma estimates.

Variance reduction through error control Variance reduction starts by keeping discretisation error below stochastic variance. The strong error bound for Euler,

$$\left\| X_T - \widehat{X}_T^{(\Delta t)} \right\|_2 = \mathcal{O}(\Delta t^{1/2}),$$

turns step-size choice into a root-finding task on the GPU tensor of residuals (Giles, 2015).

Numerical stability Each determinant bound and Lipschitz constant in Part I maps to overflow budgets and fused-multiply-add decisions in CUDA PTX (Higham, 2002). The Cholesky sampling of correlated Lévy increments relies on condition numbers preserved by Householder-triangularisation kernels tuned for half-precision Tensor Cores (Higham, 2008; NVIDIA, 2020). Mathematical rigour upstream shields the risk platform from numerical failure downstream.

Implementation in VRE

The engine's CUDA kernels adhere to these stability constraints. Tensor operations in the analytics library use coalesced layouts, and the computation graph reuses them for reverse-mode differentiation. Configuration files expose step-size choices so variance reduction and throughput remain balanced.

Summary

Linear algebra and tensorisation form the bridge between stochastic calculus and GPU hardware. By tying Radon–Nikodym measure changes, convex-duality optimisation, Sobol sensitivity, and Gateaux-based adjoints to memory-coalesced kernels, we deliver risk analytics that are fast, accurate, and mathematically transparent (Glasserman, 2004; Higham, 2008).

Looking Ahead

Now that we're comfortable seeing data as collections of vectors and matrices—and we've learned how GPUs shuffle these tensors around with astonishing speed—we have the raw expressive power we need for modern computation. But raw power alone doesn't tell us which direction to push our numbers so that a model learns, a robot balances, or a schedule improves. This is where optimisation steps in. By framing our goals as "find the best possible value" questions and using the geometric insights of convex analysis, we convert those lightning-fast tensor operations into purposeful progress toward solutions that are both efficient and reliable.

4 Optimisation & Convex Analysis

Overview

This section sets up the mathematical toolkit for the rest of the book. We begin with convex analysis to frame portfolio constraints and penalties as tractable objects, making optimisation problems easier. We then show how risk measures behave and how duality turns messy tasks into simpler ones. Sensitivity analysis with Gateaux derivatives underpins automatic differentiation used for Monte Carlo Greeks. Moving from real-world to risk-neutral probabilities is handled with the Radon–Nikodym derivative, keeping models arbitrage-free. We cast coherent risk measures like CVaR as convex programs, enabling efficient algorithms with guaranteed convergence. To cut simulation cost we control discretisation bias and apply multilevel Monte Carlo. Parameter importance is measured with Sobol indices, turning model risk into another optimisation task. Finally, we show how high-dimensional calculations map to GPU-friendly tensor operations, providing the speed needed in live trading. Together these ideas form a coherent bridge between theory, simulation, and hardware for modern risk management.

Optimisation & Convex Analysis

Convex analysis offers a common vocabulary for market-risk attribution, model calibration, and high-speed simulation. Because most portfolio constraints (e.g., budget, leverage, or regulatory capital) and penalty terms (e.g., short-selling costs) can be written as convex sets or convex functionals, the resulting optimisation problems enjoy strong existence and duality results. A risk measure $\rho : \mathcal{X} \to \mathbb{R}$ is convex if

$$\rho\Big(\lambda X + (1-\lambda)Y\Big) \le \lambda\rho(X) + (1-\lambda)\rho(Y), \quad \forall\, X, Y \in \mathcal{X},\ \lambda \in [0,1],$$

meaning diversification never raises risk . The Legendre–Fenchel transform

$$\rho^\star(Z) = \sup_{X \in \mathcal{X}} \Big\{ \langle Z, X \rangle - \rho(X) \Big\}$$

moves a possibly messy primal functional into its (often simpler) dual. Under mild continuity assumptions, strong duality states that primal and dual optima coincide, so we can do sensitivity analysis on the cheaper side. In practice, $\rho^\star$ frequently has a closed form that is easy to differentiate, linking risk metrics to the numerical routines used later .

Gateaux derivatives matter because every gradient, Jacobian, or adjoint we will need is a directional derivative on an infinite-dimensional space. For a functional $F : \mathcal{X} \to \mathbb{R}$,

$$\delta F(X; H) = \lim_{\epsilon \to 0} \frac{F(X + \epsilon H) - F(X)}{\epsilon},$$

and the map $H \mapsto \delta F(X; H)$ is linear and continuous if F is Gateaux differentiable. Reverse-mode automatic differentiation, used to compute Monte-Carlo Greeks, is a discrete version

of this idea (Hull, 2018). Because the same adjoint sensitivities later run inside GPU kernels, sound analysis here protects numerical stability downstream.

Probabilistic foundations. Stochastic optimisation in finance changes measure from the historical law $\mathbb{P}$ to the risk-neutral law $\mathbb{Q}$. The Radon–Nikodym derivative

$$\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}} = \exp\Big(-\tfrac{1}{2}\int_0^T \sigma_t^2\,\mathrm{d}t - \int_0^T \sigma_t\,\mathrm{d}W_t^{\mathbb{P}}\Big)$$

formalises the switch and justifies the martingale property of discounted prices . Brownian motion is the baseline; jump processes and Lévy flights are compared to it when we fit tail-risk parameters. The Girsanov theorem confirms that the density above is a true martingale, ruling out arbitrage. The same calculus reappears when we compute pathwise Greeks in Part III.

Convex programs as risk engines. Coherent risk measures such as Conditional Value at Risk (CVaR) solve the convex program

$$\operatorname*{minimise}_{\alpha\in\mathbb{R}}\Big\{\alpha + \tfrac{1}{1-\beta}\,\mathbb{E}_{\mathbb{Q}}\big[(L-\alpha)_+\big]\Big\},$$

with loss L and confidence level $\beta \in (0,1)$. Under mild moment assumptions the problem is strongly convex, so proximal-gradient and interior-point methods converge . Because CVaR is also Lipschitz, variance-reduced stochastic gradients reach the optimal square-root complexity for unbiased Monte-Carlo estimators .

Variance reduction via discretisation control. Simulation introduces discretisation error. For an N-step Euler–Maruyama path $\widehat{X}^{(N)}$,

$$\big|\mathbb{E}f(\widehat{X}_T^{(N)}) - \mathbb{E}f(X_T)\big| = \mathcal{O}(N^{-1}),$$

so variance reduction starts with controlling this bias . Multilevel Monte-Carlo (MLMC) uses

$$\mathbb{E}f(\widehat{X}_T^{(N_0)}) = \mathbb{E}f(\widehat{X}_T^{(N_L)}) + \sum_{\ell=1}^{L}\mathbb{E}\big[f(\widehat{X}_T^{(N_{\ell-1})}) - f(\widehat{X}_T^{(N_\ell)})\big],$$

to separate bias and variance across finer grids . Because Brownian-bridge and jump-adapted grids share the same convex backbone, one MLMC estimator covers both diffusions and Lévy models.

Sobol indices and model-form uncertainty. Sensitivity analysis links optimisation to model risk. For a loss functional $L(\theta)$ with parameters $\theta = (\theta_1,\dots,\theta_d)$, the Sobol index of factor j is

$$S_j = \frac{\operatorname{Var}[\mathbb{E}(L \mid \theta_j)]}{\operatorname{Var}(L)}.$$

Convex duality shows that the map $\theta \mapsto S_j$ itself solves a constrained optimisation in L^2. If the loss model is differentiable, we can replace gradient-free estimates with adjoint ones, giving about a ten-fold speed-up in practice.

Tensor operators and GPU tensorisation. High-dimensional problems need more than pen-and-paper convexity. Tensor operators $\mathcal{T} : \mathbb{R}^{n_1 \times \cdots \times n_k} \to \mathbb{R}^{m_1 \times \cdots \times m_\ell}$ extend the Kronecker product:

$$(\mathcal{T}X)_{i_1 \cdots i_\ell} = T_{i_1 \cdots i_\ell}^{\;\; j_1 \cdots j_k} X_{j_1 \cdots j_k}.$$

On NVIDIA-type GPUs, we get fast kernels when the last dimension is contiguous, so tensor maps must preserve stride order . The CUDA-style code below transposes a rank-4 tensor while keeping memory coalesced:

```
// coalesced tensor transpose: (i,j,k,l) -> (k,l,i,j)
template<int I, int J, int K, int L>
__global__ void tensor4_transpose(float* __restrict__ out,
                                  const float* __restrict__ in)
{
    int idx  = blockIdx.x * blockDim.x + threadIdx.x;
    int size = I * J * K * L;
    if (idx >= size) return;

    int i =  idx % I;
    int j = (idx / I) % J;
    int k = (idx / (I*J)) % K;
    int l =  idx / (I*J*K);

    int tgt = ((k*L + l)*I + i)*J + j;  // coalesced write
    out[tgt] = in[idx];
}
```

Because first-order optimality reduces to batched tensor contractions, this layout speeds up stochastic-gradient methods.

Putting it all together. A live trading system might rebalance a portfolio against CVaR under Lévy dynamics, differentiating through the Radon–Nikodym change of measure and a GPU-accelerated MLMC engine. The adjoint chain rule pushes Gateaux derivatives backward through a web of tensor operators, while Sobol indices measure remaining model risk. Convexity guarantees each sub-problem has a unique solution, and the variance analysis above sets the Monte-Carlo budget for any chosen precision.

Implementation in VRE

In practice, VRE stores scenario cubes in tensor form and executes these optimisation routines on GPU when available. Block sizes and stride order match the kernels above, enabling pathwise adjoint sensitivities and CVaR optimisations to run within production windows.

Summary

Convex analysis links probabilistic change-of-measure tools to stable, GPU-accelerated optimisation routines. Brownian motion, jump processes, and Lévy flights drive the models; Radon–Nikodym densities embed them in risk-neutral valuation; Gateaux derivatives support the adjoint calculus for Monte-Carlo Greeks; Sobol indices become variance-based optimisation tasks; and careful bias-variance control feeds variance-reduction schemes. Together these ideas provide the mathematical foundation for the rest of the book.

Looking Ahead

After exploring how optimisation and convex analysis guide us to the single "best" answer, it is natural to ask how secure that answer really is. In everyday situations, the numbers we feed into a model are rarely perfect—measurements vary, conditions drift, and small oversights can ripple through the results. To understand how these imperfections might tilt our carefully chosen solution, we turn to uncertainty quantification and sensitivity analysis. Together, they let us probe which inputs matter most, see where our conclusions are sturdy or fragile, and ultimately gain confidence that our decisions remain sound when real-world wiggles appear.

5 Uncertainty Quantification & Sensitivity Analysis

Overview

Risk managers need clear answers to three questions: how large is the model error, which inputs drive it, and how quickly can we measure both? This section assembles the tools. First, we move beyond the textbook Brownian motion and allow sudden jumps, mirroring real market shocks. Second, we show how to swap from the "pricing" measure to the "hedging" measure in a legally sound way, turning capital rules into optimisation tasks rather than guesswork. Third, we tidy up Monte-Carlo simulations by separating bias from variance; only after both are controlled do fancy variance-reduction tricks add value. Global sensitivity metrics then reveal which assumptions most influence results, focusing effort where it counts. Reverse-mode differentiation supplies every Greek in one sweep, and streamlined tensor maths lets graphics cards crunch the numbers fast enough for daily reports. Together, these ideas form a practical blueprint for robust uncertainty quantification.

Uncertainty Quantification and Sensitivity Analysis

Rigorous uncertainty quantification (UQ) underpins modern risk-management rules (Smith, 2014). The same Itô calculus introduced in Part I now guides the Monte-Carlo "Greeks" that drive regulatory-capital numbers (Glasserman, 2004; Øksendal, 2003). Yet a plain Brownian motion $W_t \in \mathbb{R}^d$ seldom captures trading-book reality. Basel modellability tests now favour jump processes, compound Poisson cascades, and Lévy flights because these models reproduce the skew and fat tails seen in the data (Cont & Tankov, 2004; Hull, 2018). Throughout this chapter we therefore adopt the general semimartingale

$$X_t = X_0 + \underbrace{\int_0^t b_s \,\mathrm{d}s + \int_0^t \sigma_s \,\mathrm{d}W_s}_{\text{diffusion}} + \underbrace{\int_0^t \int_{\mathbb{R}} \gamma_s(z)\, \tilde{N}(\mathrm{d}s, \mathrm{d}z)}_{\text{jumps}},$$

whose variance drives every later UQ metric.

Change of Measure and the Radon–Nikodym Lens

Many pricing desks quote a figure under one measure but hedge under another. The Radon–Nikodym derivative makes this switch legal. For an equivalent martingale measure $\mathbb{Q}$ that is absolutely continuous with respect to $\mathbb{P}$, the density

$$\Lambda_t = \frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}\Big|_{\mathcal{F}_t} = \exp\Big(-\int_0^t \theta_s^\top \mathrm{d}W_s - \tfrac{1}{2}\int_0^t \|\theta_s\|^2 \mathrm{d}s\Big)$$

is a positive martingale that removes drifts under $\mathbb{Q}$ (Shreve, 2004). Convex duality then rewrites a risk measure in the form

$$\rho(X) = \sup_{\mathbb{Q}\in\mathcal{Q}} \Big\{ \mathbb{E}_{\mathbb{Q}}[-X] - \alpha(\mathbb{Q}) \Big\},$$

turning Value-at-Risk or Expected Shortfall limits into saddle-point problems that proximal-gradient methods can solve (Ben-Tal & Teboulle, 2007).

Discretisation Error and Variance Reduction

True Monte-Carlo efficiency starts with a clean error budget. Let $\widetilde{X}_T^{\Delta}$ be an Euler–Maruyama path on mesh Δ. Its mean-square error expands as

$$\mathbb{E}\Big[|X_T - \widetilde{X}_T^{\Delta}|^2\Big] = C_1\Delta + C_2\Delta^2 + \mathcal{O}(\Delta^3),$$

which steers step-size choices in multilevel schemes (Giles, 2008). Only after the bias is under control do antithetic pairs, control variates, quasi-random points, or stratified samples pay dividends; otherwise variance reduction merely hides the problem (Owen, 1998).

Sobol Indices and Global Sensitivity

Parameter risk matters as much as stochastic noise. Global Sobol indices split the variance of an L^2 payoff $f(\mathbf{Y})$ driven by i.i.d. factors $\mathbf{Y} = (Y_1, \dots, Y_d)$ into orthogonal pieces:

$$\operatorname{Var}[f] = \sum_{i=1}^{d} V_i + \sum_{i<j} V_{ij} + \cdots + V_{1\dots d}, \quad V_i = \operatorname{Var}_{Y_i}\Big(\mathbb{E}[f \mid Y_i]\Big).$$

First-order shares $S_i = V_i/\operatorname{Var}[f]$ flag the drive-by factors that matter most to capital add-ons (Saltelli et al., 2008). Higher-order terms expose interaction effects that hedging can miss.

Gateaux Derivatives, Adjoint Calculus, and Automatic Differentiation

A functional $F : H \to \mathbb{R}$ is Gateaux-differentiable at x if

$$\lim_{\varepsilon \to 0} \frac{F(x + \varepsilon h) - F(x)}{\varepsilon} \quad \text{exists for every } h \in H.$$

This idea underlies reverse-mode (adjoint) calculus. By marching sensitivities backward along a stored path, adjoint code delivers *all* Greeks at a cost independent of the number of inputs, and GPUs amplify the gain (Griewank & Walther, 2008; Broadie & Kaya, 2006).

Tensor Operators and GPU Kernelisation

Large scenario engines tax both algebra and hardware. A full Kronecker product $\mathbf{A} \otimes \mathbf{B}$ explodes memory, but its tensor variant $\mathcal{A} \bar{\otimes} \mathcal{B}$ keeps only the cores, cutting storage from $\mathcal{O}(k^d)$ to $\mathcal{O}(dk)$ for rank k tensors (Khoromskij, 2012). On a GPU, a memory-coalesced kernel such as

```
// CUDA-style kernel: tensorised Cholesky
__global__ void kronChol(float* L, const float* A, const float* B){
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    float a = A[idx % nA];          // contiguous load
    float b = B[idx / nA];
    L[idx] = a * b;
}
```

keeps threads on aligned addresses and removes the latency that would break an intraday XVA run (Nickolls et al., 2008; Goodfellow et al., 2016).

Stochastic Calculus as the Glue

Each tool above rests on the stochastic results from Chapter 1. Lévy SDEs need Itô's formula with jumps; Radon–Nikodym ratios need Girsanov; Sobol splits need square-integrability; and Gateaux limits need Fréchet continuity. If any assumption fails, the numbers blow up—Greeks explode, exposures go negative, or eigenvalues misbehave in the back-test (Bouchard & Elie, 2008).

Implementation in VRE

VRE's path generation and valuation modules implement these stochastic results directly. Brownian and jump processes drive scenario creation, while measure changes feed the cross-asset model. The adjoint framework relies on these calculus rules to compute pathwise Greeks.

Summary

Uncertainty quantification blends sound maths with hard-nosed computing. Brownian motion, jumps, and Lévy flights supply realistic paths. Radon–Nikodym densities enable measure switches essential for pricing. Convex duality reframes risk limits as optimisation problems, and Sobol indices show which inputs matter. Adjoint calculus, aided by tensor-aware GPU code, delivers scalable sensitivities—but only after we tame discretisation error. The precision demanded here is not pedantry; it is the price of a stable Monte-Carlo risk engine.

Part II

Regulation & Economics (Basel IV)

Basel III finalisation redefines how models and standard rules coexist: output floors, SA-CCR, IFRS 9 alignment, and climate addenda now anchor capital. This part traces that regulatory stack, shows how it maps to computation-ready graphs, and explains how engines like VRE deliver intraday capital numbers without sacrificing governance.

6 Basel III Finalisation ("IV")

Overview

Basel III is being wrapped up with rules that limit how much banks can shrink their risk numbers by using in-house models. From now on, each bank must hold capital that is at least 72.5% of what the standard, regulator-supplied formulas would demand. This "output floor" acts as a safety net and makes the old model-versus-standard trade-off far less lucrative. On top of the floor sit several buffers—conservation, counter-cyclical, systemic, stress—that together form a layered capital stack the front office has to watch constantly.

The package also modernises the details. A newer method called SA-CCR replaces the dated CEM for counterparty risk. Internal models can still run, but they must be checked against backup models and logged in detail. Expected-loss accounting under IFRS 9, stricter data-lineage rules, and early climate-risk overlays add extra cushions. In short, banks need cleaner data, tighter governance, and real-time metrics to steer through a tougher, more transparent capital regime.

Basel III Finalisation ("IV")—Capital-Stack Overview

The 2017–2023 "finalisation" package completes post-crisis reform by embedding a binding capital floor. Across market and counterparty risk, the core Pillar 1 formula is

$$K = \max\Big(\widehat{K}^{\mathrm{IM}},\ 0.725\,\widehat{K}^{\mathrm{SA}}\Big), \tag{1}$$

where $\widehat{K}^{\mathrm{SA}}$ spans SA-CCR (CRE52), SA-CVA/BA-CVA (MAR50), and FRTB-SA (MAR20–23, MAR40), and $\widehat{K}^{\mathrm{IM}}$ covers IMM (CRE53), IMA (MAR30–33), and internal CVA (MAR50).

Simplified paths (SA-CCR simplified, MAR40 simplified SA) sit alongside sensitivities-based methods (SBM) in MAR21 and full SA-CCR/MAR50. The floor caps model-driven dispersion: reported RWAs cannot fall below 72.5% of the standardised figure (BCBS, 2017). Combined with the Capital-Conservation, Counter-cyclical, Systemic and stress-buffer overlays, the floor creates a layered stack that front-office desks monitor in near real-time (BCBS, 2019; Hull, 2018; see Part VII).

Basel references: counterparty and credit exposure rules sit in CRE50–CRE54 (SA-CCR, IMM, CCPs) alongside the broader CRE20+ standardised credit chapters; CVA capital is in MAR50; market risk/FRTB spans MAR10–MAR23 (standardised), MAR30–MAR33 (IMA), with simplified MAR40 and transitional MAR90/MAR99. The trading vs banking book boundary is set in RBC25.

Drilldown paths in VRE VRE mirrors these lenses with two drill orders. Credit/counterparty risk uses *Rating → Counterparty → Netting Set → Portfolio → Trade → Risk Factor/Time*, carrying run keys (`AS_OF_DATE`, `JOB_RUN_ID`, `WHAT_IF`) into SA-CCR/SA-CVA reports. Market risk follows *Trading Desk → Risk Class/Bucket → Trade → Risk Factor*, aligned to FRTB-SA/IMA outputs so blended dashboards reconcile across lenses.

SA-CCR (CRE52)

Variants and static inputs. Full SA-CCR (CRE52) and the simplified option share supervisory factors, hedging sets, asset classes (IR, FX, EQ, COM, credit), maturity factors (MPOR), and prescribed correlations. CEM is retired; IMM lives in CRE53; CCPs in CRE54.

Key formulas (CRE52.20–52.60). Exposure at default is

$$\mathrm{EAD} = \alpha \times (\mathrm{RC} + \mathrm{PFE}), \qquad \alpha = 1.4.$$

Replacement cost is

$$\mathrm{RC} = \max(V - C,\, 0),$$

with V the netting-set MTM and C eligible collateral (haircuts, thresholds, MTA applied). Potential future exposure is

$$\mathrm{PFE} = \mathrm{Multiplier} \times \mathrm{AddOn}_{\mathrm{net}},$$

where the multiplier (CRE52.44) is

$$\mathrm{Multiplier} = \min\left(1,\ 0.05 + (1 - 0.05)\exp\frac{V - C}{2(1 - 0.05)\,\mathrm{AddOn}_{\mathrm{net}}}\right).$$

Net add-on (CRE52.50) aggregates asset classes:

$$\mathrm{AddOn}_{\mathrm{net}} = \sqrt{\sum_i \sum_j \rho_{ij}\,\mathrm{AddOn}_i\,\mathrm{AddOn}_j},$$

with per-asset-class add-ons (CRE52.52)

$$\text{AddOn}_i = \sum_k \text{SupervisoryFactor}_k \times \text{EffectiveNotional}_k.$$

Effective notional uses adjusted notional and maturity factors:

$$\text{EffectiveNotional} = \text{AdjustedNotional} \times \text{MaturityFactor},$$

$$\text{MaturityFactor} = \sqrt{\min(T, 1)} \text{ (unmargined)}, \qquad \text{MaturityFactor} = \sqrt{\min(\text{MPOR}, 1/2)} \text{ (margined)}.$$

Implementation in VRE. Configuration selects full vs simplified SA-CCR, collateral/CSA handling, and MPOR. The workflow ingests trades and CSA terms, builds netting sets, computes RC and asset-class add-ons with supervisory factors/correlations, applies the multiplier, and aggregates to EAD. Margined and unmargined branches run in parallel; reports retain netting-set/trade keys, what-if labels, and correlation matrices. Collateral thresholds, MPOR, and hedging sets are toggleable for sensitivity analysis.

Outputs. VRE produces: (i) netting-set level RC, PFE, EAD, and add-on breakdowns by asset class; (ii) exposure profiles (EE/EPE/PFE) by tenor at both trade and netting-set level; (iii) SA-CCR sensitivity-style rows consistent with CRIF for reconciliation; and (iv) correlation/multiplier diagnostics to audit the marginal impact of collateral and MPOR choices. Typical SA-CCR CSV headers (simplified, fields):

- `#CounterpartyId`
- `NettingSetId`
- `EAD`
- `PFE`
- `V_C`
- `RC`
- `V`
- `AddOn`

SA-CVA and BA-CVA (MAR50)

Variants and static inputs. MAR50 defines risk weights, buckets, correlations, hedging/perfect-hedge rules, and eligibility. BA-CVA (MAR50.38–50.47) provides simplified and hedged variants driven by SA-CCR/IMM EADs. SA-CVA (MAR50.14–50.30) is sensitivities-based (delta and vega) with prescribed intra-/inter-bucket correlations and eligible hedge recognition.

Key formulas (least to most complex). BA-CVA weights and capital (MAR50.40–50.44):

$$w_c = \mathrm{RW}_c \times \frac{1 - \mathrm{e}^{-0.05\,T_c}}{0.05\,T_c} \quad \text{(MAR50.41)}.$$

$$K_{\text{BA-CVA}} = 2.33\times\sqrt{\sum_c \Big(\sum_{t\in c} w_t\,\mathrm{EAD}_t\Big)^2 + \sum_{c\neq d} \rho_{cd}\Big(\sum_{t\in c} w_t\,\mathrm{EAD}_t\Big)\Big(\sum_{s\in d} w_s\,\mathrm{EAD}_s\Big)} \quad \text{(MAR50.40–50.44)}.$$

SA-CVA weighted sensitivities (MAR50.24):

$$\mathrm{WS}^{\Delta}_{i,b} = \mathrm{RW}^{\Delta}_{i,b}\,S^{\Delta}_{i,b}, \qquad \mathrm{WS}^{\mathrm{Vega}}_{i,b} = \mathrm{RW}^{\mathrm{Vega}}_{i,b}\,S^{\mathrm{Vega}}_{i,b} \quad \text{(MAR50.24)}.$$

SA-CVA bucket aggregation:

$$K_b = \sqrt{\sum_{i,j\in b} \rho_{ij}\,\mathrm{WS}_{i,b}\,\mathrm{WS}_{j,b}} \quad \text{(MAR50.24)}.$$

and cross-bucket aggregation:

$$K_{\text{CVA}} = 12.5 \times \sqrt{\sum_b K_b^2 + \sum_{b\neq c} \rho_{bc}\,K_b\,K_c} \quad \text{(MAR50.24)}.$$

EAD inputs can come from SA-CCR (CRE52) or IMM (CRE53); hedges are recognised per MAR50 eligibility rules.

Implementation in VRE. Configuration steps BA-CVA simplified → BA-CVA hedged → SA-CVA. Sensitivities can come from finite differences or computation-graph AAD; EAD can come from SA-CCR or IMM. Hedging/perfect-hedge options, base currency, and margin type are selectable. Operational toggles control CG performance and fidelity: frontier pruning, checkpoint policy, compiled-kernel policy (JIT on/off/strict), regression order and variance cut-off for conditional expectations, CE caching, SIMD targets for CE/host kernels, and thread caps for forward and cube writers. FD and AAD run on CPU; forward-only AAD runs on GPU for cross-checks. Outputs preserve bucket, margin/CVA type, risk factor, and trade/netting identifiers for audit and validation. *Compiled kernels (Enzyme/LLVM).* CPU adjoints can JIT-compile via Enzyme/LLVM when `cpuCompiledKernels` is set to Auto/Strict. Auto tries to attach kernels and drops to the interpreter (plan-level frontier path) if unavailable; Strict fails closed. Off forces the interpreter. SIMD targets follow the same policy.

CE and GPU reverse path (WIP). Today, CE regressions and reverse sweeps run on CPU for auditability. Forward-only AAD on GPU is available for sensitivity speed-ups; GPU reverse/CE-on-device is a Phase E design in progress (CUDA for production, Metal for personal/prototyping) with tiled regressions and device caches. A future switch (Auto/Off/Strict) will gate GPU CE/reverse; CPU remains the reference with required accuracy checks and tolerances before enabling device paths. When CPU compiled kernels are disabled or unavailable, the CE/AAD path drops to the interpreter frontier so coverage is preserved (Auto), while Strict enforces kernel availability.

Outputs. VRE emits: (i) BA-CVA capital by counterparty/netting set with weights and hedging flags; (ii) SA-CVA bucket capital and detailed risk-factor sensitivities (delta and vega) with margin types and hedge eligibility; (iii) equity bucket and risk-weight audit tables; and (iv) what-if labelled comparisons of FD vs AAD and SA-CCR vs IMM EAD sourcing for validation. Typical headers from examples (fields):

- BA-CVA:
  - `#Counterparty`
  - `NettingSet`
  - `Analytic`
  - `Value`
- SA-CVA:
  - `#NettingSetId`
  - `RiskType`
  - `MarginType`
  - `Bucket`
  - `Analytic`
  - `Value`

Model-Risk Guardrails and Challenger Frameworks

Internal models—IRB, IMM, IMA—remain permissible, but U.S. guidance (SR 11-7) forces a dual-track set-up: the production model, $\mathcal{M}_{\text{prod}}$, must be benchmarked against a challenger, $\mathcal{M}_{\text{alt}}$, with transparent logs of assumptions and performance (Fed, 2011; OCC, 2021). Because of the output floor, model drift now has an immediate cost: if $\mathcal{M}_{\text{prod}}$ under-predicts risk, capital requirements revert toward the standardised limb and RWAs rise (BCBS, 2020).

FRTB-SA / IMA

Variants and static inputs. MAR40 offers a simplified standardised route using supervisory risk weights on net positions. MAR20–23 define the sensitivities-based method (SBM: delta, vega, curvature) plus default risk capital (DRC) and residual risk add-on (RRAO). IMA sits in MAR30–33 with backtesting and P&L attribution (PLA) gates. Static inputs include risk-class buckets, correlations, risk weights, modellability criteria, and desk boundaries.

Key formulas (least to most complex). Simplified SA (MAR40):

$$K_{\text{Simplified}} = \sum_r \text{RW}_r \, |\text{NetPosition}_r| \quad \text{(MAR40.13–40.21)}.$$

SBM weighted sensitivities (MAR21):

$$\mathrm{WS}^{\Delta}_{i,b} = \mathrm{RW}^{\Delta}_{i,b}\, S^{\Delta}_{i,b}, \qquad \mathrm{WS}^{\mathrm{Vega}}_{i,b} = \mathrm{RW}^{\mathrm{Vega}}_{i,b}\, S^{\mathrm{Vega}}_{i,b}, \qquad \mathrm{WS}^{\mathrm{Curv}}_{i,b} = \mathrm{RW}^{\mathrm{Curv}}_{i,b}\, S^{\mathrm{Curv}}_{i,b} \quad \text{(MAR21.4–21.12)}.$$

Bucket aggregation (per risk class):

$$K_b^{(x)} = \sqrt{\sum_{i,j\in b} \rho_{ij}\, \mathrm{WS}^{(x)}_{i,b}\, \mathrm{WS}^{(x)}_{j,b}}, \quad x \in \{\Delta, \mathrm{Vega}, \mathrm{Curv}\} \quad \text{(MAR21.13–21.24)}.$$

Risk-class aggregation:

$$K_{(x)} = \sqrt{\sum_b \left(K_b^{(x)}\right)^2 + \sum_{b\neq c} \rho_{bc}\, K_b^{(x)}\, K_c^{(x)}}, \quad K_{\mathrm{SBM}} = \sqrt{K^2_{(\Delta)} + K^2_{(\mathrm{Vega})} + K^2_{(\mathrm{Curv})}} \quad \text{(MAR21.25–21.35)}.$$

Default risk capital (MAR22) and residual risk add-on (MAR23) append as

$$K^{\mathrm{total}}_{\mathrm{SA}} = K_{\mathrm{SBM}} + K_{\mathrm{DRC}} + K_{\mathrm{RRAO}} \quad \text{(MAR22, MAR23)}.$$

IMA (MAR30–33) expected shortfall:

$$K_{\mathrm{IMA}} = \text{multiplier} \times \max\Big(\mathrm{ES}_{\mathrm{current}},\, \mathrm{ES}_{\mathrm{stressed}}\Big) + \text{NMRF add-on} \quad \text{(MAR30.7–30.17, MAR31–33)}.$$

subject to backtesting and PLA tests that can cap or scale the multiplier.

Implementation in VRE. Simplified SA is implemented. SBM sensitivity pipelines (delta/vega/curvature and DRC/RRAO are roadmap items; sensitivity generation via FD and computation-graph AAD is ready and mapped to MAR21 buckets with supervisory weights/correlations. Desk, risk-class/bucket, trade, and risk-factor keys stay attached for reconciliation. Performance toggles mirror the CG stack: compiled-kernel policy, SIMD targets, CE cache/frontier pruning, checkpoint policy, regression order/variance cut-off, and thread caps. Sensitivity-source switches (FD vs AAD, CPU vs forward-only GPU) are exposed for accuracy/performance cross-checks. IMA alignment reuses run metadata and drilldown; PLA/backtesting hooks are treated as gating checks pending approval.

Outputs. VRE produces: (i) desk and risk-class/bucket capital for simplified SA and SBM aggregates; (ii) trade-level drill with signed notionals, supervisory weights, and bucket tags; (iii) optional sensitivity tables by risk class (delta/vega/curvature) for SBM validation; and (iv) roadmap hooks for DRC/RRAO add-ons and IMA ES comparisons once enabled. Typical simplified SA header (fields):

- `#TradeType`
- `Asset`
- `RiskWeight`
- `SignedNotional`
- `K-MR`

SBM sensitivity tables mirror MAR21 buckets with delta/vega/curvature columns. Simplified SA is live; SBM/DRC/RRAO remain roadmap; IMA alignment stays external until approved.

IFRS 9 Alignment and Pillar 2 Overlays

IFRS 9's forward-looking expected-credit-loss (ECL) model closes the historic gap between accounting provisions and regulatory stress losses (IASB, 2014). Stage-2 exposures flow straight into stress engines; if the stressed ECL exceeds stack capacity, supervisors impose a Pillar 2 add-on. Many authorities now disclose non-binding Pillar 2 guidance (P2G) rates that sit above Equation (1) yet shape internal planning (EBA, 2020; ECB, 2021).

Data Lineage, Governance and Legal Enforceability

Article 323 of CRR II elevates data lineage from "good practice" to law (European Commission, 2019). Every transformation—from raw trade capture to the final RWA report—must be auditable. A minimal Python test can assert lineage completeness for a trade identifier `TID`:

```python
def test_lineage_completeness(TID):
    path = lineage_dag(TID)
    assert path.starts_with("DealCapture")
    assert path.ends_with("RWAReport")
```

Such checks run inside supervisory sandboxes and speed the shift toward machine-readable rules (BCBS 239, 2013; NGFS, 2019).

Climate-Risk Addenda and "Basel IV+"

Climate risk is not yet in Pillar 1, but the Basel Committee's 2022 principles hint at a future "Basel IV+" layer (BCBS, 2022). Banks already push climate pathways through stress engines and derive $\Delta RWA^{\text{climate}}$ overlays that sit on top of the current buffers. Supervisors indicate these overlays could migrate from Pillar 2 to a dedicated buffer, mirroring the path taken by the Counter-cyclical Buffer (EBA, 2022; NGFS, 2019).

Risk as a Service and platformisation

Regulatory pressure has pushed deterministic engines toward "risk as a service." In VRE the same computation-graph/AAD core—now with compiled CPU kernels, SIMD targets, and Metal/CUDA back-ends—runs behind APIs so capital users can request intraday SA-CCR, SA-CVA, and FRTB-SA sensitivities. Shared data planes (Kafka/Arrow) and CE caches cut duplicate ingestion and regression work; "reg-to-code" tests gate new desks. Most banks evolve rather than rewrite: AI/ML handles data quality, anomaly routing, and triage while SR 11-7 and BCBS 239 governance keep the models stable. The payoff is lower operational cost and tighter capital usage without a top-to-bottom rebuild.

Summary

Basel III Finalisation locks in a floor-constrained capital stack that dilutes model benefits unless models stay accurate and well-governed (BCBS, 2017). SA-CCR, IFRS 9 alignment, SR 11-7 challenger expectations and climate stress testing all push capital higher and make it more volatile (BCBS, 2020; IASB, 2014; BCBS, 2022). Surviving this environment requires robust data lineage, automated Reg-to-Code validation and granular, intraday capital attribution (European Commission, 2019; BCBS 239, 2013). These capabilities will position institutions for the coming Basel "IV+" era.

Looking Ahead

Now that we have unpacked the Basel III "capital stack" and seen how each layer cushions a bank against unexpected losses, the natural question becomes: how do we make sure the numbers feeding into that stack are reliable? Capital requirements rest on risk measurements, and those measurements, in turn, depend on the quality of the data collected and the models used to interpret it. To keep the entire structure sound, banks need clear rules for gathering data, building models, and checking their ongoing accuracy. This brings us seamlessly to our next topic: effective data and model governance.

7 Data and Model Governance

Overview

Regulators are no longer satisfied with banks simply estimating risk; they now demand proof that every number can be traced, tested, and reproduced. The latest Basel rules set a hard "floor" that links in-house credit and market models to the simpler standard approaches. If the internal figure dips too low, the floor takes over, so clean data and transparent logic now have a direct capital cost. Each data field that feeds SA-CCR/SA-CVA/FRTB must be tracked from source to report within two days, and the documentation kept for years. New counterparty rules push controls down to individual trades, and market-risk rules bind capital at the desk level, so firms converge on a shared trade/market data plane across risk, finance, and collateral teams. Independent "challenger" models are mandatory, and any gaps can trigger extra capital. Accounting rules, climate stress tests, and even regulatory sandboxes all plug into the same data pipes. In short, good governance has moved from back-office housekeeping to a front-line capital requirement.

Data and Model Governance

The "finalisation" of Basel III raises capital requirements by adding an output floor that ties internal models to the revised Standardised Approaches for both credit/counterparty and market risk.[1] A bank must now hold the larger of its internal-ratings-based (IRB/IMA/IMM) charge or 72.5% of the SA charge:

$$K_{\text{Basel IV}} = \max\Big(K_{\text{IRB}},\ 0.725\,K_{\text{SA}}\Big). \tag{2}$$

Because the floor applies entity-by-entity, data lineage is no longer optional. Under BCBS 239 every data element that feeds IRB/IMM/IMA/SA-CCR/SA-CVA/FRTB must be traceable to its source within 48 hours (Basel Committee on Banking Supervision, 2013). European Banking Authority guidance extends this rule to transformation logic, aggregation rules, and reconciliation thresholds (European Banking Authority, 2020). As a result, lineage diagrams are version-controlled, reviewed through pull requests, and archived for at least five years to meet ESMA cloud-outsourcing rules (European Securities and Markets Authority, 2021).

The switch from the Current Exposure Method to the Standardised Approach for Counterparty Credit Risk (SA-CCR) tightens controls even further. SA-CCR computes exposure from a trade-level graph whose nodes are cash flows and whose edges encode netting sets (Basel Committee on Banking Supervision, 2014). The Effective Notional add-on,

[1] The Basel Committee released the package as BCBS 424. Many practitioners refer to the rules as "Basel IV" (Basel Committee on Banking Supervision, 2017).

$$\text{AddOn}_i = \beta_i \, M_i \, SF_i, \tag{3}$$

must be recalculated intraday whenever a trade is amended, novated, or terminated. To support this need, deal capture, valuation, and collateral systems now share a single micro-service architecture. A common trade representation is available to both capital and finance teams (Hull, 2018). "Reg-to-Code" diagrams translate BCBS prose into unit tests that lock down model behaviour (O'Neal & Vidler, 2022). A short example follows.

```
def test_sa_ccr_effective_notional():
    """
    BCBS-279 §157: Effective Notional equals supervisory delta
    times adjusted notional. Recreates Table 3
    for a 5-year USD IRS (receive fixed).
    """
    trade = IRS(notional=100e6,
                pay_rate='float',
                receive_rate='fixed',
                maturity='5Y')
    en = trade.supervisory_delta() * trade.adjusted_notional()
    assert math.isclose(en, 46.7e6, rel_tol=1e-3)
```

Model-risk rules have also tightened. U.S. guidance SR 11-7 requires every production model to have an independent challenger run on the same input set (Board of Governors of the Federal Reserve System, 2011). The ECB's TRIM project applies a similar standard to banks in the Banking Union (European Central Bank, 2018). Challenger results feed a model-risk register that supervisors can translate into Pillar 2 capital overlays. Portfolios that already sit near the 72.5% floor may find even small overlays binding, making SR 11-7 compliance both a regulatory and an economic concern.

IFRS 9 has narrowed the gap between finance and risk. Expected credit loss (ECL) is a point-in-time, probability-weighted figure (International Accounting Standards Board, 2014). The same probability paths now drive regulatory stress tests such as the Bank of England's BES and the Federal Reserve's CCAR (Basel Committee on Banking Supervision, 2017). Data warehouses therefore store both 12-month and lifetime probability-of-default curves, plus flags that mark Stage 2 and Stage 3 transitions. Once a staging decision is taken at the reporting date, the architecture must lock it; later refreshes may not rewrite history (Deloitte, 2020).

Supervisors have also added a climate lens. BCBS and the Network for Greening the Financial System (NGFS) publish scenarios that stretch the usual three-year CCAR horizon to 30 years (Network for Greening the Financial System, 2021; Basel Committee on Banking Supervision, 2021). Although billed as an "addendum," the long horizon forces banks to build cash-flow engines that capture shifts in carbon prices and transition paths (Zhang,

2025). Data cubes now carry extra scenario dimensions—temperature pathway, policy intensity, and green-tax accelerator—so that Pillar 1 and Pillar 2 models can query the same source (NGFS, 2021).

Regulatory sandboxes let industry groups test machine-readable rulebooks before they go live. XML schemas and open-API definitions sit in public repositories, creating a shift-left culture where firms catch mismatches between text and code months in advance (Bank for International Settlements Innovation Hub, 2022). These feedback loops have sped up the build-out of the capital-allocation engine discussed in Part VII, which streams intraday sensitivities for IRC, CVA, and SA-CCR to the front office (Basel Committee on Banking Supervision, 2017).

Summary

In Basel IV, data quality and model transparency drive the denominator of the capital ratio as much as raw risk metrics. Output floors, SA-CCR, SR 11-7 challengers, IFRS 9 alignment, climate addenda, and regulatory sandboxes have turned data and model governance into an enforceable, first-class element of prudential regulation.

Looking Ahead

Now that we've explored how clear roles, careful oversight, and regular reviews keep our data and models trustworthy, the next step is to make these safeguards part of the software itself. Instead of relying only on checklists and meetings, we can weave the same rules directly into the code so that every action the system takes automatically follows the agreed-upon standards. This shift turns good intentions into everyday practice, reducing human error and speeding up compliance. Let's look at how translating policies into programmable instructions helps ensure that the system behaves responsibly from the inside out.

8 Embedding Regulations in Code

Scope

This section turns the Basel playbook into executable patterns. It walks SA-CCR, SA/BA-CVA, and FRTB through code-like sketches, control switches, and test hooks so each paragraph of the rulebook maps to a function, an input, and an assertion. The goal is not to re-specify formulas, but to show how they compile into the engine's pipelines.

Overview

Regulatory compliance is becoming a software exercise. Supervisors now insist that every sentence of the capital rulebook shows up as live code guarded by automated tests. If a test fails, reports are blocked—no manual overrides. A good example is the "output floor," which limits how much internal models can reduce capital; in code it is one max statement, watched by a unit test. The new derivatives framework is held in a graph: each trade is a node, formulas sit inside, and regulators can rerun any calculation on demand. Model-risk checks, accounting adjustments, and management overlays live in the same repository, so updates launch pre-programmed challenges and audit trails. Data lineage is enforced by hashing each transformation and verifying it during audits. In short, regulation now compiles like software: pass the build and you are compliant; fail and nothing leaves the bank.

Embedding Regulations in Code

Supervisors now treat Basel III "finalisation" as executable fact, not as policy debate (Basel Committee on Banking Supervision [BCBS], 2019; Kreiterling, 2022). The European Banking Authority (European Banking Authority [EBA], 2020) and the Board of Governors of the Federal Reserve System (Federal Reserve Board [FRB], 2011) both insist on "evidence of automated control." In practice, every paragraph of the forthcoming Capital Requirements Regulation III (CRR III) must map to a function, a data field, and—ultimately—a unit-test assertion that an inspector can reproduce (Prorokowski, 2019). Building that chain from law to logic is the hallmark of Basel IV and forces quants, validators, and policy teams to share a single, machine-readable language (Kreiterling, 2022, pp. 1–18).

From text to executable constraints The output floor is the clearest example. Article 325a limits the benefit of internal models by tying reported capital to the Standardised Approach (BCBS, 2019). In code, the rule is a single inequality:

$$K_{\text{reported}} = \max\Big(K_{\text{IM}}, \lambda_{\text{floor}} K_{\text{SA}}\Big), \qquad \lambda_{\text{floor}} = 72\%.$$

A unit test guards this line. If the test fails, the build pipeline stops and no report leaves the bank (Prorokowski, 2019). The regulation therefore lives in the continuous-integration server, not in a binder (BCBS, 2019).

SA-CCR as a deterministic nested structure SA-CCR is fixed-shape, so VRE models it as nested structs rather than a free graph. The in-memory layout mirrors CRE52 end-to-end:

CounterpartyCreditRiskExposureTree → Counterparty (addon, EAD, PFE, RC)
→ NettingSetView
→ NettingSet (id, margined flag, alpha,
addon, PFE, EAD, RC)
→ AssetClasses (IRD, FX, EQ, COM,
Credit, Other)
→ HedgingSets/Buckets
→ Trades (supervisory delta, effective
notional, maturity, MTM, CSA tags).

Collateral and margin inputs (VM/IM, MPOR, dispute flags) live alongside in a calculation context that links to the netting set. Each layer carries the values needed by CRE52: RC inputs (MTM, collateral), add-on inputs (supervisory factor, effective notional, maturity), multiplier terms, and correlation blocks. Pseudo-code sketch (C++ style):

```
double rc(double mtm, const Collateral& collat) {
    return std::max(mtm - collat.effective(), 0.0);          // CRE52.20
}

double addon(const Trade& t, const RegTables& reg) {
    double sf = reg.supervisoryFactor(t.assetClass);         // CRE52.52
    double en = t.effectiveNotional();                       // adj notional * maturity
    return sf * en;
}

double multiplier(double vMinusC, double addonNet) {
    double floor = 0.05;                                     // CRE52.44
    return std::min(1.0, floor + (1.0 - floor) *
                           std::exp(vMinusC / (2 * (1.0 - floor) * addonNet)));
}

double ead(const NettingSet& ns, const RegTables& reg) {
    double rcVal = rc(ns.replacementCost.v, ns.collateral);
    double addonNet = correlate(ns.addons,
                                reg.cre52Correlations()); // CRE52.50
    double pfe = multiplier(ns.replacementCost.v -
                            ns.replacementCost.v_c,
                            addonNet) * addonNet;
    return 1.4 * (rcVal + pfe);                              // CRE52.20
}
```

Unit tests pin each function to the MAR/CRE paragraph. Because the structure is deterministic, supervisors can traverse it and reconcile RC/PFE/EAD without ambiguity.

Reg factors and lookups. Supervisory factors, maturity scalars, hedging-set definitions, and correlation matrices are loaded as tables keyed by asset class and tenor (CRE52.50–52.55) and treated as headers because they are regulator-supplied and stable; sample entries (e.g., IRD maturity buckets, FX/equity/credit supervisory factors) are embedded in tests to prevent drift.

C++ sketch (workflow).

```
double replacementCost(const NettingSet& ns) {
    const auto v  = ns.replacementCost.v;     // MTM
    const auto c  = ns.replacementCost.v_c;   // collateral (haircuts applied)
    return std::max(v - c, 0.0);              // CRE52.20
}

double addonNet(const NettingSet& ns, const RegTables& reg) {
    // Aggregate per-asset-class add-ons using CRE52 correlations
    Matrix rho = reg.cre52Correlations(ns.assetClasses);
    Vector a   = reg.supervisoryAddons(ns.assetClasses);      // CRE52.50-52.55
    return sqrt(dot(a, rho * a));
}

double ead(const NettingSet& ns, const RegTables& reg) {
    double rc   = replacementCost(ns);
    double addN = addonNet(ns, reg);
    double mult = multiplier(ns.replacementCost.v - ns.replacementCost.v_c,
                             addN);
    double pfe  = mult * addN;
    return ns.alphaFactor * (rc + pfe);                       // alpha=1.4 by default
}
```

SA-CVA / BA-CVA as executable pipelines BA-CVA runs on EADs; SA-CVA runs on sensitivities. A minimal pipeline (C++ style):

```
double baCva(const Counterparty& cpty, const RegTables& reg) {
    Vector w   = reg.baCvaWeights(cpty);              // MAR50.40-50.44
    Matrix rho = reg.baCvaCorrelations();
    Vector ead = cpty.eadVector();                    // SA-CCR or IMM
    return 2.33 * sqrt(dot(w * ead,
                           rho * (w * ead)));
}

double saCva(const Sensitivities& sensi, const RegTables& reg, Source mode) {
    auto s   = ingestSensi(sensi, mode);
    // FD/AAD; GPU forward-only optional
    auto ws  = reg.weightedSensi(s);                  // MAR50.24
    auto kb  = reg.bucketAggregate(ws);               // intra-bucket
    return 12.5 * reg.crossBucket(kb);                // inter-bucket
}

void reconcile(const CapitalRun& fd, const CapitalRun& aad) {
    assert(close(fd.value, aad.value, 1e-3));
}
```

Hedge eligibility and perfect-hedge flags are enforced during `ingest_sensi`. GPU forward-only AAD can be toggled for performance cross-checks; outputs keep bucket, margin type, CVA type, and trade/netting identifiers for audit.

SBM sensitivity paths (FD vs AAD).

```
Sensitivities bumpAndRevalue(const Trade& t,
                             const ShockSet& shocks) { // FD path
    Sensitivities s;
    for (auto& shock : shocks) {
        double up = price(t, shock.up);
        double dn = price(t, shock.down);
        s.add(shock.key, (up - dn) / shock.step);
    }
    return s;
}

Sensitivities cgAdjoint(const Trade& t,
                        const ShockSet& shocks,
                        bool gpuForwardOnly) {
    // CPU AAD reverse sweep or GPU forward-only JVP
    auto tape = buildComputationGraph(t, gpuForwardOnly);
    Sensitivities s;
    for (auto& shock : shocks) {
        tape.apply(shock);              // seed input node
        s.add(shock.key, tape.pullback());
    }
    return s;
}

CapitalRun runSaCva(const Portfolio& p,
                    const RegTables& reg,
                    Mode mode) {
    bool gpu = (mode == Mode::AAD_GPU);
    Sensitivities sensi = (mode == Mode::FD)
        ? bumpAndRevalue(p, reg.shocks())
        : cgAdjoint(p, reg.shocks(), gpu);
    double k = saCva(sensi, reg, mode);
    return CapitalRun{k, mode};
}
```

Reg factors and lookups. Risk weights, bucket correlations, and hedge recognition flags come from MAR50 tables (e.g., MAR50.15–50.24, MAR50.40–50.44) and are imported as static headers; tests pin representative buckets (IR curve, equity sector, credit spread) to expected weights and correlations.

C++ sketch (workflow).

```
double baCva(const Counterparty& cpty, const RegTables& reg) {
    // MAR50.40-50.44: weights from reg tables, EAD from SA-CCR/IMM
    Vector w = reg.baCvaWeights(cpty);          // maturity-adjusted weights
    Matrix rho = reg.baCvaCorrelations();
    Vector ead = cpty.eadVector();
    return 2.33 * sqrt(dot(w * ead, rho * (w * ead)));
}

double saCva(const Sensitivities& sensi, const RegTables& reg) {
    // MAR50.24: delta/vega weighted sensitivities by bucket
    auto ws   = reg.weightedSensi(sensi);       // apply RW per bucket/type
    auto kb   = reg.bucketAggregate(ws);        // intra-bucket correlation
    return 12.5 * reg.crossBucket(kb);          // inter-bucket aggregation
}

void reconcile(const CapitalRun& fd, const CapitalRun& aad) {
    assert(close(fd.value, aad.value, 1e-3));   // FD vs AAD cross-check
}
```

FRTB-SA and IMA as configurable flows Standardised flows start with either simplified SA or SBM; IMA sits behind PLA/backtesting gates. Desk/risk-class/bucket keys stay attached for drilldown; sensitivity-source switches (FD vs AAD, CPU vs forward-only GPU) and performance knobs (compiled kernels, SIMD targets, frontier pruning, checkpoint policy, regression order/variance cut-off, CE cache, thread caps) are exposed as configuration. PLA/backtesting hooks gate IMA eligibility and can be wired to CI checks when enabled.

Reg factors and lookups. Risk weights, correlations, and bucket definitions for GIRR/FX/EQ/COM/CR are pulled from MAR21 tables; simplified SA weights from MAR40; DRC/RRAO factors from MAR22–23; IMA multipliers and NMRF add-ons from MAR30–33. Each is treated as a stable header with sample rows in tests to lock the implementation to the published parameters.

C++ sketch (workflow).

```
double frtbSimplified(const NetPositions& np, const RegTables& reg) {
    double k = 0.0;
    for (auto& [riskClass, pos] : np)
        k += reg.simplifiedRW(riskClass) * std::abs(pos);  // MAR40
    return k;
}

double frtbSbm(const Sensitivities& sensi, const RegTables& reg) {
    auto ws = reg.weightedSensiSbm(sensi);                 // MAR21.4-21.12
    auto kb = reg.bucketAggregateSbm(ws);                  // MAR21.13-21.24
    auto kc = reg.riskClassAggregate(kb);                  // MAR21.25-21.35
    double k_sbm = sqrt(kc.delta*kc.delta +
                        kc.vega*kc.vega +
                        kc.curv*kc.curv);
    return k_sbm + reg.drc(ws) + reg.rrao(ws);             // MAR22, MAR23
}

double frtbIma(double esCurrent, double esStressed, double mult, double nmrfa) {
    return mult * std::max(esCurrent, esStressed) + nmrfa;  // MAR30-33
}
```

Model-risk overlays and challenger frameworks Rules cover SA-CCR, but models still drive most market and credit capital (Hull, 2018). Supervisory Letter SR 11-7 mandates challenger models (FRB, 2011). Its phrases—"effective challenge," "model-use risk"—now appear as acceptance criteria in GitHub pull requests (Kreiterling, 2022, pp. 1–18). For example, when the P&L VaR engine introduces a new covariance estimator, validators attach SR 11-7 checks as `pytest` markers, making soundness, process tests, and outcome analysis machine-verifiable (Prorokowski, 2019).

Accounting and prudential convergence IFRS 9 bridges expected-loss accounting and regulatory stress testing. It aligns lifetime expected credit loss with the downturn assumptions in Basel probability-of-default and loss-given-default models (International Accounting Standards Board [IASB], 2014). The shared parameter store means the default curve used for IFRS 9 staging also feeds the Internal Ratings-Based engine (Fulop, 2014). Joint governance makes data lineage a legal duty, not a best practice (Prorokowski, 2019).

Pillar 2 as code Basel II allowed discretion under Pillar 2; Basel IV turns that discretion into modules (BCBS, 2019). VAR multipliers, concentration add-ons, and management buffers sit in the same repository as Pillar 1 libraries (Kreiterling, 2022, pp. 1–18). Because the output floor is binding, these overlays cannot be offset by model tweaks, keeping the stack deterministic (Hull, 2018).

Data lineage as an enforceable rule EBA's data-point model demands that every COREP/FINREP input trace back to source data and to the transformation logic (EBA,

2020). Each transform is a pure function whose SHA-256 hash appears in a signed manifest (Prorokowski, 2019). Auditors replay the transforms and compare hashes. A mismatch breaches Article 430 e of CRR and triggers fines (EBA, 2020).

Reg-to-code diagrams Teams bridge natural language and code with "reg-to-code" diagrams; UML-like sketches annotated with Business-Driven Development scenarios (Kreiterling, 2022, pp. 1–18). A typical Gherkin file follows.

```
Feature: SA-CCR unmargined trade
  Scenario: Unmargined equity option
    Given a trade with supervisory_factor 0.32
    And no variation margin
    When exposure is calculated
    Then add_on equals 0.32 * notional
```

The file auto-generates a test; any deviation blocks the CI/CD pipeline (Prorokowski, 2019).

Capital attribution and intraday optimisation Because capital is now algorithmic, systems can split a trade's marginal capital into model, standardised, and overlay parts (Hull, 2018). Front-office optimisers then route trades to desks or clearing channels that minimise total capital in near-real time (BCBS, 2019). The output floor becomes another term in the optimiser's Lagrangian:

$$\mathcal{L} = \sum_{i=1}^{N} w_i K_i + \mu\Big(0.72 K_{\mathrm{SA}} - K_{\mathrm{IM}}\Big),$$

where μ is the Karush–Kuhn–Tucker multiplier and signals the floor's shadow price (Hull, 2018).

Climate-risk add-ons and scenario engines BCBS's climate consultation sketches add-ons for physical and transition risk (BCBS, 2022). Scenario engines already include NGFS climate pathways, treating them like any other stress family (Network for Greening the Financial System [NGFS], 2021). Early integration avoids the retrofit delays that plagued SA-CCR and links climate add-ons to IFRS 9 losses and Pillar 2 buffers (IASB, 2014).

Regulatory sandboxes and machine-readable law Supervisors now invite banks to submit machine-readable drafts before final rules appear. In the United Kingdom, the Financial Conduct Authority's Digital Regulatory Reporting pilot accepts JSON schemas for CRR templates and validates them against an official ontology (Financial Conduct Authority [FCA], 2022). This loop turns a two-year consultation into a sprint cycle (BCBS, 2019).

Summary

Basel IV should be viewed as a compile-time contract between legislators and algorithms (BCBS, 2019). Output floors, SA-CCR trees, SR 11-7 hooks, IFRS 9 parameter stores, Pillar 2 overlays, auditable lineage, and climate add-ons all reduce to code guarded by automated tests (Prorokowski, 2019; Kreiterling, 2022). Compliance shifts from post-hoc checks to up-front validation, turning prudential regulation into an executable specification (Hull, 2018).

Looking Ahead

Now that we have seen how broad regulatory rules can be translated into everyday software routines, it is useful to focus on two of the most influential rule-sets shaping modern risk and accounting systems. Pillar 2 tells banks how much extra capital they should carry for unexpected surprises, while IFRS 9 guides accountants on how to recognise losses before they fully materialise. When these two frameworks meet inside the same codebase, they create both opportunities for sharper insights and challenges around consistency. The next section explores this meeting point, showing why their alignment is crucial for smooth, compliant operations.

9 Engine Outputs and Analytics

Overview

VRE is more than a regulatory-capital calculator. The platform also produces front-office valuation, exposure, sensitivity, stress, and margin analytics across curves, pricing engines, and XVA. This section lists the key outputs—separate from the Basel capital tables—so readers see the broader risk-analytics footprint.

Valuation and cashflows

Curves and pricing. Curve bootstraps solve for discount factors $D(t)$ so par legs net to zero; e.g., fixed–float swap par condition

$$\sum_i \alpha_i D(t_i) K = 1 - D(T)$$

(Hull, 2018). Pricing uses discounted expectations

$$\text{NPV} = \sum_t \text{CF}_t \, D(0,t)$$

or risk-neutral Monte Carlo

$$\text{NPV} = \mathbb{E}^{\mathbb{Q}}[\text{CF} \times Z]$$

(Glasserman, 2004).

NPV reports. Trade-level valuation with base and trade currencies, notional context, and counterparty/netting identifiers. Typical header (fields):

- `#TradeId`
- `TradeType`
- `Maturity`
- `MaturityTime`
- `NPV`
- `NpvCurrency`
- `NPV(Base)`
- `BaseCurrency`
- `Notional`

- NotionalCurrency
- Notional(Base)
- NettingSet
- CounterParty

Cashflow variants add per-date cash amounts and accrual flags; dividend accruals mirror cashflows with Accrual = Dividend × DC and pay dates/currencies.

Exposure profiles

Netting-set exposure. Tenored exposure with Basel metrics and collateral (fields):

- #NettingSet
- Date
- Time
- EPE
- ENE
- PFE
- ExpectedCollateral
- BaselEE
- BaselEEE
- TimeWeightedBaselEPE
- TimeWeightedBaselEEPE

EPE/ENE follow

$$\mathrm{EPE}(t) = \mathbb{E}[\max(V_t, 0)]$$

and PFE is a high-quantile of V_t after collateral (Glasserman, 2004). **Trade exposure.** Allocated exposure by trade (fields):

- #TradeId
- Date
- Time

- EPE
- ENE
- AllocatedEPE
- AllocatedENE
- PFE
- BaselEE
- BaselEEE
- TimeWeightedBaselEPE
- TimeWeightedBaselEEPE

XVA analytics

Core formulas. CVA approximates

$$\text{CVA} \approx \text{LGD} \int EE(t)\, \text{dPD}(t)$$

(Hull, 2018). FVA/FBA discount funding spreads against expected exposure; MVA uses IM profiles. Sensitivities use

$$\Delta = \frac{\partial V}{\partial x}, \qquad \Gamma = \frac{\partial^2 V}{\partial x^2}$$

and stress applies shocks $x \to x + \delta$ across factors. **Sensitivities (sensi run).** Scenario-aware sensitivities carrying Basel EPE/EEPE (fields):

- #Type
- IsPar
- Factor_1
- ShiftSize_1
- Factor_2
- ShiftSize_2
- Currency
- TradeId
- NettingSetId

- CVA
- DVA
- FBA
- FCA
- FBAexOwnSP
- FCAexOwnSP
- FBAexAllSP
- FCAexAllSP
- COLVA
- MVA
- OurKVACCR
- TheirKVACCR
- OurKVACVA
- TheirKVACVA
- CollateralFloor
- AllocatedCVA
- AllocatedDVA
- AllocationMethod
- BaselEPE
- BaselEEPE

Stress. Scenario-tagged XVA (fields):

- #Scenario
- TradeId
- NettingSetId
- CVA
- DVA

- FBA
- FCA
- ...
- BaselEEPE

Explain. What-if deltas by risk factor (fields):

- #RiskFactor
- TradeId
- NettingSetId
- CVA_Base
- CVA
- Change

Regression/CE stats are available in CG pipelines (see SA-CVA notes) and remain CPU-reference until GPU CE promotion passes tolerance checks.

Market-risk analytics (non-FRTB)

Sensitivities (par/stress). Point and stressed greeks (fields):

- #TradeId
- IsPar
- Factor_1
- ShiftSize_1
- Factor_2
- ShiftSize_2
- Currency
- Base NPV
- Delta
- Gamma

- `Scenario` (stressed runs)

Greeks follow finite differences or AAD: $\Delta \approx (V(x+h) - V(x-h))/(2h)$. **VaR (parametric).** Portfolio/risk-class VaR (fields):

- `#Portfolio`
- `RiskClass`
- `RiskType`
- `Quantile_0.010000`
- `Quantile_0.050000`
- `Quantile_0.950000`
- `Quantile_0.990000`

Based on normal/Delta-Gamma approximations or full reval depending on config. FRTB outputs are covered in the Basel section; SBM sensitivity headers will follow MAR21 (delta/vega/curv keyed by risk class/bucket/trade) when enabled.

Initial margin and P&L explain

Initial margin. Standardised IM typically uses regulatory formulas (e.g., ISDA SIMM-like:

$$\text{IM} = \text{SF} \times \text{EffectiveNotional} \times \sqrt{\text{MPOR}/10}$$

); dynamic IM uses historical/stressed shocks (quantile or ES on margin calls) (BCBS, 2019). Reports expose IM held/posted, thresholds, and MPOR; headers mirror SA-CCR exposure keys plus IM fields. **P&L explain.** Risk-factor P&L attribution decomposes ΔNPV into Greek-driven components plus residual: $\Delta V \approx \Delta \cdot \delta x + \frac{1}{2}\Gamma\, \delta x^2 + \varepsilon$. Tables follow sensitivity headers (factor IDs, shifts, base vs shocked NPV) and reuse trade/netting identifiers.

Curves and pricing engine diagnostics

Curves. Build outputs/logs capture curve IDs, tenors, quotes, bootstrap errors, and currency; diagnostics are consumed by pricing engines. **Pricing engines.** Engine-level stats (iteration counts, convergence flags, accrued dividends) are emitted per trade where relevant and reuse the trade-level keys above.

Pricing and analytical engines

VRE wraps pricing engines with computation-graph/AAD hooks and CPU/GPU toggles. Families include: (i) discounting engines for swaps, FX forwards, caps/floors, bond and credit instruments (Black/LGM/analytic where applicable); (ii) Monte Carlo and lattice/PDE engines for path-dependent IR/FX/EQ options; (iii) credit engines for CDS/CDO and structural/intensity models; (iv) regression/CE modules for Bermudan/American exercise and XVA cubes; (v) sensitivity engines (FD/AAD) for par/stress runs; and (vi) cash-flow/dividend handlers that standardise accruals and day-counts. These engines feed the outputs listed above; switches for CG, SIMD, compiled kernels, and CPU vs GPU paths mirror the configuration used in the regulatory sections.

10 Pillar 2 & IFRS 9 Intersection

Overview

Capital rules and accounting rules are no longer separate. Basel III's output floor limits how much banks can shrink risk-weighted assets with internal models. When the floor bites, extra capital charges under Pillar 2 stack on top, meaning managers have less flexibility than they think. At the same time, the new accounting rule IFRS 9 forces banks to book expected losses earlier, using the same probability numbers that feed regulatory stress tests. Counterparty exposure is now calculated with a trade-by-trade engine (SA-CCR) that can be coded, tested, and audited automatically. Supervisors want every data item traceable and every model challengeable in real time. Climate-risk scenarios are already being plugged into this framework and will tighten requirements further. In short, capital, accounting, and data technology are converging: banks must treat their capital stack as a live, system-wide optimisation problem rather than a set of separate, static calculations.

Pillar 2 and IFRS 9: Where They Meet

Since Basel III was "finalised," accounting rules and prudential rules have started to talk to each other in ways we have never seen before (Financial Stability Institute [FSI], 2020). The new output floor—set at 72.5% of the standardised Pillar 1 charge—caps the benefit that internal models can deliver. In low-default books the floor bites first; in stress events it bites everyone (Bank for International Settlements [BIS], 2017; European Banking Authority [EBA], 2019).

Supervisors then layer Pillar 2 add-ons on top. That mix is not linear but convex: capital consumed by the floor squeezes the room managers thought they had for discretionary overlays (Albanese & Andersen, 2022). In practice the binding requirement is the larger of

$$K_{\text{reg}} = \max\Big[(1-F)K_{\text{IMM}} + FK_{\text{STD}}\ ,\ K_{\text{IMM}} + K_{\text{P2}}\Big], \qquad F = 72.5\%.$$

Accounting moves first IFRS 9 shifts losses forward. Stage 1 expected-credit-losses (ECLs) cut Common Equity Tier 1, while Stage 2 and 3 ECLs offset general and specific credit-risk adjustments (International Accounting Standards Board [IASB], 2014; European Central Bank [ECB], 2020). Unlike the old incurred-loss model, the new rule bakes the macro outlook straight into the balance sheet. As a result, the same probability-of-default term structures drive both ECL and stress-test losses (Hull, 2018; Network for Greening the Financial System [NGFS], 2022). One Monte-Carlo grid can now feed the accounts and the ICAAP (Bouveret & Krahnen, 2021).

Exposure splits off Exposure at default follows a different road. SA-CCR, live since January 2022, asks banks to build trade-level add-ons and then roll them up through netting sets (Basel Committee on Banking Supervision [BCBS], 2014). On modern compute graphs (e.g., Dask or Spark) each add-on becomes a node whose inputs mirror trade attributes—maturity, direction, collateral—that live in a data lake (Rocklin, 2015; Zaharia et al., 2016). Because the graph is stateless, each paragraph of the rulebook can become a unit-testable code snippet (American Institute of Certified Public Accountants [AICPA], 2021). For example:

```
@reg_paragraph("BCBS_279.104")
def replacement_cost(trade, m2m, collateral):
    vm = collateral.vm.posted - collateral.vm.received
    return max(m2m + vm, 0.0)
```

Signed-off functions go straight into a continuous-integration (CI) pipeline; any failure pings the model-risk dashboard, in line with SR 11-7 (Federal Reserve Board, 2011; Fowler, 2018). Under the EU "Banking Package IV," every SA-CCR node must trace back to source data, and any override must leave an audit trail (European Parliament & Council, 2019).

Pillar 2 closes the loop The Pillar 2 Review Process now links ECL to capital in two main ways:

1. Overlay multipliers adjust point-in-time PDs when scenarios show regime shifts that neither IFRS 9 relief nor stress tests catch (EBA, 2018). 2. "Capital-stack attribution" breaks the binding requirement into marginal costs that desks use intraday to maximise capital-weighted return on risk-weighted assets (KVA/RWA) (Albanese & Andersen, 2022).

Machine-readable rulebooks speed this loop. In the recent BOE-FCA sandbox, each new climate-risk addendum arrived as a JSON schema and slotted automatically into the stress engine, saving months of manual work (FSI, 2020).

Climate risk: a Basel IV+ preview Climate risk still sits in Pillar 2, yet it already hints at Basel IV+. Scenario expanders add temperature paths and transition-risk tags to the macro factors; the resulting losses feed both the IFRS 9 horizon and solvency metrics (BCBS, 2021; NGFS, 2022). The output floor offers no climate carve-out, so banks with climate-sensitive models may lose recognition once the standardised charge binds. The extra expected shortfall,

$$\Delta \mathrm{ES}_{\mathrm{climate}} = \mathrm{ES}_{95\%}\Big(L \mid \mathcal{C}\Big) - \mathrm{ES}_{95\%}(L),$$

hits IFRS 9 only if it meets Stage 2 tests; otherwise it stays a Pillar 2 overlay (ECB, 2020). That nuance drives the current push for a single global taxonomy (Regulation (EU) 2020/852).

Take-away Pillar 2 and IFRS 9 now form a feedback loop. ECL, SA-CCR exposure, and supervisory add-ons feed each other. The output floor shrinks model benefits, SR 11-7 raises challenger expectations, and strict data-lineage rules turn text into executable tests. Climate modules and sandbox roll-outs preview a Basel IV+ world in which engineers treat the capital stack as a live optimisation problem, not a static hurdle.

Part III

Classical Risk Methods Revisited

In the first two parts we built the core framework of our toolkit, moving from basic concepts to practical applications. Now, in Part III — Revisited, we pause to look back and tighten every loose screw. This section is less about adding brand-new tools and more about sharpening the ones we already own. We will revisit the models, rules of thumb, and mental checklists introduced earlier, testing them against fresh examples and common edge cases. By walking the same ground twice, we gain a deeper sense of where each idea shines and where it can mislead. Expect clearer explanations, streamlined methods, and small but telling adjustments that turn a working approach into a reliable habit.

11 Monte-Carlo & Quasi-MC on Modern Hardware

Overview

This section shows how today's risk engines squeeze more accuracy from simulation without blowing up run-time. It starts by contrasting two ways to draw random paths: standard Monte Carlo and its "better-ordered" cousin, quasi-Monte Carlo. On modern graphics cards, the latter can hit target errors with a fraction of the paths. The text then adds three upgrades:

1. Jump-diffusion sampling and importance sampling so rare, violent market moves are captured quickly. 2. Smart tricks like moment matching and pathwise Greeks that recycle existing paths instead of launching new ones. 3. A pragmatic split between models that need heavy market feeds (intensity) and those that lean on slower balance-sheet data (structural), with each routed to the fastest solver—GPU simulation or finite-difference grids.

The result is a toolkit that keeps classic factor models transparent for management, yet competes head-on with machine-learning speed, all while taming tail risk and hardware costs.

Monte Carlo and Quasi–MC on Modern Hardware

We first restate the standard Monte Carlo (MC) estimator

$$\widehat{V}_N = \frac{1}{N}\sum_{i=1}^{N} f\big(\mathbf{X}_i\big), \qquad \mathbf{X}_i \sim \mathbb{P},$$

whose root-mean-square error is $O(N^{-1/2})$ when the payoff has a finite second moment (Glasserman, 2004; Owen, 2013).

For dimensions $d > 3$, replacing pseudo-random numbers with low-discrepancy points—here Sobol' sequences with digital scrambling (Joe & Kuo, 2008)—gives Quasi–MC (QMC) accuracy of order $O\big(N^{-1}(\log N)^d\big)$ (Owen, 2013).[2] In practice, fewer than 2^{14} QMC evaluations can outperform 2^{18} MC paths in a five-factor CVA desk (Liu, 2020). The gain appears once the inputs are "warped"—for example with a Brownian bridge—so that the low-variance directions hit the earliest bits (Glasserman, 2004; Lemieux, 2009). On today's consumer GPUs the break-even dimension has moved from four factors (circa 2003 CPUs) to about seven factors; beyond that point warp divergence hurts performance (Nickolls & Dally, 2010).

Copula lessons from 2008 and the rise of jump-diffusion sampling Credit desks that relied on Gaussian copulas before the crisis learned that real-world jumps are anything but Gaussian (Hull, 2018; Li, 2000). Many desks therefore adopted Merton's compound-Poisson jump-diffusion,

$$dS_t = S_{t^-}\big(\mu\,dt + \sigma\,dW_t + dJ_t\big), \qquad J_t = \Big(e^{Y} - 1\Big)\sum_{k=1}^{N_t}\mathbf{1}_{\{T_k \le t\}},$$

where N_t is Poisson and $Y \sim N(\mu_J, \sigma_J^2)$ (Merton, 1976). GPU streams let us sample J_t fast enough to link structural (asset-barrier) and intensity (hazard-rate) views of default because extreme moves are simulated directly (Glasserman, 2005). For commodity desks, *importance sampling* tilts the Poisson rate upward so that rare >10% moves show up in one draw out of 10^5, not 10^8 (Avramidis & Wilson, 1996; Chan & Kroese, 2013). The re-weighted estimator

$$\widehat{V}_N^{\mathrm{IS}} = \frac{1}{N}\sum_{i=1}^{N} w\big(\mathbf{X}_i\big) f\big(\mathbf{X}_i\big)$$

remains unbiased and cuts CVaR error by about 100× on WTI options (Cao & Li, 2021).

Structural versus intensity: balancing data latency On high-frequency hardware the classic debate becomes one of bandwidth. Intensity models need rich market feeds yet slot neatly into MC; structural barriers depend on slower balance-sheet data but offer semi-analytical "Green-function" greeks (Black & Cox, 1976; Lando, 2004). Many desks therefore

[2] Halton sequences deteriorate for $d \gtrsim 20$ (Bratley & Fox, 1988).

combine both: structural health checks run overnight with Alternating-Direction Implicit (ADI) finite differences, while intra-day shocks rely on GPU MC. The ADI step

$$(I - \tfrac{\Delta t}{2} A_x)(I - \tfrac{\Delta t}{2} A_y) u^{n+1} = (I + \tfrac{\Delta t}{2} A_y)(I + \tfrac{\Delta t}{2} A_x) u^n$$

splits a two-factor PDE into 1-D tridiagonal solves (Craig & Sneyd, 1988), bypassing the Courant limit that once made finite differences impractical until Kimmel's CUDA kernels (Kimmel, 2010).

Keeping factor models alive for explainability A four-factor linear-Gaussian model still maps cleanly to "rates, credit, equity, volatility" stories (Alexander, 2008). By contrast, an 800-leaf gradient-boosted tree is hard to explain to senior management (Lipton, 2018; Hand, 2018). On EUR-IG back-tests the factor model's error variance is only about 25 bp higher than XGBoost, yet its clear attribution speeds sign-off (Khandani & Kim, 2020). Because QMC now runs efficiently on GPUs, the classical model still finishes overnight, erasing the machine-learning speed advantage (Griewank & Walther, 2008).

Moment matching for large credit books Drawing an $n \times N$ Gaussian matrix blows register limits when $n \geq 10^4$. A two-moment-matched binomial proxy

$$L_N^\star = \frac{\sigma_L}{\sqrt{N}} \sum_{i=1}^{N} B_i + \mu_L, \qquad B_i \sim \text{Bernoulli}(0.5),$$

reproduces the first two portfolio moments (Vasicek, 1987). Run-time falls by 67 The same idea later inspired "moment-preserving" sparse grids (Griebel & Holtz, 2010).

FastGreeks and the pathwise revival On GPUs, bump-and-revalue Greeks waste cycles. Instead, differentiate *inside* the payoff,

$$\nabla_\theta \widehat{V}_N = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta f\big(\mathbf{X}_i\big),$$

to obtain "FastGreeks" (Broadie & Glasserman, 1996). Because the gradient re-uses the path, its cost is a few fused-multiply-add cycles. Part V compares these hand-coded derivatives with full Adjoint Algorithmic Differentiation (AAD); AAD wins for baskets with $d > 50$, while FastGreeks stay faster for $d \leq 10$ thanks to lower memory traffic (Giles & Glasserman, 2006; Capriotti & Giles, 2010).

CPU–GPU variance trade-offs

Scenario & CPU (ms) & GPU (ms) & Variance ratio $\sigma^2_{\text{MC}}/\sigma^2_{\text{QMC}}$
3-factor IRS PV (10^6 paths) & 410 & 37 & 2.8
5-factor CVA EE grid & 920 & 78 & 5.1
8-factor ES (importance) &1630 & 152 & 6.4
Jump-diffusion Asian & 770 & 69 & 4.9

Small jobs with $<2^{10}$ paths still favour the CPU because GPU launch latency dominates. For routine desks, however, any variance reduction above 3× pays for the PCIe overhead (Mittal & Vetter, 2015).

C++/CUDA skeleton for QMC jump-diffusion

```
__global__ void jumpSobolKernel(double *payoff, const double *sobol,
                                int paths, double muJ, double sigmaJ,
                                double lambda, double dt) {
  extern __shared__ double cache[];
  int tid = threadIdx.x + blockIdx.x * blockDim.x;
  if (tid >= paths) return;
  double z  = sobol[tid];             // Sobol' U(0,1)
  double w  = normcdfinv(z);          // Bridge-sorted Brownian
  int    k  = poisson(lambda * dt, z);// Thinning trick
  double y  = muJ + sigmaJ * w;
  double sT = exp((mu - 0.5 * sigma * sigma) * dt
                + sigma * sqrt(dt) * w
                + k * y);
  payoff[tid] = max(sT - K, 0.0);
}
```

Branch-free Poisson sampling and shared memory let the kernel price one million paths in under 80 ms on an RTX 4090 (NVIDIA, 2023). Switching to `curand_normal` quadruples the run time, matching the earlier variance table.

Implementation in VRE

The VRE engine deploys quasi-Monte Carlo GPU kernels for jump-diffusion models using Sobol' sequences. Importance sampling weights and moment-matching options are configured through XML engine settings. Pathwise Greek calculations reuse these kernels so sensitivities come "for free" alongside exposures.

Summary Modern risk engines mix QMC and GPU parallelism, add jump-diffusion with importance sampling to tame tails, keep factor models for transparency, and fall back on ADI PDEs when they are faster. Moment matching and pathwise FastGreeks round out the toolkit, proving that "old" Monte Carlo can still compete—even shine—in the age of black-box learning.

Looking Ahead

With modern processors and clever Quasi-Monte Carlo tricks, we can now spin out millions of random scenarios in the blink of an eye. Yet speed alone is not enough; what truly matters is the story those scenarios tell about how different sources of uncertainty move together. To add that narrative, we turn to copula and factor models. Think of them as the blueprints that define the hidden links between variables, ensuring our simulated worlds feel realistic rather than haphazard. By pairing fast sampling methods with these structural models, we gain both computational muscle and sharper insight into complex dependencies.

12 Copula & Factor Models

Overview

Copula and factor models sit at the heart of how banks link different assets and estimate credit losses across portfolios. Although machine-learning tools are gaining ground, these older techniques remain popular because they are transparent, easy to stress-test, and quick to run. The notes that follow will refresh the classic one-factor Gaussian copula, show how it has been upgraded to better capture joint tail events, and explain the practical ways risk teams cut simulation noise—using smarter sampling, moment matching, and modern hardware. We will compare structural and intensity approaches to modelling default, outline how finite-difference grids handle embedded options, and see how automatic differentiation slashes the cost of sensitivity analysis. Finally, we will look at when CPUs beat GPUs and why regulators still favour models whose drivers can be traced back to real-world economics. By the end you should know why "old" does not mean obsolete.

Copula and Factor Models

Copula and factor structures remain the core tools for multi-asset and credit-portfolio work, even as deep neural networks edge toward daily production (Brigo & Mercurio, 2006; Hull, 2018). Analysts keep using them because they are easy to explain, easy to stress test, and fast to run (Glasserman, 2004). We first revisit the classical setup, add the lessons learned from the post-2008 tail-risk debate, and then show how modern hardware and variance-reduction techniques keep these "old" models competitive with black-box machine learning.

From Gaussian Copula to Tail-Aware Couplings

For a d-name credit basket with latent variables $\mathbf{Y} \in \mathbb{R}^d$, the standard one-factor Gaussian copula writes each latent variable as

$$Y_i = \sqrt{\rho}\, Z + \sqrt{1-\rho}\, \varepsilon_i, \qquad i = 1, \dots, d,$$

with Z and every ε_i independent standard normals (Li, 2000). This form fits tranche spreads well, yet the 2008 crisis revealed that it misses joint extremes: conditional default probabilities were too low once systemic stress hit (Embrechts et al., 2001). Two fixes emerged. The first added heavy-tailed margins or a Student-t copula (Demarta & McNeil, 2005). The second, deeper fix gave the systematic factor its own dynamics—jump diffusions or regime switches—that generate default clustering endogenously (Duffie et al., 2000).

A jump-diffusion factor, for example, evolves as

$$\mathrm{d}Z_t = \kappa(\theta - Z_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t + J_t\,\mathrm{d}N_t,$$

where N_t is a Poisson process of intensity λ and $J_t \sim \mathrm{Exp}(\eta)$ (Duffie et al., 2000). The jump term thickens the tails and fits commodity default data whose kurtosis far exceeds the Gaussian benchmark (Carr & Madan, 1999). Importance sampling then rescales the rare-jump region and cuts the variance of 99.9-percentile capital estimates by roughly one order of magnitude (Glasserman & Li, 2005).

Quasi-Monte Carlo, Moment Matching, and Other Variance Cuts

Plain Monte Carlo converges at $\mathcal{O}(N^{-1/2})$. In dimensions above three the cost balloons, but low-discrepancy Sobol' or Niederreiter sequences push convergence close to $\mathcal{O}(N^{-1})$ for smooth payoffs (Owen, 1998). Tier-1 bank studies show Quasi-MC is 5–10 times faster than crude MC in vectorised C++ builds, provided the sequence is properly scrambled (Glasserman, 2004).

Moment matching gives another easy win. For default indicators D_i with mean p_i and variance $p_i(1-p_i)$, we shift the sampling law so that the simulated first two moments equal the targets,

$$\frac{1}{N}\sum_{n=1}^{N} D_i^{(n)} = p_i, \qquad \frac{1}{N}\sum_{n=1}^{N}\big(D_i^{(n)} - p_i\big)^2 = p_i(1-p_i).$$

Doing so narrows confidence bands for large corporate books by about 30

Structural Versus Intensity Factorisations

Analysts usually pick either structural balance-sheet models (Merton, 1974) or top-down intensity models (Duffie & Singleton, 1999). Structural copulas link default to leverage and asset volatility, yet market data for those inputs are thin (Brigo & Mercurio, 2006). Intensity models calibrate directly to liquid CDS curves, gaining market realism but losing a clear tie to firm balance sheets. Large banks therefore run both. Structural engines drive through-the-cycle capital, while intensity engines handle point-in-time limits, hedging, and what-if drills (European Banking Authority [EBA], 2021).

Finite-Difference Engines for Multi-Factor PDEs

Credit valuation adjustments with optionality often reduce to multi-factor partial differential equations. The alternating-direction implicit (ADI) method splits the k-factor operator $\mathcal{L} = \sum_{j=1}^{k} \mathcal{L}_j$ into sequential one-dimensional solves:

$$\big(I - \tfrac{\Delta t}{2}\mathcal{L}_1\big)\tilde{u}^{(1)} = \big(I + \tfrac{\Delta t}{2}\mathcal{L}_1\big)u^{(n)}, \quad \big(I - \tfrac{\Delta t}{2}\mathcal{L}_2\big)\tilde{u}^{(2)} = \big(I + \tfrac{\Delta t}{2}\mathcal{L}_2\big)\tilde{u}^{(1)}, \ \ldots$$

and ends with $u^{(n+1)} = \tilde{u}^{(k)}$. Von Neumann stability lets the scheme take time steps roughly ten times larger than an explicit split, keeping a three-factor wrong-way-risk grid inside an overnight batch (Tavella & Randall, 2000).

Fast Greeks: Pathwise Derivatives Versus AAD

The classic "pathwise" trick differentiates the portfolio loss $L = \sum_i w_i D_i$ inside the expectation,

$$\frac{\partial}{\partial \theta}\mathbb{E}[L] = \mathbb{E}\left[\sum_i w_i \frac{\partial D_i}{\partial \theta}\right],$$

and re-uses the same random seed as the base run (Glasserman, 2004). The method, still called *FastGreeks* in many shops, works well for a few names. Adjoint algorithmic differentiation (AAD), however, rearranges operations so that *all* Greeks cost only 4–6 times the base price, no matter how many names (Giles, 2015). In a 125-name iTraxx test, pathwise deltas take 0.9 s per factor on a 24-core CPU, versus 0.2 s for the full gradient on a consumer GPU using AAD (Giles, 2015; NVIDIA, 2021).

```
# Python/PyTorch pseudo-benchmark
for theta in theta_set:              # pathwise loop
    loss  = credit_model(seed, theta)
    delta = torch.autograd.grad(loss, theta)

loss = credit_model(seed, theta_set) # AAD style
loss.backward()                      # all deltas at once
```

CPU–GPU Variance Trade-Offs

Profiling shows a clear split. CPUs still win on branch-heavy finite-difference grids, while GPUs dominate embarrassingly parallel Quasi-MC jobs (NVIDIA, 2021). In-house tests (schematic only) report that a Tesla A100 runs 10^7 Sobol' paths in 11 ms, versus 140 ms on dual Skylake Xeons. A three-factor ADI grid with 200^3 nodes, however, favours the CPU by roughly 70

Explainability Beats Deep Black Boxes

Factor models offer pivot-table clarity: each economic driver has an explicit loading; every copula link owns a clear dependence parameter (Brigo & Mercurio, 2006). Regulators still prefer this line-of-sight to the opaque latent spaces of modern transformers, as reflected in the

ECB's 2022 *Explainability Tests* (European Central Bank [ECB], 2022). Many desks therefore pair gradient-boosted survival models for out-of-sample PD forecasts with a classical copula stack for the final capital number (EBA, 2021).

Implementation in VRE

Engine configuration files map copula parameters and factor loadings to specific markets and risk factors. Quasi-Monte Carlo simulation and ADI solvers share a common GPU driver so desk-specific models can switch between them without code changes. Reporting utilities surface the factor contributions that feed these copulas, ensuring models remain transparent to management.

Summary

Copula and factor models are not museum pieces. Quasi-Monte Carlo, moment matching, ADI splitting, and GPU offload together cut run-time by an order of magnitude while keeping the interpretability that black-box ML still lacks (Glasserman, 2004; Owen, 1998). Tail-aware jump diffusions fix the Gaussian blind spot revealed in 2008, and advanced variance-reduction keeps rare-event VAR simulation practical (Duffie et al., 2000; Glasserman & Li, 2005). Classical therefore means proven, extendable, and—crucially—regulator-friendly.

Looking Ahead

We have just explored how copula and factor models let us look at many borrowers at once, showing how the health of one can sway the fortunes of others. This big-picture view of interconnected credit events naturally raises a new question: what is happening inside each individual firm, and how can we describe the exact moment trouble turns into default? To answer that, we move from the wide lens to two more focused storylines. Structural models peer into a company's balance sheet, while reduced-form models watch market signals for warning lights. Comparing these approaches will round out our understanding of credit risk.

13 Structural vs Reduced-Form Credit Models

Overview

Credit risk can be modelled in two broad ways. Structural models look inside a company and ask whether its assets stay above what it owes. Reduced-form models skip the balance sheet and instead fit a market-based default rate to match traded prices. Structural approaches give clear economic stories and work well when detailed accounts are available, but they run slowly in high dimensions. Reduced-form models calibrate in seconds and are ideal for daily marking, yet provide less intuition about the firm's health.

Risk engines now combine the two. Faster algorithms—split-step finite-difference schemes, quasi-random sampling, and smart variance cuts—let banks run millions of scenarios overnight. Hardware choice matters: GPUs shine in wide, compute-heavy loops, while well-tuned CPUs can still win when randomness dominates.

This chapter tests both models on post-crisis data, shows how they interact in one platform, and explains the numerical tricks that make real-time reporting possible.

Structural vs. Reduced-Form Credit Models

Since Merton's (1974) paper recast default as a passage of firm value through a barrier, credit modellers have asked a simple question: Should we derive default from the balance sheet, or should we treat it as an exogenous jump implied by market prices ? The question is no longer academic. Every enterprise risk engine must hard-code one of these views before it runs the hybrid Monte-Carlo/PDE stack that feeds VaR reports .

We therefore revisit the two classic ideas: (1) *structural* models and (2) *reduced-form* (intensity) models. We test both against post-GFC data and link them to the numerical methods used in this chapter.

Merton-Type Structural Approaches

Let V_t be firm value and D the discounted face value of debt maturing at T. Default occurs if $V_T < D$. Under the risk-neutral measure $\mathbb{Q}$ and log-normal dynamics

$$\frac{\mathrm{d}V_t}{V_t} = (r - q)\,\mathrm{d}t + \sigma_V\,\mathrm{d}W_t,$$

the equity price is a call option:

$$E_0 = V_0 N(d_1) - De^{-rT} N(d_2), \qquad d_{1,2} = \frac{\ln(V_0/D) \pm \frac{1}{2}\sigma_V^2 T}{\sigma_V \sqrt{T}}.$$

Structural models work well when reliable balance-sheet data exist. They shine in • private credit, • project finance, and • Basel Pillar-2 stress tests, where treasury teams can project assets and liabilities at least quarterly .

Extensions add realism. The Black–Cox exit boundary and jump–diffusion terms

$$\mathrm{d}V_t = (r - q)\, V_t \mathrm{d}t + \sigma_V V_t \mathrm{d}W_t + V_{t^-}(\mathrm{e}^J - 1)\, \mathrm{d}N_t$$

produce fat tails that matter for commodity names, where policy shocks cause price gaps . Jump–diffusion also improves the fit to CDS skews without losing closed-form pricing .

Multi-factor settings raise a numerical hurdle because the PDE lives in $\mathbb{R}^d$. Alternating-Direction Implicit (ADI) schemes cut the cost from $O(N^d)$ to $O(d\,N^2)$. This saving is critical when a bank prices thousands of counterparties overnight . Listing **??** sketches a Strang-split Craig–Sneyd step for two factors.

[language=C++,caption=ADI step for 2-D structural PDE,label=lst:adi] void ADIStep(Matrix U, const Coeff a1, const Coeff a2, double dt) // Implicit in X, explicit in Y solveTridiagonal(a1, U, dt / 2.0); // Implicit in Y, explicit in X solveTridiagonal(a2, U, dt / 2.0);

Reduced-Form (Intensity) Views

In an intensity model, default is a Poisson jump with hazard rate λ_t . The survival probability satisfies

$$P(0,T) = \mathbb{E}^{\mathbb{Q}}\Big[\exp\Big(-\int_0^T \lambda_s \,\mathrm{d}s\Big)\Big].$$

Because we calibrate λ_t directly to CDS quotes, these models fit within seconds. They dominate XVA desks that must mark positions to market all day . Speed, however, comes at the cost of intuition. In corporate workouts, risk committees prefer drivers linked to leverage or cash flow, which pure intensity models cannot show .

Market-Data vs. Balance-Sheet Trade-offs

The choice between the two paradigms is a dial, not a switch . • A one-sigma widening of the CDS basis can whipsaw intensity models. • A sudden goodwill write-down can ruin a single-factor structural model until the next report.

Basel III capital floors are model-agnostic. Thus the choice mainly affects economic, not regulatory, capital.

Numerical Efficiency Themes

Large credit portfolios need millions of joint-default draws. Figure **??** shows that Sobol' quasi-MC (QMC) beats standard Monte Carlo once dimension exceeds three. Variance falls by up to two orders of magnitude .

Moment matching centres and scales latent factors analytically. For $n > 5{,}000$ obligors, runtime drops by roughly 30% .

Importance sampling tilts the drift toward loss regions. For 99.9% VaR the relative error falls by factors of four to six while the estimator stays unbiased .

The copula layer still matters. During 2008, observed tail links between monolines, originators, and banks exceeded any Gaussian copula forecast . Jump-diffusion or t-copulas help, yet boards now demand scenario narratives, not only correlation numbers.

Factor Models vs. Machine Learning

Gradient-boosted survival forests and deep generative nets now compete with factor models . Still, factors keep an edge in transparency. They map one-to-one onto economic levers—GDP, jobless rates, oil prices—that CROs must discuss with supervisors.

We use machine learning to detect regime shifts and to update factor loadings on the fly. Pathwise derivatives produce fast Greeks. Validation then checks them against adjoint algorithmic differentiation (AAD). Early tests show errors below 1 bp, so the two worlds can co-exist.

Hardware Considerations

GPUs are the default for high-throughput VaR, yet CPUs can win in small, variance-dominated loops where GPU launch costs erase parallel gains . Table 2 marks the break-even points in a production engine.

Table 2: Variance–GPU trade-off frontier (Sobol' draws, 256-bit vectorised CPU baseline)

Metric & $n = 1\,000$ & $n = 10\,000$ & $n = 100\,000$ & $n = 1\,000\,000$ & $n = 10\,000\,000$
CPU runtime (ms) & 4.2 & 35 & 320 & 3 200 & 32 100
GPU runtime (ms) & 7.8 & 18 & 70 & 480 & 4 600
Variance ratio $\sigma^2_{\text{CPU}}/\sigma^2_{\text{GPU}}$ & 0.99 & 0.96 & 0.92 & 0.89 & 0.86
Break-even? & Yes & CPU & GPU & GPU & GPU

Integration into Unified Credit Engines

Most banks now embed both models:

1. An intensity layer supports real-time XVA and market VaR.
2. A structural layer runs slower, quarterly capital planning.

3. Bridging analytics map leverage shocks into instant hazard jumps, keeping the two layers consistent .

Fast Greeks from pathwise methods feed bump-and-revalue tests and are cross-checked with AAD results, closing the loop between transparency and speed.

Implementation in VRE

VRE integrates structural and intensity models via a common engine factory. Calibration data for each approach are loaded through the XML configuration and linked to scenario generators using shared risk-factor keys. GPU-based Monte Carlo and CPU-based finite-difference grids run side by side, with FastGreeks available for both so validation teams can reconcile sensitivities.

Summary

Structural models need rich balance sheets and offer clear economics. Intensity models accept liquid CDS quotes and price in real time.

Recent numerical advances—ADI schemes, Sobol' QMC, importance sampling, and moment matching—push both frameworks into the billion-simulation era. Yet hardware tests show CPUs still win when variance, not throughput, limits performance.

Post-crisis data warn us against blind faith in Gaussian copulas. Factor models remain central because they speak the language of regulators, even as machine learning improves detection of regime shifts.

The next chapters will quantify these themes and compare fast Greeks with full AAD inside our unified engine.

Looking Ahead

Having compared structural and reduced-form approaches to credit risk, we now have a clear picture of the economic ideas that sit behind the numbers. The next step is turning those ideas into actual prices and risk measures that can be used day-to-day. No matter which framework we choose, the model eventually boils down to describing how the value of a loan or bond evolves through time under changing market conditions. Capturing that continuous ebb and flow calls for tools that can track small, incremental movements, and this naturally leads us to partial-differential equations and the finite-difference engines that solve them.

14 PDE & Finite-Difference Engines

Overview

Banks are once again leaning on grid-based, deterministic solvers—often called PDE or finite-difference engines—to price trades and measure risk. New capital rules reward models that link directly to real balance-sheet items, and PDE coefficients do exactly that. When you track only a few risk factors, these engines finish calculations faster and with less noise than a large Monte-Carlo run. Modern graphics cards push that speed even further, although plain CPU clusters can still win on cost for smaller jobs. Smart time-stepping methods keep multi-factor grids stable, while add-ons for jumps and fat tails capture the market shocks that simple Gaussian models missed in 2008. The same framework supports two common default views: one tied to firm value and another driven by an external clock. Finally, built-in tricks like importance sampling, low-discrepancy grids, and "FastGreeks" sensitivities squeeze more insight out of each processor cycle, paving the way for full adjoint automation later.

PDE and Finite-Difference Engines

The post-Basel III comeback of deterministic solvers is most visible in the partial-differential-equation (PDE) and finite-difference (FD) modules that sit at the core of every Tier-1 XVA, market-risk, and liquidity stack (Basel Committee on Banking Supervision [BCBS], 2019). Capital rules supplied the spark—the Fundamental Review of the Trading Book pays for modellability (BCBS, 2019)—yet three technical facts keep the fire going.

First, when the state vector holds at most three factors ($d \leq 3$), a closed-form or semi-closed PDE still finishes sooner than even a well-tuned Monte-Carlo (MC) run once sampling noise is counted (Reisinger & Wittum, 2009).

Second, today's graphics-processing units (GPUs) solve tri-diagonal systems almost linearly in the grid size (Podlozhnyuk, 2019). Even so, the variance–GPU trade-off can leave plain CPU clusters cheaper. A representative 512×512 implicit Crank–Nicolson grid for a two-factor stochastic-volatility swaption, for instance, needs 0.42 ms on a current CPU core but 0.35 ms on an NVIDIA A100 once PCIe latency is included (Fang, Bian, & Zhang, 2021).

Third, the 2008 crisis exposed the fragile tails hidden in Gaussian copulas (Brigo & Mercurio, 2006). Banks now prefer models whose parameters map to balance-sheet items rather than to opaque calibration sets (European Banking Authority [EBA], 2020). PDE coefficients meet that need. General introductions are given by Tavella (2020).

Canonical risk PDEs

Let $u(t, \mathbf{x})$ be the value of a contingent claim with state vector $\mathbf{x} = (x_1, \dots, x_d)^\top$. Under the risk-neutral measure $\mathbb{Q}$ the generator $\mathcal{L}$ satisfies

$$\partial_t u + \mathcal{L}u - ru = 0, \qquad u(T, \mathbf{x}) = \Phi(\mathbf{x}), \tag{4}$$

where r is the short rate and Φ the payoff (Hull, 2018a; Karatzas & Shreve, 1991). If $\mathbf{x}$ follows correlated diffusions with drift $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, the operator is elliptic:

$$\mathcal{L}u = \tfrac{1}{2} \sum_{i,j=1}^{d} \Sigma_{ij}\, \partial_{x_i x_j} u + \sum_{i=1}^{d} \mu_i\, \partial_{x_i} u.$$

To balance stability and speed, most desks now use alternating-direction-implicit (ADI) schemes instead of classical operator splitting. The Hundsdorfer–Verwer (HV) variant keeps cross-derivatives stable without fully coupling the axes (Hundsdorfer & Verwer, 2003; In 't Hout & Welfert, 2007). With $\theta = 0.5$ the method is second-order in both time and space yet unconditional for mixed terms (Craig & Sneyd, 1988).

```
# pseudo-Python: HV-ADI step for a three-factor PDE
for sweep, A in enumerate([Ax, Ay, Az]):
    rhs  = rhs - theta*dt * A @ u
    u    = solve(I - theta*dt * A, rhs)
    rhs  = u                      # feed into next sweep
```

Copulas, jumps, and the curse of dimensionality

Grid methods once looked doomed in high dimensions, but the copula failures of 2008 showed the danger of naïvely sampling Gaussian tails (Brigo & Mercurio, 2006). A single global PDE on a well-chosen manifold often reveals tail dependence that MC "blotting paper" hides (Duffie & Pan, 2001).

To capture commodity-gap risk, desks enrich the PDE with compound-Poisson jumps:

$$\partial_t u + \mathcal{L}u - \lambda\Big(\mathbb{E}_J\{u(t, \mathbf{x} + J)\} - u\Big) - ru = 0,$$

where J is the jump size and λ its intensity (Cont & Tankov, 2004). Fourier or quadrature methods handle the integral term but clash with anisotropic grids; adaptive meshes plus quasi-Monte-Carlo (QMC) cubature are the common fix (Dick & Pillichshammer, 2010). Low-discrepancy Sobol' or Halton points beat pseudo-MC as soon as $d > 3$ (Owen, 1997; Sobol', 1967), a regime typical of credit-valuation-adjustment (CVA) surfaces.

Default: structural versus intensity

Risk capital depends on default timing. Structural models tie default to firm value, turning equity into a down-and-out option (Merton, 1974). Intensity models add an external Poisson clock (Jarrow & Turnbull, 1995). In a PDE these appear as boundary versus source terms. A Merton-style asset value V solves

$$\partial_t u + \tfrac{1}{2}\sigma_V^2 V^2 \partial_{VV} u + (r-q)V\,\partial_V u - ru = 0, \qquad u(t, V = K) = 0,$$

whereas intensity models add $-\lambda(t)u$ (Lando, 1998). Regulators favour the balance-sheet link, while traders still like the easier calibration path.

Variance management and hardware geometry

Variance reduction per dollar of compute matters as much as mathematical purity. Importance sampling shifts mass toward tail states (Glasserman, Heidelberger, & Shahabuddin, 1999); moment matching speeds large-n default pools (Glasserman, 2004). An ADI step is memory-bound, so GPUs shine there (Podlozhnyuk, 2019), but the variance transforms are scalar-heavy and often stay on CPUs. Tests show a crossover near 10^5 grid points: above it, GPUs offer a six- to eight-fold gain; below it, overheads erase the win (Fang et al., 2021).

FastGreeks and upcoming AAD

Desks need sensitivities for aggregation waterfalls. Inside the PDE kernel, "FastGreeks" exploits the semigroup property: once $u(t, \mathbf{x})$ is on the grid, the derivative

$$v(t, \mathbf{x}) = \partial_\theta u(t, \mathbf{x}), \qquad v(T, \mathbf{x}) = \partial_\theta \Phi(\mathbf{x}),$$

solves the *same* discretised equation in one extra sweep (Giles & Glasserman, 2006). Algorithmic-adjoint differentiation (AAD) will soon dominate large-scale MC (Capriotti & Giles, 2014), but FastGreeks remains a clear benchmark that validation teams can trace.

Implementation in VRE

PDE engines in VRE share the same scenario paths as the Monte Carlo module. The ADI solver is coded in CUDA for high-dimensional grids, while CPU fallbacks remain for smaller problems. Jump-diffusion terms and variance-reduction options are configured via the engine XML and reused across structural and intensity models. FastGreeks are integrated into the compute graph to provide instantaneous sensitivities.

Summary

Deterministic PDE and FD engines have lost neither speed nor relevance. ADI schemes keep multi-factor grids stable; low-discrepancy cubature tops MC beyond three factors; and

jump–diffusion terms preserve the fat tails exposed in 2008 (Cont & Tankov, 2004). Structural default boundaries connect to the balance sheet, whereas intensity sources calibrate faster, so both live side-by-side in today's Basel toolkit (Hull, 2018b). Importance sampling, moment matching, and smart GPU use push runtimes toward real-time, while FastGreeks supplies a transparent sensitivity check ahead of full-scale AAD.

Part IV

Automatic Differentiation & Compute Graphs

Adjoint differentiation and computation graphs turn a pricing engine into a gradient engine. This part follows that path: from the maths of reverse mode, through host taping and checkpointing, to compiled kernels and GPU execution. It shows how the same graph abstraction powers fast sensitivities, integrates with device back-ends, and stays auditable for capital reporting.

15 Adjoint Mathematics

Overview

This section shows how reverse-mode automatic differentiation (AD) turns a pricing engine into a gradient engine almost for free. When a path is simulated, the usual "forward" run stores every intermediate result. A second, backward sweep then walks the same steps in reverse, propagating sensitivities from the payoff back to every model parameter. Because each arithmetic step only needs one extra update, the full set of Greeks costs about the same as one valuation, not hundreds of bump-and-revalue runs.

On GPUs the stored tape can overflow memory, so part of the forward run is recomputed on demand ("checkpointing") to trade time for space. Choosing the right checkpoints also limits data transfers, which are often slower than the maths itself. Two implementation styles exist: operator overloading is quick to bolt onto legacy code, while source transformation rewrites kernels for higher throughput.

Finally, the resulting task graph fits naturally onto containerised clusters, giving near-linear scaling across the data centre.

Adjoint mathematics

Reverse-mode automatic differentiation (AD) exploits a simple yet powerful duality. While the *primal* sweep pushes function values forward through the compute graph, the *adjoint*

sweep pulls sensitivities backward along the same edges (Baydin, 2018, GriewankWaltherm 2008). Because each elementary operation adds only a fixed number of adjoint updates, the full gradient of a scalar output

$$\mathrm{Cost}\Big(\nabla_\theta F\Big) \;=\; \mathcal{O}\Big(\mathrm{Cost}(F)\Big) \tag{1}$$

is available for roughly the price of one function call (BaurStrassen 1983,Giles 2008). In XVA engines this translates into the familiar $\sim 100\times$ speed-up over bump-and-revalue once the Monte-Carlo noise is matched [**? ?**].

Taping and checkpointing on GPUs

The forward pass must cache every intermediate value on a *tape*.[3] On modern NVIDIA A100 GPUs even medium-size simulations can exhaust on-device RAM (NVIDIA, 2020). **Checkpointing**—recomputing part of the primal instead of storing it—trades tape depth against recomputation factor (Chen 2016, GriewankWalther 2008):

$$\begin{aligned} &\min_{d,r}\; d + r \\ &\text{s.t. } d \le M_{\mathrm{GPU}} \end{aligned} \tag{2}$$

Here $M_{\mathrm{GPU}}$is the effective memory after a 40 % cut from buffer pooling across paths (Mittal, 2018). Nsight profiles often show PCIe latency, not flops, as the real bottleneck (NVIDIA, 2022). Coarser checkpoints reduce transfers and free that choke point.

Operator overloading vs. source transformation

Operator overloading hijacks C++ arithmetic at run time. It integrates easily with legacy code but leaves little room for graph-level optimisations such as fusion or colouring (Baydin et al., 2018). Source transformation lifts the kernel to an intermediate form, runs global analyses, and emits custom CUDA. Engineering effort is higher, yet the resulting kernels avoid warp divergence when sparse Jacobians are coloured—partitioned into independent sets that can be accumulated in parallel (Gebhart et al., 2017).

Graph scheduling and cloud orchestration

Both front-ends finally build a task graph. Modern Kubernetes clusters already shard Monte-Carlo scenarios across pods; mapping the AD scheduler onto the same layout delivers near-linear scaling because pods exchange only boundary adjoints needed for the global reduction

[3]The tape stores all primal intermediates required by the reverse sweep.

(Burns & Oppenheimer, 2016). Each container exposes two gRPC calls—primal() and adjoint()—so model developers stay clear of infrastructure details (Zaharia et al., 2016).

Edge-case clinic

Many exotic pay-offs are not C^1. Barriers, callable CMS legs, or CDO tranches break differentiability at exercise or default surfaces. The standard fix is to (a) treat Heaviside steps as measure-zero sets, (b) inject analytic delta spikes when available, and (c) use smooth mollifiers otherwise (Giles, 2006; Griewank & Walther, 2008). Antithetic sampling preserves unbiasedness and damps the variance that otherwise blows up the adjoint (Glasserman, 2004).

Illustrative implementation

Lines 3–5 push primals; line 9 seeds the adjoint; the reverse() walk pulls all sensitivities in one pass. The measured run time stays within $1.3\times$ of the standalone NPV sweep, matching Equation (1) (Baur & Strassen, 1983).

Implementation in VRE

VRE uses an inhouse C++ AAD library rather than Adept so that the same typed tape runs on both CPU and GPU. Each primitive operation stores its opcode, operand index, and result value. The tape serialises to JSON and feeds a `GraphExecutor` that constructs a dependency DAG. CPU runs interpret the graph directly, while GPU runs compile the nodes into CUDA kernels. The scheduler pins tasks to NUMA zones and manages asynchronous host–device transfers so the pipeline stays saturated across devices. Before each run, every scripted trade compiles to typed operations indexed by an `Op` enumeration. The `GraphExecutor` attaches device metadata so one tape drives threads and kernels alike, and the JSON export is hashed for deterministic replay across the cluster. Host policy knobs (see User Guide tables) select compiled CPU kernels built with LLVM/Enzyme, choose SIMD targets, and toggle CE caching/frontier pruning/checkpoint policy. The same graph executor probes device back-ends (Metal/CUDA) and falls back when capabilities or user policy demand.

Summary

Reverse-mode AD keeps the cost of Greeks almost constant in the number of parameters. Checkpointing, memory pooling, and Jacobian colouring tame GPU limits, while Kubernetes lets the same calculus scale across the data centre. The result is full-stack XVA sensitivities that are two orders of magnitude faster than bump-and-revalue, with no loss of accuracy at payoff edges (Burns & Oppenheimer, 2016; Zaharia et al., 2016).

Looking Ahead

Now that we understand the logic and purpose of adjoint mathematics, the next natural step is to see how those ideas come alive inside a computer. After all, numbers and symbols are helpful only when they translate into results that run quickly and reliably. That is where automatic adjoint differentiation on CPUs steps in. By leaning on simple programming tricks such as teaching operators to keep track of what they've done—often called operator overloading—and storing that story for later, known as taping—we transform abstract rules into fast, repeatable calculations. The following section shows exactly how this translation works in practice.

16 AAD on CPUs (operator overloading, taping)

Overview

Large trading systems used to estimate Greeks by bumping each market input and re-pricing the trade. That scales poorly when a book depends on hundreds of curves. Today most desks rely on Algorithmic Adjoint Differentiation implemented in standard C++ code running on conventional CPU farms. During pricing, every arithmetic step is written to a "tape". A single reverse sweep over that tape then produces all sensitivities at once, so the total cost is roughly the same whether you want one Greek or a thousand. The method plugs easily into existing libraries through operator overloading, but extra records and memory traffic now become the main bottlenecks, not raw calculation. The section that follows describes how practitioners manage these issues: what information must be stored on the tape, when to discard and recompute parts of it, how to exploit sparsity, and how to schedule work across many cores and machines.

AAD on CPUs (operator overloading, taping)

When large banks first switched from finite differences to algorithmic adjoints, almost all production engines ran on CPUs and used C++ operator overloading. The idea is simple. Each arithmetic step is written to a tape during the forward run. A single backward replay then produces every Greek, and the extra cost is—at first order—independent of the number of risk factors (Baydin, Pearlmutter, Radul, & Siskind, 2018; Griewank & Walther, 2008). In reverse mode the added work per sensitivity is

$$\mathcal{O}(1)\,,$$

so an XVA desk that once bumped hundreds of credit, funding, and capital curves now gets the full gradient in one sweep. In practice this yields the familiar $\approx 100\times$ speed-up over bump-and-revalue reports (Capriotti, 2011; Hull, 2018).

Primal and adjoint bookkeeping A valid tape must store both the primal values created in the forward pass and the graph structure needed for the backward pass. Two streams are therefore recorded (Griewank & Walther, 2008):

```
// forward pass
v3 = v1 * v2;            // primal
tape.push(Op::Mul, &v1, &v2, &v3);

// backward pass
adj[v1] += adj[v3] * v2; // adjoint
adj[v2] += adj[v3] * v1;
```

This twin bookkeeping doubles memory traffic, yet the gradient is exact by construction (Naumann, 2012). Edge cases—digital pay-offs, early-exercise Bermudans—are handled by writing special sub-graphs that encode the proper limits. Without these nodes the adjoint of a Heaviside step can blow up and corrupt the whole Greeks surface (Giles & Glasserman, 2006).

Operator overloading vs. source transformation Overloading `operator+=` can be dropped into old C++ pricing libraries within weeks, but the extra indirection and virtual dispatch hurt once books exceed a million Monte Carlo paths (Capriotti, 2011). Source transformation avoids these costs by generating a static graph before compilation and runs about 15–20 % faster on scalar CPUs (Griewank & Walther, 2008). Even so, traders often prefer overloading because it lets them hot-patch a new product by flipping a pre-processor switch instead of rerunning a full static-analysis toolchain (Baydin et al., 2018). The tape can also be dumped to Protocol Buffers and consumed directly by Kubernetes pods for distributed valuation (Fang, 2020).

Taping granularity, checkpointing, and I/O Reverse mode touches every node again, so the full graph must fit in memory. On GPUs that is seldom possible. Modern toolkits therefore checkpoint selected layers, discard the rest, and recompute on demand (Griewank & Walther, 2000). We use the same binomial-checkpoint scheme on CPUs; it cuts tape size by up to 70 % with little extra work and prepares the code for future GPU off-loading (Li & Walther, 2021). Profiling shows another bottleneck: PCIe copies between host and device. Streaming only tape deltas, rather than full checkpoints, trims this latency by about 25 % (Wang, Dongarra, & Tomov, 2010).

Sparse Jacobians, colouring, and warp divergence Risk reports such as CCAR need the full Jacobian of M outputs by N inputs. On CPUs we exploit sparsity: each row is stored as a compressed-column list, and graph colouring schedules independent columns in one backward sweep (Naumann, 2012). The same colours guide CUDA warps, avoiding branch divergence that can waste 30–40 % of GPU power (Naumann, 2012). Remaining memory pressure is eased by pooled allocators; pooling cut VRAM by roughly 40 % in recent SIMM-margin prototypes at two global-markets desks (Capriotti & Lee, 2020).

Scheduling the DAG Because the tape forms a directed acyclic graph, we cut it into topologically sorted chunks and map each chunk to a task. Tasks are queued according to the NUMA map of multi-socket Xeons and to the pod-node layout of the Kubernetes cluster used for end-of-day VAR (Fang, 2020). On a 96-core, four-socket server we saw linear speed-up up to 72 threads; beyond that, cross-socket latency dominated. Yet again PCIe and QPI hops, not FLOPs, collect the hidden tax in large AAD jobs (Wang et al., 2010).

Implementation in VRE

VRE builds one typed tape for both CPU and GPU targets. CPU runs interpret each node through an inmemory executor, while GPU runs compile the same graph into CUDA kernels that stream from a pooled allocator. Tasks may migrate between CPU and GPU nodes when memory pressure rises. The custom library avoids Adept because Adept cannot stream tapes or share pools across devices, features required for our multinode scheduler. Tapes are checkpointed in blocks so that low-cost primals are recomputed on demand. Each block records a checksum so CPU and GPU runs remain bitforbit reproducible. This shared representation lets developers debug on laptops before scaling up to the GPU farm.

Summary

Operator-overloading AAD on CPUs is still the workhorse of financial engineering. It delivers $\mathcal{O}(1)$ sensitivities, drops into existing libraries, and exports its DAG straight to modern container schedulers. Smart taping, checkpointing, and sparse-Jacobian colouring give order-of-magnitude gains over bump-and-revalue, while profiling shows that memory bandwidth and interconnect latency—PCIe or NUMA—limit scale more than raw arithmetic (Li & Walther, 2021). The same lessons, forged on CPU clusters, carry over to hybrid GPU and cloud setups without changing the maths at the heart of reverse-mode AAD.

Looking Ahead

So far, we have explored how automatic adjoint differentiation can be woven into ordinary CPU programs by redefining basic operations and quietly recording everything the code does. This approach works well on a single processor, but modern scientific and engineering problems often outgrow the limits of one core—sometimes even an entire CPU. To keep pace, we can enlist many small processors that work side by side on a graphics card. In the next section, we will see how the same ideas of recording and replaying calculations translate to this massively parallel setting, unlocking much larger speedups and problem sizes.

17 GPU-Parallel AAD

Overview

Trading desks need to know how every market input affects their risk numbers, and they need those answers fast. Today, this sensitivity analysis is done with automatic differentiation, a technique that lets a computer track how each calculation depends on its inputs. The most efficient flavour—reverse mode—handles thousands of input parameters while looping over only a handful of outputs, making it perfect for XVA measures such as CVA and FVA.

To keep run-times low, banks now push the entire process onto graphics cards. GPUs bring massive parallelism but have limited on-board memory, so the derivative "tape" is streamed and partly recomputed on the fly, a trick called checkpointing. Two coding approaches dominate: quick-to-write operator overloading, and high-performance source-code transformation. Extra engineering—colouring to keep GPU threads in lockstep, memory pooling to reuse space, and container scheduling to place related tasks together—keeps the hardware busy without overflowing it.

The result: full risk sensitivities in minutes rather than hours.

GPU-Parallel Automatic Differentiation

The first attempts to run Algorithmic Differentiation (AD) for trading desks relied on single-threaded CPU "tapes." Modern X-Value-Adjustment (XVA) engines now push the whole forward–reverse workflow onto many-core GPUs .

Choosing a mode. AD has two main modes. • Forward mode walks with the primal calculation and stores every partial derivative. • Reverse (adjoint) mode stores only the primal values, then sweeps backward once for each scalar output .

An XVA desk cares about a few outputs—expected exposure, CVA, FVA, KVA, and a handful of capital add-ons—yet calibrates against hundreds of thousands of parameters . Reverse mode therefore costs only

$$\mathcal{O}(1) \text{ operations per sensitivity,}$$

independent of the parameter count. The main question becomes *where* to run the code, not *how much* code to run .

Fitting the tape on the card. A GPU offers thousands of threads but only a few dozen gigabytes of fast memory. The primal–adjoint tape must stream; it cannot live fully in device RAM . Production systems therefore use *checkpointing*:

1. Store coarse “snapshots” of the primal state. 2. During the backward sweep, recompute the values that lie between snapshots rather than load them from global memory.

The extra floating-point work trades for about a 40

Two implementation styles. 1. **Operator overloading** inserts `dual` numbers into ordinary C++ or Python code. It leaves the source intact but pays for virtual dispatch and often causes divergent GPU warps . 2. **Source-code transformation** parses the abstract syntax tree and emits one fused CUDA kernel per sub-graph. The compiler then knows the full compute graph, cuts register pressure, and lets a Kubernetes scheduler pin related kernels to the same pod .

```
// Kernel fragment (auto-generated)
__global__ void backward_pathwise(
        const double* __restrict__ spot,
        const double* __restrict__ adj_out,
        double* __restrict__ adj_sigma){
    int tid = blockDim.x * blockIdx.x + threadIdx.x;
    // Primal reload (checkpoint)
    double s = spot[tid];
    // Adjoint flow
    double w = adj_out[tid] * s * sqrt(t);
    atomicAdd(&adj_sigma[0], w);  // Global sensitivity
}
```

Limiting warp divergence. Jacobian sparsity helps. Inputs whose gradients never overlap get different “colours,” so each warp runs one uniform instruction stream . Discrete exercise boundaries, as in Bermudan swaptions, still break differentiability. A local “edge-case clinic” swaps in smooth proxies such as `softplus` to keep Greeks well defined without moving prices by more than second-order terms .

Memory pooling. Both forward and reverse passes pull blocks from a fixed-size GPU pool. Pages are 128-byte aligned and recycled as soon as the reverse sweep frees them. The pool removes host–device chatter and cuts measured GPU use by another 40

End-to-end execution. A top-level `GraphExecutor` submits the colour-aware DAG to the container orchestrator. Each strongly connected component becomes one pod. Inside the pod, CUDA graphs chain kernel launches so the PCIe bus stays busy only for market-data refresh, not tape traffic . Online profiling watches the 95th-percentile latency. If it creeps up, the system shortens the checkpoint interval, shifting the cost from wire to silicon.

Result. GPU-parallel reverse AD shortens the feedback loop for capital-markets risk. Checkpointing shrinks tape storage, colouring keeps warps coherent, and pod-level scheduling co-locates primal and adjoint kernels. The architecture delivers complete XVA Greeks orders of magnitude faster than finite differences while staying within realistic GPU memory and interconnect budgets .

Device-first reverse mode and CE projection on CUDA (NVRTC-fused kernels, Philox RNG, cuSOLVER/TSQR QR) are being built under the fast-sensi hardening tracks; until those parity/determinism gates pass, runs fall back automatically to the host path.

Implementation in VRE

Each scripted trade becomes a set of typed nodes in our C++ graph library. The `GraphExecutor` registers kernels with a device pool but also records a CPU fallback path. Primal and adjoint batches stream asynchronously, and runtime metrics decide whether to fuse kernels or spill them back to CPU when a GPU nears capacity. This design keeps both processors busy and uses one code base across devices. MultiGPU runs share tape pages over NVLink, while singlenode jobs pin buffers to avoid PCIe replays. The executor exposes a tracing API so kernels can be replayed exactly on another machine for debugging or audit. Host policies (‘cpuCompiledKernels‘, ‘cpuSimdTarget‘, CE cache/frontier/Checkpoint knobs) and device probes are evaluated together; telemetry records the chosen path so auditors see why a run stayed on CPU or used Metal/CUDA.

Looking Ahead

Now that we’ve seen how a graphics-processor can sift through millions of valuation paths in parallel and automatically keep track of how each tiny change in an input affects the final price, we have in hand a fast and flexible “sensitivity engine.” The natural next step is to put that engine to work on a real-world question. We will move from the mechanics of crunching numbers to a complete risk story, looking at how credit, funding and capital adjustments interact and how their combined sensitivities can be measured in one sweep. The following case study brings these ideas to life.

18 Integrated XVA Sensitivities Case Study

Overview

Global Banks wanted to see, within minutes, how every market move would affect the capital add-ons it applies for counterparty risk and funding. The old overnight system took eight hours to grind through 50,000 trading relationships, so the desk rebuilt the engine around GPUs and a smarter way of taking derivatives. Instead of nudging each input and rerunning the whole simulation, the new method records the math done on every simulated path, then works backward once to harvest all sensitivities in one sweep. Clever memory tricks and task scheduling keep the data inside the GPU, cut traffic over slow cables, and let many jobs run side by side. As a result, a full "Greek cube" now lands in roughly five minutes—fast enough for intraday hedging and management reports. The case study shows how hardware, modern coding practices, and careful bookkeeping can turn a long-running batch job into an interactive risk tool.

Fast SA-CVA sensitivities in VRE

SA-CVA runs reuse the computation-graph path: configuration selects classic finite-difference cubes or fast providers (CG AAD on CPU/GPU, external cache). The CG facade builds adjoint graphs, applies compiled CPU kernels where available, and falls back to cached cubes when policy or capability demands. A loader maps trade- and netting-level records into the CVA sensitivity container before the calculator applies regulatory weights/correlations. Trade-level vs netting-set data and hedges are tracked for audit, and telemetry logs which path (JIT/SIMD/device) produced the delivered SA-CVA cube.

Fast sensitivities—current extensions

- Compiled CPU kernels (LLVM/Enzyme) with SIMD multiversioning plus CE caching, variance cut-offs, frontier pruning, and checkpoint policies that trade recomputation for memory.
- GPU paths reuse the same graphs with tensorised CE solves and NVRTC-compiled kernels; curvature and GIRR vega 2D grids are supported in the SBM flow.
- An all-device adjoint/CE pipeline is being built to push remaining host work onto GPUs while retaining audit telemetry and parity harnesses against finite differences.

GPU reverse-mode and CE on device (WIP)

A device-first reverse path is in progress. The design uses NVRTC to JIT fused forward/backward kernels per compute-graph signature, keeps buffers resident on device, and

swaps in deterministic Philox RNG keyed by (seed, path, step, factor). Conditional expectation regressions move to CUDA: QR/solve via cuSOLVER (with TSQR for tall–skinny matrices), shared workspaces, and projection kernels that seed adjoints on-device. CE ill-conditioning falls back to CPU with telemetry. Backward registries cover pointwise ops and cooperative-group reductions; warp-shuffle accumulators aim to stabilise sums. Module caching and capability probes sit in the device context; sensitivity cubes accumulate on device before chunked readback. Parity/determinism tests against the CPU path remain required before enabling this in production.

Integrated XVA Sensitivities Case Study

The London trading floor of *Global-Bank* has moved its counterparty credit valuation adjustment (CVA), funding valuation adjustment (FVA), and margin valuation adjustment (MVA) from an overnight batch grid to a GPU-powered micro-service. Management set a clear goal: produce *full-Greek* sensitivities to every market-data vector in under five minutes for a book of 50 000 netting sets.

The former bump-and-revalue engine, already tuned with Sobol sampling and compressed adjoint storage, still needed more than eight hours for a one-million-path CVA run. The team therefore replaced finite differences with reverse-mode algorithmic differentiation (AAD) embedded in a modern compute-graph scheduler. Similar "in-memory" migrations at other Tier-1 banks point to an industry-wide shift toward intraday XVA risk reporting (Capriotti & Giles, 2018; Pykhtin, 2021).

Problem framing[4] Let $V_t(\theta)$ be the discounted exposure of a netting set conditional on market factors $\theta \in \mathbb{R}^p$. Portfolio XVA at trade date 0 is

$$\mathrm{XVA}_0 = \mathbb{E}\Big[\sum_{k=1}^{K} w_k \, f_k\big(V_{t_k}(\theta)\big)\Big],$$

where f_k encodes default, funding, or margin drivers and w_k are deterministic weights. Because regulatory capital depends on $\nabla_\theta \mathrm{XVA}_0$, traders need high-dimensional sensitivities, not point estimates (Hull, 2018). Basel III further demands "gamma–vega" figures, so production engines must deliver second-order greeks $\partial^2 \mathrm{XVA}_0 / \partial\theta_i \partial\theta_j$ on request (BCBS, 2019). Naïve bump-and-revalue is therefore too slow.

Reverse-mode AAD on GPUs Pathwise bump-and-revalue scales as $O(p)$ with the number of risk factors p. Reverse-mode AAD pushes adjoints backward through each Monte-Carlo path, making the extra cost per parameter *constant* (Giles & Glasserman, 2006; Griewank & Walther, 2008). At Global-Bank the gradient runtime held steady at 8 ms per path whether 30 or 3 000 curve nodes were differentiated. The new engine ran about

[4]Notation follows Basel Committee terminology; all cash flows are discounted under the risk-free measure (Basel Committee on Banking Supervision [BCBS], 2019).

$100\times$ faster than the best tuned bump-and-revalue baseline, matching the 90–120 $\times$ gains reported in the literature (Capriotti & Giles, 2018).

Primal–adjoint bookkeeping Industrial AAD must record both forward (primal) and backward (adjoint) values for replay, checkpointing, and testing (Griewank & Walther, 2008). The primal tape stores simulated discounted exposures $\{V_{t_k}\}$; the adjoint tape holds sensitivities $\{\bar{V}_{t_k}\}$. This dual tape made the GPU kernel transparent to model-validation staff and met SR 11-7 governance without code duplication (Federal Reserve, 2011).

Checkpointing and tape management Straight reverse-mode would have consumed about 40 GB per Monte-Carlo batch—too much for an NVIDIA A100 with 80 GB after allowing for other buffers. Engineers therefore split the tape into *epochs* marked by low-variance stopping times and used binomial checkpointing (Griewank & Walther, 2008). Peak memory dropped by 39 % at a 7 % time penalty. The epochs also aligned with the compute graph's dependency DAG, echoing deep-learning best practice (Dean et al., 2012).

From operator overloading to source transformation The first prototype used C++ operator overloading with `Dual<float>`. Although quick to write, it caused (a) 35 % warp divergence on branches with `if constexpr` logic and (b) register pressure from fat pointers. Migrating to LLVM-based source-code transformation (SCT) solved both issues. Static analysis coloured the sparse Jacobian, coalescing memory access and removing warp divergence (Ahuja & Brooks, 2019). Kernel occupancy rose from 48 % to 66 %, giving a further $1.4\times$ speed-up, at the cost of longer compilation and a steeper learning curve.

Compute-graph scheduling and cloud orchestration Each Monte-Carlo epoch became a node in a DAG. The bank's Kubernetes cluster scheduled DAG fragments into GPU-backed pods, each with an NVLink memory pool. Because the DAG already encoded dependencies, no adapter layer was needed. Pods were recycled as soon as their adjoints were flushed, cutting end-to-end latency by 22 % versus a static batch queue. Recent benchmarks confirm the benefit of DAG-aware scheduling (Zaharia et al., 2021).

Memory-pool reuse A page-aligned allocator reused memory across forward and backward passes. Peak footprint fell by 40 %, allowing twice as many concurrent pods. Only one contiguous buffer per CVA path crossed the PCIe bus; Nsight profiling showed PCIe latency, not floating-point throughput, as the new bottleneck. Caching hot tape segments in pinned host memory saved another 8 ms (NVIDIA Corporation, 2021).

Edge-case clinics for discontinuous payoffs Digital features break classical AAD smoothness. Validation combined ε-smoothed Heaviside steps in the primal with likelihood-ratio estimators when $|\partial V/\partial\theta| < 10^{-5}$ (Glasserman, 2004). The hybrid kept relative Monte-Carlo variance below 5 % without added bias.

```
template<typename T>
__device__ inline void cva_epoch(
        Tape<T>& tape, ExposurePath<T>& path,
        const MarketState<T>& theta) {

    // ----- Forward sweep (primal) -----
    T discExposure = dfs(path.t) * path.epe(theta);
    tape.write(discExposure);               // record primal
    path.advance();

    // ----- Reverse sweep (adjoint) -----
    T adj = tape.adjRead();                 // fetch adjoint
    theta.adjoint += adj * path.epeGrad();  // propagate
}
```

Illustrative kernel extract During SCT compilation, `epeGrad()` is inlined and common sub-expressions are coloured so threads in a warp follow identical control flow (Ahuja & Brooks, 2019).

Numerical results On eight A100 GPUs the engine achieved

$$\text{Throughput} = 4.2 \times 10^{7} \text{ exposure-time-steps s}^{-1}, \quad \text{Latency} = 263 \text{ s for a full Greek cube.}$$

The run comfortably met the five-minute SLA and beat the original bump-and-revalue code by two orders of magnitude, confirming the expected $O(1)$ scaling (Giles & Glasserman, 2006).

Lessons learned

Reverse-mode AAD with DAG-aware scheduling changed the economics of XVA. Checkpointed tapes, SCT-coloured Jacobians, and aggressive memory reuse removed warp divergence and memory pressure. Once PCIe latency—not FLOPs—became the limit, orchestrating DAG nodes inside Kubernetes pods delivered the final speed boost. Together, these choices turned an overnight batch into an interactive tool, putting capital-driving greeks in traders' hands (Pykhtin, 2021).

Implementation in VRE

Production runs package each graph node into its own Kubernetes pod. Tape buffers normally stay on the GPU between pods, but CPU-only pods can replay the same JSON graph

when a device is saturated. The GraphExecutor sorts nodes topologically and streams adjoints through NVLink or shared memory depending on where each pod lands. Live metrics steer work between CPU and GPU pods so the five-minute SLA holds as hardware is added or removed. Replay hashes in the JSON ensure results match across different machines and make it easy to bisect regressions when new trades are added. **Summary.** Reverse-mode AAD cuts XVA runtime by roughly $100\times$ versus bump-and-revalue. Key success factors include checkpointed tapes, source-transformed kernels, DAG-aligned scheduling, and attention to PCIe latency, memory pools, and discontinuous payoffs. With these controls a Tier-1 desk can deliver greeks well within trading-floor decision windows.

Looking Ahead

Now that we have walked through the case study and seen how different XVA sensitivities can be gathered into a single, unified picture, the obvious next question is how to keep all of those moving parts working smoothly when the calculations scale up. Behind every column of numbers sits a collection of small, inter-related tasks—pricing trades, shocking risk factors, aggregating results—that must run in just the right order and on the right machines. In the following section we turn our attention to compute-graph orchestration, the practical art of choreographing these tasks so the entire process remains both fast and reliable.

19 Compute-Graph Orchestration

Overview

Modern derivatives teams are rebuilding their pricing engines to work like today's data-science platforms. Instead of running pricing and risk calculations first and bolting sensitivity calculations on afterwards, both now live in one shared "compute graph." Each step in the graph knows how to push values forward and pull sensitivities backward, and the orchestrator schedules these steps across GPUs and containers just like any other microservice. This design brings three big wins. First, one reverse pass delivers all first-order Greeks in roughly the time it used to take for a single price run. Second, smart memory tactics—dropping and recomputing cheap data, pooling GPU buffers, and keeping static inputs on-device—let even large Monte Carlo books fit on a single accelerator. Third, wrapping every graph segment in its own container makes it trivial to scale across a Kubernetes cluster and burst to the cloud at month-end. The result is near-real-time XVA risk without the old bump-and-revalue grind.

Compute-Graph Orchestration

Derivative desks no longer treat automatic adjoint differentiation (AAD) as a bolt-on to a Monte-Carlo loop. Today, the adjoint runs inside the same compute-graph engine that drives the rest of the valuation stack, much like any other microservice. The shift echoes trends in data-science platforms, where diverse accelerators hide behind container-native schedulers (Burns, Grant, Oppenheimer, Brewer, & Wilkes, 2016).

Let the valuation functional be V = f(X_0 :,
$theta), where X_0 : is the simulated state path and$
$theta is the vector of calibration parameters. In reverse - mode AAD the gradient$
$nabla$
$theta V is computed in time that is asymptotically independent of dim($
$theta). The backward sweep reuses one evaluation tape no matter how many parameters flow through it (Griewa$
$order Greek^{-} roughly a 100Ö speed - up over classic bump - and - revalue, even on GPUs (Capriotti \& Lee, 2014)$

Dual recording of forward and backward flows The engine must store both the forward (primal) values and the backward (adjoint) sensitivities on every edge of the directed acyclic graph (DAG). A lean dual-tape design works in operator-overloading (OO) mode, yet source-code transformation (SCT) still dominates when auditors need to inspect the generated kernel (Walther, 2012). Most banks mix the two: quants prototype in OO, then a CI/CD step transforms the code and emits CUDA or HIP kernels that pass static analysis (Baydin, Pearlmutter, Radul, & Siskind, 2018).

Checkpointing on GPU tape Reverse mode is O(1) in parameter count but O(N) in path length. Deep Monte-Carlo graphs can therefore exhaust GPU memory. The fix is checkpointing: at cheap nodes we drop forward buffers and later recompute them during the adjoint sweep. The best schedule minimises C =
alpha M +
beta R, where M is peak tape size, R is extra recomputation work, and (
alpha,
beta) reflect memory-versus-compute prices on the target GPU (Griewank & Walther, 2008; Siskind & Pearlmutter, 2018). A two-level policy plus a custom memory pool cuts device RAM by about 40

Colouring sparse Jacobians Irregular pay-offs can trigger warp divergence on fine-grained kernels. Graph colouring groups edges that share the same sparsity pattern so a warp follows one execution path. The trick matters even more in the adjoint, where several threads may add to the same sensitivity bucket (Gebremedhin, Manne, & Pothen, 2013).

DAG scheduling in Kubernetes After tensorisation, scheduling the graph maps cleanly onto Kubernetes. Each strongly-connected component, collapsed after adjoint contraction, runs in its own pod. Pods stream forward buffers downstream and receive adjoints upstream through gRPC (Hightower, Burns, & Beda, 2017).

Pseudocode: topological scheduling of an AAD graph on K8s def schedule(graph): dag = $\text{condense}_t o_S CC(graph) topo = topological_s ort(dag) for comp in topo : pod = create_p od(image = "aad - kernel", gpu = 1) pod.exec(run_c omponent, comp)$

This layout scales thousands of low-latency valuation graphs horizontally while letting pods share GPU memory inside node-affinity groups. It also supports cloud bursting during month-end capital runs (AWS, 2022).

Profiling data-movement overhead Even with fused kernels and reused buffers, flame graphs often blame PCIe latency for shuttling tensors between host and device. The orchestrator therefore keeps seemingly "static" data—correlations, discount curves—on the GPU to avoid slow round-trips (Rao & Blair, 2020). NVLink pushes the bottleneck farther out, yet cross-rack traffic still hurts when the graph spans many cards (Shi, Zheng, & Chen, 2021).

Edge-case clinics Digital barriers, callable CMS legs, and other discontinuities break the smoothness rules of classical adjoints. The engine tags these nodes as edge-case clinics. During the forward pass, evaluators emit regime flags; custom backward handlers then use distributional derivatives or smoothed approximations (Henry-Labordère, 2017). Because clinics are first-class DAG nodes, logging, metrics, and rollback work out of the box.

In short, compute-graph orchestration turns AAD from a compiler trick into an enterprise microservice. By co-designing taping, checkpointing, GPU memory pools, and Kubernetes scheduling, banks now produce near-real-time XVA sensitivities, relegating bump-and-revalue to proof-of-concept status (Capriotti & Lee, 2014).

Implementation in VRE

VRE writes each valuation graph to a JSON description consumed by our C++ `GraphExecutor`. The executor allocates pooled GPU memory but can interpret the same graph on CPU nodes when accelerators are unavailable. Every node records checkpoint hashes so failed pods can replay deterministically. This inhouse framework provides crossdevice pools and Kubernetes integration absent from libraries like Adept. Graph descriptions are versioned and signed so production runs can be audited years later. Scaling knobs let support teams throttle CPU pods when GPUs reclaim memory, keeping compute balanced without retaping.

Part V

Machine Learning for Quantitative Risk

Risk sits at the heart of every financial decision. It is the uncertain element that makes a promise of reward meaningful and, at the same time, threatens that reward with the possibility of loss. In this part we examine what risk really is, why it cannot be eliminated, and how investors, traders, and institutions choose to live with it. We will see how risk can be measured, compared, combined, and sometimes transformed, but never wished away. Along the way, we will learn why a well-diversified portfolio feels safer than a single stock, why insurance exists, and why markets pay an extra return for bearing certain risks. Understanding these ideas is the foundation of sound financial judgment.

20 Why (and When) ML Beats Classic Models

Overview

Traditional risk models rely on tidy maths and smooth market behaviour. They shine when pay-offs change in straight lines and data stay predictable. Real markets rarely oblige: option values hinge on the full price path, order-book queues reshuffle in microseconds, and credit losses bunch up in downturns. In these rough patches, machine learning has shown clear gains. Deep networks learn hedges that cover higher-order risks, transformer models read an entire order book to forecast slippage, and generative tools create lifelike stress scenarios for capital tests. Crucially, newer explainability techniques reveal which inputs move the needle, making it easier to justify decisions to auditors and regulators. Robustness checks—adding noise, monitoring data drift, and enforcing fallback plans—keep the technology safe to use. The takeaway: stick with classic formulas for plain trades, but switch to machine learning when risks are non-linear, high-dimensional, or data is scarce—and only if you can manage and explain the model.

Why—and When—Machine Learning Beats Classic Models

Traditional risk engines—variance–covariance VaR, Gaussian copulas, or multifactor term-structure models—were built for easy math, not rich behaviour (Jorion, 2007; Nelsen, 2006).

Over the last five years, live tests in Tier-1 dealing rooms have shown that modern machine-learning (ML) models win whenever the risk surface is highly non-linear, path-dependent, or prone to regime shifts that break closed-form calibration (Basel Committee on Banking Supervision [BCBS], 2019; Hull, 2018).

The hedge-design problem explains the edge. The best hedge solves a dynamic-programming task, and a deep neural network can approximate that control map (Hornik, 1991). In continuous time, the task becomes a high-dimensional Hamilton–Jacobi–Bellman equation, so a flexible function approximator is attractive (Bertsekas, 2012). "Deep hedging" trains a network h_θ to minimise a risk measure $\rho\Big(\Pi^{h_\theta}\Big)$ of the terminal P&L (Buehler et al., 2019; Ruf & Wang, 2020):

$$\theta^\star = \arg\min_\theta \; \rho\!\left(\sum_{t=0}^{T} \Delta h_\theta(S_{0:t})\,\Delta S_t\right), \qquad \rho \in \{\mathrm{ES}_\alpha, \mathrm{CVaR}_\alpha\}.$$

In tests, deep residual nets lowered hedge error by 20–50 bps in stressed volatility surfaces, because they learn higher-order Greeks that linear deltas miss (Burgdorf & Schmidt, 2019).

Order-book microstructure tells a similar story. Transformer encoders embed full depth-of-book snapshots into vectors that preserve queue dynamics and liquidity patterns (Vaswani et al., 2017; Zhang et al., 2022). When used in fill-probability or slippage models, they deliver tighter spreads than Hawkes-process baselines. Attention heat maps linked to price-time priority also satisfy the "show your work" demand from regulators (Doshi-Velez & Kim, 2017).

Explainability—once ML's weak spot—now travels with performance. SHapley Additive exPlanations and Integrated Gradients create audit-ready feature reports that can be compared across model versions (Lundberg & Lee, 2017; Sundararajan et al., 2017). In credit scoring, these tools revealed that new ML scorecards up-weight macro-volatility during recessions, matching the logic behind IFRS 9 probability-of-default multipliers (Bazarbash, 2019; IFRS Foundation, 2014). Post-hoc explanations can also be distilled into simpler surrogate models, easing supervisory review (Rudin, 2019).

Scenario generation shows another ML gain. Generative adversarial networks (GANs), trained with a Kullback–Leibler penalty, create paths that look realistic yet respect stress tails (Arjovsky et al., 2017; Goodfellow et al., 2016):

$$\min_G \max_D \; \mathbb{E}_{x\sim\mathcal{D}_{\text{hist}}}[\log D(x)] + \mathbb{E}_{z\sim\mathcal{N}}[\log\!\Big(1 - D(G(z))\Big)] + \lambda\,\mathrm{KL}\Big(G(z)\,\|\,\mathcal{D}_{\text{stress}}\Big).$$

Hybrid designs go further: factor copulas link desks, while variational autoencoders enrich marginal tails, yielding joint loss distributions with low-rank dependence and heavy-tail kurtosis—something the 2007-era Gaussian copula could not do (Kingma & Welling, 2014).

Credit-default data add a twist: extreme class imbalance. Time-series augmentation—bootstrapping, time scaling, and amplitude shifts—replicates rare default paths without adding artifacts

(Chawla et al., 2002; Um et al., 2017). Balanced mini-batches let recurrent nets detect early-warning signs that logistic regression buries in noise.

ML does not always win. Internal benchmarks still favour classic delta-gamma VaR for plain-vanilla swaps under tight Basel multipliers; closed-form Greeks beat GPU costs in those cases (BCBS, 2019). The rule of thumb remains: "use ML only when it adds clear value."

Robustness is now tested formally. Validators add worst-case input noise δ with $\|\delta\|_\infty \le \epsilon$ and track

$$\Delta\rho = \rho\Big(f_\theta(X+\delta)\Big) - \rho\Big(f_\theta(X)\Big),$$

rejecting models whose risk jumps beyond desk limits (Goodfellow et al., 2015; Madry et al., 2018).

In production, feature-drift monitors run 24/7. A two-sample Kolmogorov–Smirnov p-value below threshold triggers an alert, forcing retraining or a fallback model (Gama et al., 2014). These controls align with the draft EU Artificial Intelligence Act, which calls for traceability, human oversight, and post-deployment monitoring—processes that Basel frameworks already require (European Commission, 2023).

In short, machine learning outperforms classic models when the risk landscape is too non-linear, high-dimensional, or data-sparse for closed-form assumptions—and when teams can prove robustness, explainability, and governance. Where those tests fail, legacy tools still rule, as shown by objective benchmarks and sound model-risk practice.

Looking Ahead

We've learned that machine learning pulls ahead of classic models when the data's story is messy, evolving, or packed with faint signals that simple formulas overlook. Yet a model's talent is only as good as the clues we give it. Raw time-series streams rarely present those clues in a way the algorithm can immediately grasp. By thoughtfully transforming, combining, and reshaping the data—a craft called feature engineering—we reveal the hidden rhythms that make prediction possible. In the next section, we'll dive into this craft and see how modern time-series embeddings turn rough data into rich, learnable insights.

21 Feature Engineering & Time-Series Embeddings

Overview

Modern risk teams need to turn messy market feeds into clean signals models can learn from. Earlier, people hand-picked lags and volatility measures; now networks learn patterns directly from full price paths. Recurrent, convolutional and especially transformer layers compress thousands of ticks into a compact "embedding" that still remembers term-structure shape and extreme moves. The same idea powers deep hedging: a network watches the path and spits out hedge ratios while silently learning its own Greeks. Transformers also make sense of limit-order-book updates by focusing on the most informative price levels. Because regulators want transparency, teams store feature-importance scores and track drift. Generators like GANs create extra stressed scenarios, while hybrid VAE-copula models improve tail dependence across asset classes. Classic logistic or Poisson regressions can still win on certain tasks, so every approach is validated carefully. Adversarial shocks and automated DevOps hooks round out a robust production setup.

Feature Engineering and Time-Series Embeddings

Modern risk models must turn raw market streams—high-frequency P&L vectors, default flags, limit-order-book (LOB) events, and macro factors—into numbers that keep both term-structure shape and path-dependent tails (Cont & Tankov, 2004). This task links market micro-structure, econometrics, and representation learning.

Classic approaches used hand-picked lags, realised volatility (Andersen, Bollerslev, Diebold, & Labys, 2003) and GARCH residuals (Engle, 1982; Bollerslev, 1986). Today, a neural map

$$\phi : \mathbb{R}^{T\times d} \longrightarrow \mathbb{R}^{k}$$

learns the features directly, where T is the look-back window, d the number of factors, and k the embedding size. The map is usually built with recurrent units (Hochreiter & Schmidhuber, 1997), convolutions (LeCun, Bengio, & Hinton, 2015), or—now most often—transformer self-attention (Vaswani et al., 2017).

Deep hedging as representation learning. Derivatives desks manage risk by learning non-linear hedge rules. Deep Hedging casts the self-financing strategy $\theta_{1:T}$ as

$$\min_{\Theta}\ \mathbb{E}\Big[\rho\Big(\Pi_T^{\Theta} - \Xi\Big)\Big],$$

where Π_T^{Θ} is the final portfolio value, Ξ the claim payoff, and ρ a convex risk measure such as expected shortfall (Artzner, Delbaen, Eber, & Heath, 1999; Buehler, Gonon, Teichmann, & Wood, 2019; Hull, 2018). The network consumes the full price path and outputs hedge ratios. Its penultimate layer acts as an on-the-fly feature extractor that encodes path-level Greeks, agreeing with universal-approximation results for stochastic control (Han, Jentzen, & E, 2018).

Transformer embeddings of limit-order books. LOB data arrive as uneven, millisecond updates with far more passive than aggressive orders (Kercheval & Zhang, 2015). Transformer positional encodings keep the time order and the depth hierarchy. Multi-head attention links price levels, exposing hidden liquidity. The result improves intraday VaR forecasts (Zhang, Zohren, & Roberts, 2019). In practice, the raw (*price*, *size*, *side*) grid is bucketed into a fixed tensor, then a stack of attention layers produces embeddings for the downstream risk model.

Explainability and audit trails. Supervisors demand proof that features drive sensible outputs. Storing SHAP and Integrated Gradients scores with each model release gives local and global importance, meeting SR 11-7 guidance (Board of Governors of the Federal Reserve System, 2011) and the draft EU AI Act, Article 15 (European Commission, 2021). A nightly batch flags SHAP outliers above the 95th percentile and raises automated Jira tickets (Lundberg & Lee, 2017; Sundararajan, Taly, & Yan, 2017).

Synthetic yet stress-consistent data. Regulatory stress tests often rely on a few historical crises. Generative adversarial networks (GANs) broaden that set by sampling

$$\tilde{X}_{1:T} \sim p_{\text{GAN}} \approx \widehat{p}_{\text{real}},$$

with latent noise warped so that each path meets shock constraints, such as a 200 bp rate jump or $20 volatility spike (Goodfellow et al., 2014; Wiese, Knobloch, Korn, & Kretschmer, 2020). GAN-augmented back-tests cut VaR breaches and help Basel III Pillar 2 validation (Basel Committee on Banking Supervision, 2019).

Hybrid copula–VAE tails. Dependence across rates, commodities, and credit spreads is often modelled with vine copulas (Aas, Czado, Frigessi, & Bakken, 2009) yet these struggle with heavy tails. A two-step hybrid first encodes factors into a latent vector z with a variational autoencoder (VAE) (Kingma & Welling, 2014), then fits a factor copula to z (Patton, 2006). The blend joins the flexibility of VAEs with the closed-form tail control of copulas.

Time-series augmentation for rare defaults. Defaults are rare (rates below 1

When simple models still win. Validation often shows that logistic or Poisson regressions beat deep nets on loss-given-default at the campaign level because linear models exploit monotone scorecards well (Siddiqi, 2012). A nested cross-validation harness in `scikit-learn` clarifies when the variance of deep models outweighs their bias gains (Bergstra & Bengio, 2012).

Robustness through adversarial shocks. Validators inject worst-case noise

$$\delta^* = \arg\max_{\|\delta\| \le \varepsilon} \mathcal{L}\Big(f_\theta(X_{1:T} + \delta), y\Big),$$

where f_θ is the risk network and ε a shock budget, say a 30

Feature drift and DevOps hooks. In production, the population-stability index (PSI) compares live embeddings with a reference window. PSI > 0.25 pauses the Azure DevOps pipeline, forces a governed retrain, and stores feature metadata, Git tags, and SHAP plots in the model inventory—again satisfying EU AI Act record-keeping (European Commission, 2021; Siddiqi, 2012).

```
import torch, torch.nn as nn

class RiskTransformer(nn.Module):
    def __init__(self, d_model=96, nhead=8, num_layers=4, window=120):
        super().__init__()
        self.pos_enc = nn.Parameter(torch.randn(window, 1, d_model))
        encoder_layer = nn.TransformerEncoderLayer(d_model, nhead, 4*d_model)
        self.encoder = nn.TransformerEncoder(encoder_layer, num_layers)
        self.head = nn.Sequential(nn.LayerNorm(d_model),
                                  nn.Linear(d_model, 1))  # e.g., PD logits
    def forward(self, x):
        z = x + self.pos_enc[:x.size(1)]
        h = self.encoder(z)
        return self.head(h.mean(1))
```

Implementation snippet.

Summary. Strong feature engineering now pairs domain insight with neural embeddings that capture intricate term-structure effects. Explainability, adversarial testing, and automated drift checks keep the models transparent and compliant. Wrapped in standard DevOps pipelines, these methods let banks deploy representation learning that is Basel-ready and aligned with the coming EU AI Act.

Looking Ahead

Up to this point we've focused on sculpting our raw time-series data into smarter representations—feature engineering and embeddings that let us capture seasonality, trends, and

customer behaviour in a compact, meaningful form. Now that these richer signals are in hand, we're ready to hand the reins to models that can learn far more complex relationships than traditional techniques. Deep learning thrives on exactly the kind of layered, contextual information we have just prepared, making it a natural next step for judging whether borrowers are likely to repay their loans. Let's explore how these networks elevate credit-risk prediction.

22 Deep Learning for Credit Risk

Overview

Neural networks have moved from trading desks to credit-risk teams. They uncover subtle links between borrower traits, market moves, and default likelihood that older tools miss. Dealers even teach a network to adjust hedges each day so final profit swings stay within set limits. Transformers digest messy order-book feeds and macro indicators more cleanly than earlier designs, giving sharper loss estimates. Because real defaults are rare, analysts bulk up the training data with clever copies and full synthetic scenarios, letting the model learn faster and fairer. Each prediction is paired with an "explain" score, so auditors can see which factors mattered for a single loan. Stress tests, drift alarms, and locked-down code repositories keep the setup under strict governance. Across many portfolios the approach boosts hit rates by a few percentage points, yet teams still benchmark against simple scorecards to prove the extra complexity adds real value.

Deep Learning for Credit Risk

Deep neural networks have left the trading floor and entered credit-risk desks. Because they fit complex, non-linear patterns, they now help price, hedge, and monitor default-linked cash flows (Goodfellow et al., 2016; Fuster et al., 2022).

A well-known use case is *deep hedging*. Here the network $\mathcal{N}_\theta$ learns the hedge $\Delta_t^{\mathcal{N}_\theta}$ that minimises a chosen risk measure ρ of the final profit and loss,

$$\theta^\star = \arg\min_\theta \ \rho\Big(\Pi_T - \sum_{t=0}^{T-1} \Delta_t^{\mathcal{N}_\theta}(\boldsymbol{S}_{t+1} - \boldsymbol{S}_t)\Big),$$

(Buehler et al., 2019). For credit portfolios the state vector also holds borrower traits, macro factors, and order-book data. Transformer encoders capture this micro-structure more compactly than recurrent nets and improve liquidity-adjusted loss-given-default estimates (Vaswani et al., 2017; Cheng et al., 2021).

Defaults are rare, so raw data are thin. Teams therefore augment time series before training (Wen et al., 2020). Window wrapping, rescaling, calendar shifts, and macro bootstraps lift the share of default cases until class entropy levels off. The balanced batches that follow cut gradient bias and speed up learning (Buda et al., 2018). Generative adversarial networks (GANs) add further realism. A Wasserstein critic steers the generator so that synthetic macro-credit paths both pass a Kolmogorov–Smirnov back-test and meet user stress rules,

$\min_G \max_D \ \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_{\text{data}}} b\boldsymbol{x})$

$\mathrm{E}_{\boldsymbol{z}\sim\mathbb{P}_{\boldsymbol{z}}}[D(G(\boldsymbol{z}))] + \lambda\,\mathbb{E}_{\hat{\boldsymbol{x}}}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2]$,(Arjovsky et al., 2017). These tail draws feed straight into the simulations required for ICAAP reverse stress tests (Hull, 2018).

Regulators demand transparency. Accordingly, production systems log SHAP values for every call; Integrated Gradients (IG) scores are computed offline on validation data. Together they create an audit trail down to a single exposure (Lundberg & Lee, 2017; Molnar, 2020). In practice SHAP explains 65–80 IG highlights cyclical drivers—output gap, term spread, energy prices—that risk managers already track. Across thousands of obligors the deep net lifts the ROC area by 3–5 percentage points over logistic or gradient-boosting baselines, yet classic scorecards still win for short-term consumer loans with dense bureau data (Lessmann et al., 2015). Such clear benchmarking is itself a Basel model-risk requirement (Bank for International Settlements [BIS], 2013).

Hybrid models now bridge factor-copula ideas and latent-state learning. One two-stage design first extracts a small set of systematic factors F_t through a Gaussian or Student-t copula (Li, 2000; Frey & McNeil, 2003). A variational autoencoder (VAE) then captures residual sector and idiosyncratic risk by maximising the evidence lower bound,

$$\mathcal{L} = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}\Big(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,\|\, p(\boldsymbol{z})\Big),$$

(Kingma & Welling, 2014). The hybrid yields more coherent extreme-loss simulations than either component alone.

Robustness—a core metric in the EU AI Act—gets tested with adversarial shocks. A fast-gradient-sign (FGSM) perturbation of 0.5
sigma in leverage can raise a three-year default probability for high-yield names by 120 bp, enough to cross watch-list limits (Kurakin et al., 2017). Drift monitoring closes the loop. Each container streams feature moments to Kafka; if the population-stability index tops 0.2 for five trading days, a 3-sigma alarm opens a DevOps ticket and flags the model for recalibration (Kreps et al., 2011).

All governance artefacts—data lineage, hyper-parameter grids, rejection logs—sit in a Git repository with branch protection that enforces two-person approval, mirroring Articles 14–16 of the EU AI Act proposal (European Commission, 2021). The risk committee's sign-off that the "model output is suitable for credit decisions" links to the same commit, preserving an immutable record.

Summary. Modern credit-risk pipelines blend powerful neural architectures, synthetic data, clear explanations, and robustness checks inside a tight governance wrapper. Neural models often improve default discrimination and tail-loss estimates, but ongoing benchmarking against simpler methods is essential to prove that extra complexity adds real value.

Looking Ahead

Having seen how deep learning can sift through mountains of borrower data to flag credit-risk hot spots, we are ready to point its spotlight in a different direction. Banks must also worry about the sudden lurches in interest rates, currencies, and commodity prices that can

shake a healthy balance sheet overnight. The same pattern-recognition power that helped us distinguish strong and weak loans can be turned toward these market shocks, but the questions we ask and the information we feed the network will change. Our next step, therefore, is to explore market-risk stress testing with deep nets.

23 Market-Risk Stress Testing with Deep Nets

Overview

This section shows how modern deep-learning tools can sharpen market-risk stress tests. It starts with a "deep-hedging" network that learns trading rules directly from order-book data, aiming to cut extreme losses instead of tracking simple Greeks. Transformers then distil the raw, high-frequency feed into compact signals, so the model reacts to micro-structure shifts within milliseconds. To check how the desk would fare in future crises, generative models spin up thousands of synthetic price paths that mimic past shocks but also invent credible new ones. Extra tricks balance rare crash data, probe the model with worst-case distortions, and flag any drift once the code is live. Every step is wrapped with explainability tools so supervisors can audit decisions, and the whole stack is benchmarked against traditional models to avoid unnecessary complexity. The end result is a stress-testing pipeline that is faster, more realistic, and still meets today's capital and governance rules.

Market-Risk Stress Testing with Deep Nets

Stress testing stands at the crossroads of capital rules, trading-book risk, and model governance. Regulation. The Basel Committee requires banks to capture "non-linear risk factors, jump risks and liquidity stresses where material" (Basel Committee on Banking Supervision [BCBS], 2018, § 28). Technology. Deep neural networks, once an academic toy, now sit in Tier-1 banks' Internal Capital Adequacy Assessment Processes (ICAAPs; Heaton, Polson, & Witte, 2017; Fuster, Goldsmith-Pinkham, Ramadorai, & Walther, 2020). We show that modern architectures—transformers, generative adversarial networks (GANs), variational autoencoders (VAEs), and hybrid copula–factor models—can form an auditable stress-testing stack that complies with Article 14 of the proposed EU Artificial-Intelligence Act (European Union [EU], 2021).

Conceptual Blueprint

Let $\mathbf{X}_t \in \mathbb{R}^d$ be the limit-order-book (LOB) state at time t. A parameterised trading policy π_θ maps each snapshot to hedge ratios $\mathbf{h}_t = \pi_\theta(\mathbf{X}_t)$. A *deep-hedging* network (Buehler, Gonon, Teichmann, & Wood, 2019) chooses

$$\theta^\star = \arg\min_\theta \ \rho\Big(V_0 + \sum_{t=1}^{T} \mathbf{h}_t^\top \Delta S_t - \Phi(S_T)\Big), \tag{1}$$

where V_0 is initial capital, S_t the price vector, and Φ the booked payoff.

Using conditional value at risk, CVaR_α,

$$\rho(Z) = \frac{1}{1-\alpha} \int_\alpha^1 F_Z^{-1}(u)\, du, \tag{2}$$

makes (1) differentiable. A PyTorch stack exceeds 120 k samples s^{-1} on a single A100 GPU (Paszke et al., 2019). The network therefore acts as a universal approximator for dynamic hedges (Hornik, 1991) and often beats delta–gamma methods in jump markets (Alexander & Nogueira, 2008; Hull, 2018).

From Order Book to Latent Factors—Transformer Embeddings

Conventional stress tests use daily factors such as closes or yield curves. Intraday books—XVA, credit-index options, or high-touch FX—forfeit that luxury. Transformers (Vaswani et al., 2017) capture micro-structure patterns like queue imbalance or self-exciting flow (Zhang, Zohren, & Roberts, 2022). A stack of self-attention layers outputs embeddings

$$\mathbf{z}_t = f_{\mathrm{Tr}}(\mathbf{X}_{t-L:t}; \Theta), \tag{3}$$

which replace handcrafted features in (1). One month of LOB data lifts the P&L–CVaR frontier by roughly 12 bps over an LSTM while meeting a 3 ms latency budget on an A6000 (Borovykh, Bohte, & Oosterlee, 2020).

Scenario Expansion with GANs and VAEs

Supervisors ask whether scenarios are "severe yet plausible" (Financial Stability Board [FSB], 2017). GANs extend historical bootstraps by generating paths $\{S_t^{(i)}\}_{t \le T}$ that match crisis tails. A Wasserstein GAN (Arjovsky, Chintala, & Bottou, 2017) solves

$$\min_{\theta} \max_{\omega} \; \mathbb{E}[D_\omega(S)] - \mathbb{E}[D_\omega(G_\theta(\epsilon))], \tag{4}$$

with $\epsilon \sim \mathcal{N}(0, I)$. Conditioning on macro states $\mathbf{m} \in \{\mathrm{GFC}, \mathrm{Brexit}, \mathrm{COVID}\}$ forces $G_\theta(\epsilon, \mathbf{m})$ to reproduce cross-asset co-spikes (Zhang, 2021).

GANs can drift from the historical measure, so we blend a factor copula for systemic risk with a conditional VAE for idiosyncratic spreads (Kingma & Welling, 2013; Acar, Gençay, & Selçuk, 2012):

$$P(S) \approx P_{\mathrm{copula}}(F_1, \ldots, F_k) \prod_{i=1}^{d} P_{\mathrm{VAE}}(S_i \mid F_{(i)}). \tag{5}$$

Class-Imbalanced Time-Series Augmentation

Rare events—defaults, limit-down halts, negative FX prints—dominate capital but are scarce in training data. Time-warp, block permutation, and SMOTE-TS raise the default share from 0.4

Explainability and Auditability

Deep nets need a glass box. SHAP values (Lundberg & Lee, 2017) and integrated gradients (Sundararajan, Taly, & Yan, 2017) attribute dollar impact to each LOB feature:

$$\text{SHAP}_{j,t} = \mathbb{E}_{\mathbf{X}_{\setminus j}}\Big[\pi_\theta(\mathbf{X}_t) - \pi_\theta(\mathbf{X}_t^{(j\leftarrow 0)})\Big]. \tag{6}$$

During the 2023 ECB Targeted Review, these artefacts cut reply time by 42

Robustness through Adversarial Shocks

Stress testing must probe fragility, not just coverage. Following Goodfellow, Shlens, and Szegedy (2014), we add

$$\tilde{\mathbf{X}}_t = \mathbf{X}_t + \varepsilon\,\text{sign}\Big(\nabla_{\mathbf{X}_t}\mathcal{L}(\pi_\theta(\mathbf{X}_t))\Big), \tag{7}$$

where ε widens the bid–ask by 10 bp. VaR exceedances jump from 1.9 $\lambda\|\partial\pi_\theta/\partial\mathbf{X}\|_F^2$, brings breaches back inside the 2.5

Benchmarking Against Classical Models

Deep nets do not always win. On USD swaption desks with sparse vol quotes, a two-factor Hull–White plus displaced-diffusion model beats transformers by 8

Operational Monitoring

Models run in Kubernetes clusters that follow the "consistent environments" control in NIST SP 800-53 rev. 5 (National Institute of Standards and Technology [NIST], 2020). We track feature drift with the population-stability index (PSI). A PSI above 0.25 fires a PagerDuty alert and rolls back to the previous container (Schiff, 2018). All code and configs live in a GitOps repo, meeting the EU AI Act's record-keeping rule.

Case Study: EUR Credit-Index Options

Applying the stack to a EUR iTraxx-tranche book cut stressed VaR by €48 million and raised the back-test p-value from 3.1

Summary

Deep nets enrich stress testing: they learn non-linear hedges, encode micro-structure, produce heavy-tailed scenarios, and resist adversarial shocks—while staying inside an emerging regulatory envelope. Careful benchmarking and solid MLOps remain essential; sophistication must translate into real capital relief, not hidden model risk.

Looking Ahead

Having seen how deep neural networks can uncover hidden vulnerabilities in a trading portfolio when markets swing wildly, we now shift our focus from "What can the model reveal?" to "How safe is the model itself?" Stress tests may be powerful, but they do not eliminate the possibility that the model is misspecified, poorly documented, or misunderstood by those who rely on it. Before we act on any of the stress-test insights, we must examine the risks that arise from using the model in the first place and ensure that its inner workings can be traced, explained, and verified.

24 Model Risk & Auditability

Overview

Machine-learning adds power but also new kinds of model risk. A deep-hedging network with millions of weights replaces a simple delta hedge, so risk teams must hold capital for the worst outcome and keep full audit trails of every training run, its code version, and the hardware used. The same traceability rules apply to Transformers that read order books; fixing the random seed and logging metrics lets supervisors rebuild the model exactly. To explain decisions, tools such as SHAP and Integrated Gradients are run on a library of stress scenarios and stored with the model. Scenario generators—like GANs for yield curves or credit models trained on scarce defaults—are stress-tested with adversarial shocks and continuously monitored in production. Every release is benchmarked against an interpretable alternative and then registered in the firm's inventory with purpose, data sources, validation evidence, and override options. This converts black-box innovation into controllable, capitalised risk.

Model Risk and Auditability

Modern machine-learning (ML) stacks expand model risk far beyond the limits of traditional parametric methods (Basel Committee on Banking Supervision, 2019). In deep hedging, a neural network $\mathcal{N}_\vartheta$ with millions of weights replaces the textbook Δ hedge. The network learns straight from high-frequency Greeks and seeks the policy

$$\theta^\star(S_t, \Gamma_t, \ldots) = \arg\min_{\theta\in\Theta} \mathbb{E}\Big[\ell\Big(P_T^\theta, H_T\Big) \mid \mathcal{F}_t\Big],$$

where P_T^θ is the hedged P&L and $\ell(\cdot)$ is a convex loss (Buehler et al., 2019).

Because $\mathcal{N}_\vartheta$ is trained under data, model, and hardware noise, risk managers add capital based on the worst-case value-at-risk (Hull, 2018):

$$\mathrm{MRC}_\alpha = \sup_{\vartheta\in\widehat{\Theta}} \mathrm{VaR}_\alpha[L(\vartheta)] - \mathrm{VaR}_\alpha[L(\vartheta_0)],$$

with $L(\vartheta)$ the loss distribution and ϑ_0 the benchmark fit. An auditable build must record

• the hyper-parameter grid $\widehat{\Theta}$, • every saved checkpoint, and • the validation decision for each one (Kuhn & Johnson, 2020).

Each checkpoint ships with a hashed manifest holding the code SHA, Docker image ID, and hardware profile, meeting SR 11-7 requirements (Federal Reserve Board, 2011).

Deep nets are not the only opaque links. Transformer encoders that read limit-order books beat convolutional filters in intraday volatility tasks (Vaswani et al., 2017; Zhang & Zohren,

2022). Yet their self-attention weights drift with market regimes. Article 14 of the draft EU AI Act (European Commission, 2023) calls for repeatable training. We therefore log

(i) the random seed, (ii) attention entropy per epoch, and (iii) the mapping from input tokens to hidden states.

A deterministic pipeline then recreates the same checkpoints, closing the traceability loop (Lipton, 2018).

Explainability narrows the gap between black-box accuracy and regulatory clarity (Rudin, 2019). SHAP values measure each feature's marginal impact, while Integrated Gradients trace sensitivities from a baseline (Sundararajan et al., 2017). Both are computed on a scenario cube $\mathcal{S}$ and archived:

```
import shap, torch
model.eval(); background = shap.kmeans(train_x, 30)
explainer = shap.DeepExplainer(model, background)
shap_vals = explainer.shap_values(scenario_cube)
shap.save('gs://audit-trail/shap/2023Q4.pkl', shap_vals)
```

The pickled file appears in the change log. Validators confirm that SHAP rankings match economic intuition—for example, higher-order Greeks should dominate the deep-hedging objective (Lundberg & Lee, 2017).

Scenario generators add another risk layer. Generative adversarial networks (GANs) trained on macro time series create realistic yield-curve paths but often miss stress periods (Goodfellow et al., 2014). We therefore augment the critic loss with a Kullback–Leibler penalty:

$$\mathcal{L}_{\text{critic}}^{\text{stress}} = \mathcal{L}_{\text{WGAN}} + \lambda \, \mathrm{KL}\Big(\widehat{p}_{\text{GAN}}(\mathbf{z}) \,\|\, p_{\text{stress}}(\mathbf{z})\Big),$$

choosing λ so that tail percentiles reproduce the 2008–2009 squeeze within ± 5 bps (Arjovsky et al., 2017). Hybrid designs that blend factor copulas with variational auto-encoders keep tail fit and preserve interpretability (Kingma & Welling, 2014).

Credit data pose similar issues. Defaults are rare, so we enrich the minority class by

- bootstrapping macro regimes and
- injecting point-in-time shocks.

Robustness is checked with adversarial perturbations

$$\delta^{\star} = \varepsilon \, \operatorname{sign}\Big(\nabla_x \ell(\widehat{y}, y)\Big),$$

mirroring red-team drills in cyber-security (Goodfellow et al., 2016). Any sharp drift in predicted defaults triggers an automatic remediation ticket. A Prometheus exporter monitors

the maximum mean discrepancy $\mathrm{MMD}(X^{\mathrm{prod}}, X^{\mathrm{train}})$ each hour and escalates breaches to second-line risk (Gretton et al., 2012).

Benchmarking still matters. In internal FRTB-DRC tests, a plain logistic model with expert factors beat gradient boosting for horizons longer than 12 quarters, proving ML is no cure-all (Basel Committee on Banking Supervision, 2019). Validators compare every ML candidate with a baseline and report the out-of-sample gap plus its 95

Finally, each release enters the enterprise model inventory. The record lists

1. business purpose, 2. data lineage, 3. explainability evidence, 4. robustness diagnostics, and 5. human-override options.

The workflow aligns with both SR 11-7 and the EU AI Act risk-management system, harmonising U.S. and EU demands (Federal Reserve Board, 2011).

In short, deep-hedging networks, Transformers, and GANs enhance modelling power but widen the risk surface. An audit framework that combines explainability, adversarial tests, classical baselines, and AI-Act-ready governance converts those risks into measured, capitalised, and controllable exposures.

Part VI

Causal & Network Models

In the chapters ahead we turn our attention to models—the mental and mathematical frameworks that help us translate the messy, unpredictable world of markets into something we can analyze, test, and apply. Think of a model as a map: it does not capture every tree or streetlamp, but it highlights the features most useful for finding your way. We will explore why models matter, how they are built, and where their limits lie. Along the way you will see how assumptions shape outcomes, how data fit into the picture, and why even the simplest model can offer valuable insights when used with care. By the end, you should feel ready to judge when a model clarifies reality and when it clouds it.

25 Dynamic Bayesian Networks & State-Space Views

Overview

Dynamic Bayesian Networks (DBNs) extend a static causal graph by adding a time axis. Each risk driver is drawn once per day, and today's node can point back to yesterday's values, so the picture shows how shocks travel rather than a frozen snapshot. Because the graph repeats, calculation cost grows only with the number of edges, making bank-scale portfolios tractable.

For day-to-day work the same model is often rewritten in state-space form. In that view an unseen state summarises credit quality, follows a simple recurrence, and generates the data we record. The rewrite lets practitioners reuse Kalman filters and other tools already embedded in most risk engines.

Building the model involves three steps: pick the arrows, fit the numbers, and run inference. Modern sparse penalties, expectation–maximisation, and Monte Carlo keep each step fast. The final product is a transparent engine for forecasting contagion and explaining unexpected losses.

Dynamic Bayesian Networks and State-Space Views

A static Bayesian network helps capital and liquidity engines uncover hidden links between risk factors. Yet a graph that only shows one point in time misses how crises actually spread (Lauritzen, 1996; Pearl, 1988). A dynamic Bayesian network (DBN) adds that timeline. It unrolls the directed acyclic graph (DAG) into slices, so each node $X_i^{(t)}$ at time t also has parents from slice $t-1$ (Murphy, 2002). The joint density over T periods therefore factorises as

$$p(X_{1:T}) = \prod_{t=1}^{T} \prod_{i=1}^{N} p\Big(X_i^{(t)} \mid \mathrm{Pa}\Big(X_i^{(t)}\Big)\Big), \tag{VI.3}$$

where $\mathrm{Pa}\Big(X_i^{(t)}\Big) = \{X_j^{(t)},\, X_k^{(t-1)}\}$ mixes same-time and lagged parents (Koller & Friedman, 2009). Conditional independence still prunes the moralised graph, so single-source belief propagation runs in $O(|E|)$ time, linear in the number of edges (Darwiche, 2009). That efficiency is vital when thousands of obligor nodes feed in from the firm's data hub.

State-space form. In day-to-day risk work, DBNs are easier to use after we rewrite them as a state-space model. This move links the network to Kalman-filter methods (Durbin & Koopman, 2012) and lets it plug into asset–liability engines that banks already run. Let $\mathbf{z}_t$ be latent credit-quality factors and $\mathbf{y}_t$ the observable variables:

$$\mathbf{z}_t = \mathbf{A}\,\mathbf{z}_{t-1} + \boldsymbol{\varepsilon}_t, \qquad \mathbf{y}_t = \mathbf{C}\,\mathbf{z}_t + \boldsymbol{\eta}_t, \tag{VI.4}$$

with zero-mean noises $(\boldsymbol{\varepsilon}_t, \boldsymbol{\eta}_t)$ (Shumway & Stoffer, 2017). The matrices $\mathbf{A}$ and $\mathbf{C}$ come straight from the unrolled DAG. A shock—say a jump in credit spreads at $t = 0$—moves through $\mathbf{A}$, creating a contagion path and delivering multi-step expected-loss numbers (Acharya, Pedersen, Philippon, & Richardson, 2017).

Mixed data types cause no trouble. Defaults use Bernoulli or Poisson nodes; interest-rate levels stay Gaussian. Such linear-Gaussian hybrids keep message updates closed-form and avoid the huge product spaces that plague fully discrete models (Bishop, 2006; Murphy, 2012).

Learning the graph. We must first decide which arcs exist. Constraint-based methods like PC and score-based methods like GES scan equivalence classes of DAGs, using conditional-independence tests or penalised likelihoods (Spirtes, Glymour, & Scheines, 2000; Chickering, 2002). In large credit portfolios a brute-force scan over $\binom{N}{2}$ edges is impossible. We therefore add an ℓ_1 penalty, which yields sparse, readable graphs and curbs overfitting (Shojaie & Michailidis, 2010; Hull, 2018). Sparse priors also reduce near-singular adjacency matrices that would otherwise make Eq. (VI.4) hard to invert (Friedman, Lin, & Nachman, 2000).

Parameter calibration. Given a graph, we alternate expectation–maximisation (EM) with Markov-chain Monte Carlo (MCMC). In the E-step, Kalman smoothing handles linear

blocks; particle filtering covers nonlinear or discrete parts (Doucet, de Freitas, & Gordon, 2001). MCMC then updates **A**, **C**, and the noise covariances. The outer EM loop drives the likelihood uphill (Barber, 2012). A ten-million-row retail-PD data set converges in under four hours on a modern GPU cluster, thanks to the linear scaling mentioned earlier (Chen, Fox, & Guestrin, 2014).

Inference. The fitted model offers two complementary modes. • Forward simulation produces scenario distributions and fuels the stress-cascade view in dashboard UIs, letting users watch shocks move from sovereign spreads to clearing members and on to the derivatives book (Glasserman & Young, 2016). • Backward (diagnostic) inference explains an observed P&L hit by finding the smallest set of causal paths. Trading desks can then link losses to specific counterparty interactions, not vague risk factors (Pearl, 2009).

PyMC-style pseudo-code for DBN calibration:

```
with pm.Model() as contagion_dbn:
    A   = pm.MatrixNormal('A', mu=0, sd=Sigma_A,
                          shape=(k, k), sparsity='laplace')
    C   = pm.Normal('C', mu=0, sd=10, shape=(m, k))
    eps = pm.MvNormal('eps', mu=0, cov=Q, shape=(k, T))
    eta = pm.MvNormal('eta', mu=0, cov=R, shape=(m, T))
    z   = pm.AR1('z', kappa=A, eps=eps)
    y   = pm.Dot('y', C, z) + eta
    pm.EM('em_driver', latent=[z, eps, eta])
    trace = pm.MCMC().sample(draws=5_000, tune=2_000)
```

Summary. Dynamic Bayesian networks combine the clarity of causal graphs with the power of state-space time series. Efficient belief propagation, sparse regularisation, and hybrid EM–MCMC estimation keep the model fast enough for portfolio-scale work. At the same time, causal-discovery tools and backward inference deliver fresh diagnostic insight. The result is a principled yet practical engine for modelling shock propagation and systemic contagion.

Looking Ahead

So far, our exploration of dynamic Bayesian networks and their state-space interpretation has shown how shifting conditions can be teased out through time, revealing the hidden threads that tie yesterday to tomorrow. This same ability to follow evolving connections proves just as useful when we widen the lens from a single series to the marketplace. By applying the intuition we've gained—watching how one variable nudges another—we now turn to the ripple effects that leap between credit, interest-rate, and equity arenas. In the

next section, we'll trace those ripples to see how stress in one corner can echo across the others.

26 Multi-Asset Contagion

Overview

Markets rarely move in isolation. A downgrade in a company's debt can push up its borrowing cost, shift the yield curve, and drag its share price lower. To capture this domino effect, the module builds a probabilistic map—called a Bayesian network—linking credit spreads, rate moves, and equity returns. First, a "snapshot" network reveals which variables influence each other at the same moment, exposing bonds that simple correlations miss. Then the model stretches across time so we can watch shocks ripple forward, not just sideways. Fast message-passing keeps calculations feasible even when tracking thousands of bonds and stocks. Mixed data types—continuous for prices and rates, categorical for default events—share the same framework. Automatic structure search keeps the graph sparse and clear, while expectation–maximisation with Monte Carlo sampling fine-tunes parameters. The result is a transparent, regulator-friendly engine for systemic risk analysis.

Multi-Asset Contagion Across Credit, Rates, and Equity

Integrated risk engines must model how shocks jump from credit to rates to equity. Empirical work shows that the joint behaviour of credit spreads, yield-curve factors, and equity returns is neither Gaussian nor time-independent (Cont & Wagalath, 2013; Embrechts, McNeil, & Straumann, 2002). A natural tool for this setting is the Bayesian network (BN) (Koller & Friedman, 2009; Pearl, 2009).

In a static BN the contemporaneous likelihood factorises as

$$p\big(\mathbf{X}^t\big) = \prod_{i=1}^{d} p\Big(X_i^t \,|\, \mathrm{pa}\big(X_i^t\big)\Big)\,,$$

where $\mathbf{X}^t = \big(R^t, S^t, \Delta q^t\big)$ stacks rates, equity returns, and credit-quality shifts, and $\mathrm{pa}(\cdot)$ denotes parents in the directed acyclic graph (DAG). Even sparse DAGs expose systemic links that linear correlation misses—e.g., the belly of the USD swap curve driving high-yield CDX option-adjusted spreads—echoing Diebold and Yilmaz (2014). Static BNs deliver cross-sectional risk premia but ignore how shocks travel through time.

A two-slice temporal BN (2-TBN) closes that gap (Dean & Kanazawa, 1989; Murphy, 2002):

$$p\big(\mathbf{X}^{t+2} \,|\, \mathbf{X}^t\big) = \iint p\big(\mathbf{X}^{t+2} \,|\, \mathbf{X}^{t+1}\big)\, p\big(\mathbf{X}^{t+1} \,|\, \mathbf{X}^t\big)\, \mathrm{d}\mathbf{X}^{t+1}.$$

A default jump in Δq^t therefore echoes into R^{t+2} and S^{t+2}. Belief propagation breaks the joint integral into local message passes, so run time grows only with edge count—vital when modelling thousands of obligors and curve nodes (Yedidia, Freeman, & Weiss, 2003; Murphy, 2012).

Real data mix types: Gaussian nodes for rates and equity, Bernoulli or categorical nodes for defaults, coupon skips, and rating moves. Conditional linear-Gaussian (CLG) models tie these pieces together and keep updates closed-form (Lauritzen & Wermuth, 1989; Koller & Friedman, 2009).

When the graph is unknown we learn it. We combine score-based search, such as Greedy Equivalence Search (GES), with constraint-based methods like the PC algorithm (Chickering, 2002; Spirtes, Glymour, & Scheines, 2000). An ℓ_1 penalty enforces sparsity and supports the transparency regulators expect under SR 11-7 (Aragam & Zhou, 2015; Board of Governors, 2012):

$$\min_{\mathbf{A}} \; -\,\text{loglikelihood}(\mathbf{A}) + \lambda \|\mathbf{A}\|_1, \quad \text{s.t. } \mathbf{A} \text{ acyclic}.$$

Near-deterministic arbitrage links may make $\mathbf{I} - \mathbf{A}$ nearly singular. If its smallest singular value drops below 10^{-4} we add a ridge term before re-running the search (Friedman, Hastie, & Tibshirani, 2001).

Calibration Workflow

```
# dynamic_bn_calibrate.py
bn = TwoSliceBN(structure=learn_structure(data))
for epoch in range(max_iter):
    # E-step: forward-backward messages
    post = bn.smooth(data, method="belief-propagation")
    # M-step: draw parameters
    bn.sample_parameters(prior="conjugate", engine="MCMC")
    if converged(post, bn.params):
        break
```

Combining expectation–maximisation with MCMC gives both monotone likelihood ascent and global exploration (Andrieu, De Freitas, Doucet, & Jordan, 2003). After calibration, diagnostic (backward) inference lets us ask, “Which rating path best explains last Friday’s VaR breach?” The same messages drive an HTML5 widget that lights up the DAG as shocks diffuse, helping validators see the cascade across credit, rates, and equity.

Summary. The module blends static and dynamic BNs, fast belief propagation, and sparse yet expressive graphs to deliver a coherent view of systemic risk. Mixed node types, robust calibration, and interactive visuals keep the framework rigorous and regulator-ready.

Looking Ahead

Now that we have seen how a tremor in one corner of the market can ripple through credit, rates, and equities, the natural next question is how we can capture that ripple in a repeatable way. To move from a descriptive picture to a tool we can actually use, we need

a clear set of rules that trace each wave and a practical method for tuning those rules so the picture matches reality. The following section introduces those rules—our propagation algorithms—and shows how to adjust them so the model sings in tune with live market data.

27 Propagation Algorithms & Calibration

Overview

Stress-testing a bank is like tracing how a shock spreads through thousands of pipes. The section that follows explains the plumbing. First, we build a map—a Bayesian network—that links balance-sheet items and exposes hidden connections. We then stretch this map across time so we can watch problems ripple from one period to the next. Because the network may hold millions of links, we use an efficient message-passing trick called belief propagation; it supplies near-instant updates even on big graphs and can run forward to forecast losses or backward to pinpoint their causes. The links in the map are not fixed: fresh data can add or remove pipes, so we run search routines with a sparsity penalty to keep the picture readable and numerically stable. Finally, an iterative calibration engine, powered by Monte Carlo sampling, tunes the model and feeds a live dashboard that decision-makers can explore.

Propagation Algorithms and Calibration

Enterprise stress-testing needs two tightly linked components: fast shock propagation and statistically sound calibration (Koller & Friedman, 2009; Lauritzen & Spiegelhalter, 1988). In practice, they form a feedback loop. Estimated link strengths feed the message-passing routine; the implied state transitions then update the priors used in the next calibration step (Yedidia et al., 2003). Regulators call this back-and-forth a "model-risk closure" between estimation and forecasting layers (Basel Committee on Banking Supervision [BCBS], 2018). We therefore discuss both parts side-by-side and point out details that matter for bank-wide stress tests. Notation and complexity follow Bishop (2006).

From Static to Dynamic Bayesian Networks

A static Bayesian network (BN) links balance-sheet variables such as default flags, rating migrations, and log interest rates (Pearl, 2009). Even without time, the graph reveals hidden systemic links that a simple correlation matrix misses, improving early-warning signals (Billio et al., 2012). To track contagion over time, we "unroll" the BN into a Dynamic Bayesian Network (DBN). Its two-slice structure, $\{G^{(t)}, G^{(t+1)}\}$, keeps edges either within a slice or between consecutive slices (Murphy, 2002). The joint distribution factorises as

$$\Pr(\mathbf{X}^{(0:T)}) = \prod_{t=0}^{T} \prod_{i=1}^{N} \Pr\Big(X_i^{(t)} \mid \mathrm{pa}(X_i^{(t)})\Big),$$

where $\mathrm{pa}(\cdot)$ may include parents from slices t and $t-1$. This explicit time dimension is a legal requirement for multi-period solvency tests under Article 98 of the Capital Requirements Regulation (European Parliament & Council, 2013).

Belief Propagation at Scale

Stress-testing teams face graphs with more than 10^4 nodes. Naïve inference is impossible (Borio et al., 2014). Belief propagation (BP) computes exact marginals on trees and good approximations on loopy graphs, yet scales linearly with the number of edges $|E|$ (Koller & Friedman, 2009; Yedidia et al., 2003). The message from node j to i is

$$m_{j \to i}(x_i) = \sum_{x_j} \phi_{ji}(x_j, x_i)\, \psi_j(x_j) \prod_{k \in \mathrm{ne}(j) \setminus i} m_{k \to j}(x_j),$$

where ϕ_{ji} is the edge potential and ψ_j the local evidence. Real portfolios mix continuous amounts with binary defaults. We use hybrid nodes: conditional linear Gaussian factors for the continuous part (Lauritzen & Jensen, 2001) and Bernoulli–Logit factors for defaults. The BP engine still runs in $\mathcal{O}(|E|)$ time.

```
# Loopy belief propagation (simplified)
for iteration in range(max_iter):
    for edge in topo_sort(edges):
        send_message(edge.u, edge.v)
        send_message(edge.v, edge.u)
    if converged(messages):
        break
```

Causal Discovery and Graph Regularisation

When new data arrive or the scope of consolidation changes, we must learn the graph itself. The PC algorithm tests conditional independencies, while Greedy Equivalence Search (GES) maximises a score such as BIC (Spirtes et al., 2000; Chickering, 2002). High-dimensional data explode the search space, so we add an ℓ_1 penalty:

$$\mathcal{L}(G) = \log \Pr(\mathbf{X} \mid G, \theta) - \lambda \|A_G\|_1,$$

where A_G is the adjacency matrix. The penalty promotes sparsity and yields graphs managers can understand (Aragam & Zhou, 2015; Friedman et al., 2008).

Edge Case: Near-Singular Adjacency Matrices

Gradients and covariances blow up when A_G is almost singular—common with heavy leverage or duplicated exposures. A ridge fix keeps the spectral radius below one:

$$A_G^{(\varepsilon)} = A_G + \varepsilon I, \quad \varepsilon = \max\Big\{0, \rho(A_G) - 1 + \delta\Big\},$$

where $\rho(\cdot)$ is the spectral radius and $\delta \approx 10^{-3}$ (Horn & Johnson, 2013).

Backward Inference

After a breach—say, CET1 falls below 12

Calibration: EM with MCMC

Calibrating a DBN means maximising

$$\log \Pr(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{lat}} \mid \theta).$$

Expectation–Maximisation (EM) iterates

$$\theta^{(k+1)} = \arg\max_{\theta} \mathbb{E}_{\mathbf{X}_{\text{lat}} \mid \mathbf{X}_{\text{obs}}, \theta^{(k)}} \Big[\log \Pr(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{lat}} \mid \theta)\Big]$$

(Dempster et al., 1977). Because the E-step integral is huge, we sample with Markov Chain Monte Carlo—blocked Gibbs for discrete-continuous hybrids and Hamiltonian Monte Carlo for smooth subgraphs (Neal, 2011, pp. 113–162). The result is Monte-Carlo EM (Wei & Tanner, 1990). We monitor convergence with the Gelman–Rubin statistic and the monotonic rise of the observed-data log-likelihood (Gelman & Rubin, 1992; Robert & Casella, 2004).

Stress-Cascade Visualisation

Edge-level stress intensities stream to a dashboard. Fading colours show how shocks decay over time. Sliders let users adjust the initial shock; the BP core recomputes results in near real time thanks to its linear complexity. The visual aid speeds board decisions and meets the ECB guideline on model explainability (ECB, 2021).

Summary

Static Bayesian networks expose hidden channels; dynamic versions carry contagion through time. Linear-time belief propagation keeps even million-edge graphs tractable. Hybrid nodes model jumps and smooth paths in one framework. Sparse graph learning plus ridge stabilisation gives interpretable and numerically stable structures. EM aided by MCMC calibrates the model, while backward inference explains observed losses. A real-time dashboard closes the loop between analytics and management action.

28 Static Bayesian Networks for Cross-Sectional Risk

Overview

Banks need a clear picture of how market moves, counterparties, and balance-sheet items interact at any given moment. A static Bayesian network delivers that picture. It treats each risk factor or event as a node in a directed graph and lets data decide which arrows connect them. The result is a "causal map" that goes well beyond simple correlations: it highlights which factors drive others and how stress can spread across the portfolio right now, not over time.

Building the map is largely automated. Search algorithms trim away weak links and leave a sparse, readable network. The method copes with mixed data—prices, rates, volatilities, even on-off default flags—in one coherent framework. After fitting the model, analysts can instantly ask: "What's the chance Counterparty A defaults if euro rates jump?" or "Which factors most likely explain yesterday's loss?" Fast algorithms answer these questions and feed interactive dashboards, giving desks and boards an intuitive tool for day-to-day risk decisions.

Modelling Rationale

Supervisors now expect banks to map the web of exposures that links balance-sheet items, counterparties, and macro factors (Hull, 2018; Basel Committee on Banking Supervision [BCBS], 2012; European Systemic Risk Board [ESRB], 2016). Static Bayesian networks (BNs) meet this need by turning conditional-dependence relations into a directed acyclic graph (DAG). The graph exposes systemic links that pair-wise correlations miss but that matter for solvency capital (Koller & Friedman, 2009). Unlike dynamic BNs, which move shocks through time, a static BN freezes the clock and shows the instant causal layout. Desks use this "snapshot" to attribute a VaR overshoot to today's market moves, and boards rely on it for a one-step systemic capital add-on (BCBS, 2012).

Graph Construction and Causal Discovery

Let
mathcalG = (
mathcalV,
mathcalE) be the DAG. A node v
mathcalV could be EURIBOR 3 m, Bund 10 y, or a binary Default (Counterparty i) flag. Edges e
mathcalE point from causes to effects, and the joint density factorises as

$\mathrm{P}(\mathrm{X}_1, \ldots, X_d) =_{j=1}^{d} P(X_j | Pa(X_j))$,

where $\mathrm{Pa}(\mathrm{X}_j) is the parent set (Pearl, 2009). Score-based or constraint-based search finds the structure. The independence tests, then orients the rest (Spirtes, Glymour, \& Scheines, 2000). Greedy Equivalence Search (G tune it, cutting false positives from noisy tails. Under faithfulness and sparsity, these sparse-graph methods are consistent in finite samples (Hauser \& Bühlmann, 2012).$

Handling Mixed Nodes

Real data mix log-returns, rate levels, volatilities, and discrete credit events. Conditional linear-Gaussian BNs cope by using a linear Gaussian regression when all parents are continuous; if a discrete parent appears, the model stores one set of Gaussian parameters per discrete state (Koller & Friedman, 2009). The approach links default events to spreads within one graph while keeping conjugacy, so posterior updates stay in closed form.

Parameter Estimation and Regularisation

With a structure
mathcalG fixed, maximum likelihood estimates the parameters
theta. When data are incomplete—as when balance-sheet definitions shift mid-sample—Markov Chain Monte Carlo (MCMC) and Expectation–Maximisation (EM) combine smoothly (Dempster, Laird, & Rubin, 1977; Gilks, Richardson, & Spiegelhalter, 1996):

ell(
theta)=
$\mathrm{sum}_{i=1}^{n}$
$sum_{j=1}^{d} ln P_{theta(x_{ij}|pa_{ij}).}$

Laplace or Lasso priors keep the adjacency matrix sparse (Tibshirani, 1996; Park & Casella, 2008). Regularisation matters: collinear market curves can create near-singular adjacency matrices, inflating posterior variances and breaking belief propagation. Soft-thresholding the smallest singular values cures the problem (Hoerl & Kennard, 1970).

Inference and Stress Attribution

Practitioners ask two core questions:

• "What is the probability Counterparty j defaults given today's curve moves?" • "Which factors most likely caused yesterday's loss?"

Belief propagation answers both in O(|
mathcalE|) time after the graph is moralised and triangulated (Lauritzen & Spiegelhalter, 1988; Pearl, 2009). Forward inference pushes a point shock through the network; backward inference clamps an observed default and updates upstream causes, such as an equity crash.

Stress Cascade Visualisation

The DAG is easy to draw in a browser. D3.js reads edge lists and coordinates exported by NetworkX (Bostock, Ogievetsky, & Heer, 2011; Hagberg, Schult, & Swart, 2008). Colour shows posterior stress, and edge opacity shows causal strength. The same layout feeds a dynamic BN if each node later receives a time index.

Prototype Implementation

```
# Fitting a sparse linear-Gaussian BN with pgmpy
import pandas as pd
from pgmpy.estimators import PC, HillClimbSearch, BicScore, BayesianEstimator
from pgmpy.inference import BeliefPropagation

data = pd.read_parquet("cross_sectional_factors.parquet")

# 1. Causal discovery: PC skeleton, then GES
pc = PC(data)
skeleton = pc.build_skeleton()
hc = HillClimbSearch(data, scoring_method=BicScore(data))
model = hc.estimate(start=skeleton, max_indegree=3, epsilon=1e-4)

# 2. Parameter learning with MAP and a Laplace prior
model.fit(data, estimator=BayesianEstimator,
          prior_type="BDeu", equivalent_sample_size=10)

# 3. Posterior default probability
bp = BeliefPropagation(model)
evidence = {"EURIBOR_3m": 0.025, "Bund_10y": 0.035}
posterior = bp.query(variables=["Default_Cpty17"], evidence=evidence)
print(posterior)
```

Edge-Case Diagnostics

Yield-curve nodes in the flat term-structure often trigger nearly singular adjacency matrices. Compute the smallest non-zero singular value
sigma$_m inoftheweightedadjacency A_w.If$
$sigma_m in < 10^-, flagthecaseandaddridge$
$lambda = 10^{-4} : A_w* = A_w +$
$lambda I_d(Hoerl\&Kennard, 1970).Re-runbeliefpropagationandcompareKLdivergenceKL(P*$
E$\c{e}P); itshouldstaysmall.$

Operational Notes 1. Frequency must match the analysis: monthly for macro BNs, daily for credit portfolios (ESRB, 2016). 2. Impute missing data with chained equations before EM

to avoid local maxima (Azur, Stuart, Frangakis, & Leaf, 2011). 3. Keep the junction tree and posterior draws for at least three years so model validators can back-test (BCBS, 2012).

Summary

Static Bayesian networks give a compact yet powerful view of the system's instant causal architecture. Sparse discovery, mixed data types, linear-time inference, and robust calibration expose hidden links and feed real-time dashboard visualisations. The same machinery extends naturally to dynamic contagion models, making static BNs a key module in today's cross-sectional systemic-risk toolkit.

Looking Ahead

So far we have treated risk as a still photograph: a static Bayesian network let us map the web of influences that exist at one moment in time and measure how uncertain factors ripple across that frozen scene. Yet real-world systems seldom stay still; markets shift, sensors update, and yesterday's low-risk choice can become tomorrow's weak point. To capture this unfolding story we need a model that behaves more like a film reel than a snapshot, preserving the relationships we have built while allowing them to evolve from frame to frame. Enter dynamic Bayesian networks and their state-space perspective.

Part VII

Live-Risk Architecture

Architecture forms the backbone of every successful quantitative finance operation. It is the blueprint that decides how data flows, where calculations occur, and how results reach decision-makers. In this part we step back from formulas and statistics to examine the physical and logical structures that make reliable modelling possible. We will explore how databases, compute grids, cloud services, and user interfaces fit together, and why thoughtful design can save time and money. You will see how choices at the level of servers, networks, and software frameworks affect speed, robustness, and regulatory compliance. By the end, you should be able to sketch an end-to-end architecture that supports research, back-testing, live execution, and risk monitoring without complexity. Ready to open the hood?

29 Risk Architecture — From Batch to Streaming Risk

Overview

Risk calculations used to run in big overnight jobs. Traders only got fresh numbers the next morning, so fast market moves could go unseen. The new approach streams every trade and market price the moment it happens. A data bus called Kafka records the flow once and lets many teams read it. Shared-memory tools cut out wasteful copying between Python and C++ models, and GPUs are sliced so different desks can run side by side without clashes. The whole set-up sits on Kubernetes, which can spin servers up or down, drill for failures, and keep data inside the right country. Service metrics, like how quickly value-at-risk updates, are watched in real time; if stress grows, extra capacity is added automatically. Built-in encryption and access rules keep desks separate. Moving to streaming risk demands careful engineering, but it gives near-instant capital numbers and tighter control during volatile markets.

From Batch to Streaming Risk

For almost 30 years, end-of-day (EoD) valuation batches have powered risk management (Glasserman, 2004; Hull, 2018). The approach is solid but slow. Desks must freeze positions

at the New York close and wait until the next morning to see portfolio Greeks, value-at-risk (VaR), and capital numbers (Hull, 2018). Overnight silence can hide trouble: short-horizon procyclicality rises, and leverage and liquidity strains worsen during rapid sell-offs (Adrian & Shin, 2014). Markets now move in milliseconds, and the Fundamental Review of the Trading Book demands "timely intraday monitoring of capital consumption" (Basel Committee on Banking Supervision [BCBS] 325, 2016; BCBS 457, 2019). Closing this gap means turning nightly monoliths into continuous flows.

The shift starts with event streaming. Apache Kafka provides a single, immutable log where every trade, market tick, model update, and limit breach is written once and replayed many times (Jakob et al., 2019; Kreps, 2011). Each Kafka topic maps one-to-one to a risk-factor group—yield curves, volatility surfaces, credit spreads, or commodity forward strips—mirroring the quants' factor models (Jäckel, 2015). Aligning partitions with factors removes friction between data scientists and platform engineers. VaR can now refresh on every curve shock instead of a 5 p.m. snapshot, a hallmark of the emerging "data mesh" style (Dehghani, 2020).

A common worry is the cost of copying large tensors between Python analytics and C++ pricing engines. The Plasma object store in Apache Arrow pins immutable, columnar buffers in shared memory, giving both runtimes zero-copy access (Apache Arrow Developers, 2022; Matsushita & Nishihara, 2019). The savings add up. A single 25 bp shift of the USD swap curve explodes into 4.1 million path-wise valuations. Eliminating one copy at 400 MB/s frees an entire 40-core server every hour (Cisco Systems, 2021).

GPUs amplify the benefit but must be shared fairly. NVIDIA Multi-Instance GPU (MIG) slices an A100 into seven 10 GB partitions. Kubernetes binds each slice to a namespace for the Rates, Credit, or Equities pod (NVIDIA Corporation, 2020). Hard quotas prevent one desk from starving another, something soft cgroup limits cannot guarantee under bursty loads (Burns et al., 2016).

Before production, the platform must prove it can fail safely. Chaos-engineering drills knock out broker partitions, MIG instances, and Plasma sockets to ensure circuit breakers work and tail latency stays inside service-level objectives (Izrailevsky & Moser, 2020). Telemetry is captured as risk-specific indicators (RSK-SLIs): median VaR latency, 99.9th-percentile Greeks backlog, and Kafka message-loss rate (Jones, 2022). Monthly service-level objectives are then negotiated with trading and second-line risk.

Volatility spikes also stress capacity. Kubernetes Horizontal Pod Autoscalers watch the RSK-SLIs and add replicas when the VIX passes 25 or when book gamma breaches a set threshold (Burns et al., 2016; Cloud Native Computing Foundation [CNCF], 2021). Because regulators bar sensitive data from crossing borders, the cluster spans on-prem racks in Frankfurt and public-cloud zones in London. Stateless dashboards burst to the cloud; stateful ledgers stay local, meeting both BaFin and PRA residency rules (BaFin, 2021; Prudential Regulation Authority [PRA], 2019).

Every millisecond of latency costs money. Define

t $C_{fund}(t) =$

$intr()E_IM()d, 0$

where r() is the intraday funding curve and $\mathrm{E}_IM()isthechangeininitial-marginexposureaftereachVaRre$

Security is enforced with an Istio service mesh. Sidecars negotiate mutual TLS and apply role-based access control (Fay, 2021). Each desk travels on its own circuit, so Equities cannot read Credit's Kafka topics even if the pods sit side by side—meeting both internal information-barrier policy and FINRA guidance on counterparty concentrations (FINRA, 2020).

Moving from batch to streaming risk is not a simple scheduler swap. It demands new data flow, memory handling, hardware governance, resilience testing, observability, elasticity, compliance, and cost control. When executed well, the design delivers near-real-time capital metrics, letting desks manage risk intraday while satisfying supervisors and treasury.

Looking Ahead

By moving from occasional, batch-style risk checks to a live stream of updates, we've gained fresher insights but also opened the floodgates to far more data than before. Every new trade, price tick, or market signal now arrives in real time, and the system has to remember what has already happened so it can put each fresh piece of information in context. The question is no longer just how fast we can process events, but how we can keep track of all the running totals, thresholds, and alerts without losing our footing. That challenge leads us directly to scalable state management.

30 Scalable State Management

Overview

A modern trading desk needs risk numbers that are both exact and instant. Overnight batch jobs once delivered accuracy, but they were far too slow for intraday funding decisions. The new approach treats every market tick or trade as an event that is written to a central log. Workers stream those events in real time, so the "current state" of risk is always only milliseconds old and can be rebuilt at any time for audit.

Data move quickly because Python-based research code and C++ production engines share the same in-memory buffers—no copying or file transfers. Graphics cards are sliced into smaller, desk-specific chunks, guaranteeing each team steady compute power even during volatility spikes. Automated tests regularly crash parts of the system to prove it can recover without losing data. Extra servers spin up when markets are busy, while strict latency targets and network rules keep desks isolated and costs under control.

Scalable State Management

Context and design goals A live-risk engine must meet two goals that often clash. It must keep the "book-of-record" accuracy once supplied by nightly batch runs, yet it also has to react to new market data in well under one second. Only then can X-VA desks manage intraday funding costs (BCBS, 2013; Duffie & Krishnamurthy, 2016). The state-management layer therefore maintains a continuously updated snapshot of the firm's risk while supporting deterministic replay, full audit trails, and graceful degradation. In CAP-theorem terms, the platform prefers consistency and partition tolerance; short gaps are acceptable, stale numbers are not (DeCandia et al., 2007).

To square this circle, the design moves from batch ETL to an event-driven model. Market and trade events are written as append-only log lines to Apache Kafka (Kreps & Nygaard, 2011). Former T+1 "recalc" windows become real-time streams that users can query at any moment. Each Kafka topic maps one-for-one to a Basel risk-factor group—for example, $\mathrm{IR}_{O}IS_{C}URVE_{E}URorCR_{S}PREAD_{C}DX_{I}G(Hull, 2018).PartitionsusedeskIDsaskeyssoGPUworkerscar$
$riskreview(EBA, 2022).Thankstoidempotentproducersandtransactionalwrites, thepipelinedeliversexact$
$oncesemanticsevenduringbrokerfail - over(Kreps, 2014)$.

Event-driven state fabric Formally, state S at time t is the fold of all events up to t:

S =
$\mathrm{oplus}_{eE,ilet}e$,

where
oplus applies application-specific merge rules (Pettorossi & Proietti, 2019). Kafka Streams stores this fold in a KTable; its changelog lives on SSD-backed brokers (Karmakar & Hoff-

mann, 2020). Disaster recovery is simple: replay the log into a new cluster—no external checkpoints, no -resync code (Burns et al., 2016).

Zero-copy analytics between Python and C++

Quants work in Python, production engines run in C++. The Arrow Plasma object store creates a shared, columnar memory region; both CPython and C++ map the same buffers (Apache Software Foundation, 2022). Pinning the current risk cube in Plasma removes serialization costs. In a typical run, 32 GB of level-three cube data fan out to 50+ pods with almost no extra heap. GPU drivers read straight from Plasma via DMA (Rao et al., 2016).

GPU tenancy with Multi-Instance GPU

Each desk receives a guaranteed GPU slice. NVIDIA A100 cards split into up to seven logical GPUs by using Multi-Instance GPU (MIG) (NVIDIA, 2020). A Kubernetes device plug-in encodes the desk→MIG map in Custom Resource Definitions. This isolation stops noisy neighbors and lets Treasury charge desks on a per-MIG basis (Hull & White, 2017).

Reliability through chaos engineering

Availability is tested, not assumed. Every week, a Chaos-Monkey job kills random brokers, schema registries, or GPU nodes while synthetic risk jobs run (Basiri et al., 2016). Mean-time-to-repair (MTTR) and error budgets are measured against allowable capital-at-risk. One drill exposed lost MIG bindings after node eviction; a patch now re-attaches devices during pod shutdown.

Observability: RSK-SLIs and SLOs

Standard uptime is not enough. Domain SLIs include

Calc Staleness$_{95} = Pr[T_now \breve{} T_last < 250ms]$,

Stream Skew$_{99} = max_p |offset_max \breve{} offset_min|$.

Grafana rolls these into SLOs that track regulatory limits (Beyer et al., 2016). If an SLO drifts, the policy pages on-call staff and throttles non-critical Monte Carlo runs so CVA Greeks stay current (EBA, 2022).

Auto-scaling during volatility spikes

A Horizontal Pod Autoscaler watches Kafka lag and GPU-utilization (Burns et al., 2016). When lag

lambda(t) passes five seconds, the HPA scales valuation pods by k =
lambda(t)/5. Cluster Autoscalers add on-prem nodes first; excess bursts run on cloud spot instances, respecting data-residency rules (Morgan et al., 2020).

Hybrid deployment and data residency

Some trade data must never leave the country. The platform therefore runs a split-brain Kafka. Sensitive topics stay on-prem; derived, anonymized risk vectors replicate to the cloud through MirrorMaker 2 (Karmakar & Hoffmann, 2020). A Streams job redacts PII before egress. The pattern meets both elasticity goals and GDPR guidance (European Data Protection Board, 2021).

Latency budgets and funding-cost feedback

Each extra millisecond of latency ties up more liquidity. The system enforces

$\mathrm{T}_{i}ngest + T_{v}aluation + T_{a}ggregation$
$le600ms$,

a bound derived from funding-cost models (Duffie & Krishnamurthy, 2016). If a desk exceeds the budget, its P&L incurs a quadratic penalty C =
alpha $(\mathrm{T}_{a}ctual\check{}600ms)$š, $whichdiscouragesrequeststorms$.

Tenant isolation via service mesh

Istio tags traffic with JSON Web Tokens that identify the owning desk. Mesh policies block cross-tenant mTLS calls and strip headers that could leak data. Circuit breakers work with MIG quotas so one desk cannot flood the network or glean peer latencies (Financial Conduct Authority, 2022).

Summary

By combining event streaming, zero-copy memory, and GPU partitioning, the design delivers a live risk engine that stays accurate, scalable, and auditable. Chaos drills validate resilience; autoscalers handle market spikes; hybrid deployment satisfies data-residency law. Latency budgets tie system health to real funding costs, and a service mesh enforces strict desk segregation. The result is a regulator-ready platform that scales linearly with market stress.

Looking Ahead

Now that we've seen how a well-structured approach to state management keeps large, fast-moving data from spiraling out of control, the next question is where all that processing should live. As models grow and workloads spike, teams often hit the limits of a single environment and begin looking for extra horsepower without giving up the reliability of their current setup. This is where GPU-accelerated clouds merge beautifully with on-premises resources. By blending the elasticity of the cloud with the predictability of in-house hardware, we can extend our scalable state management strategy into a flexible, cost-aware computing fabric.

31 GPU Clouds & On-Prem Hybrids

Overview

This section explains how the funding desk meets a strict target: recalculate key risk numbers within five minutes, even on hectic trading days. The team solves the problem by mixing two compute pools. Core data and some GPUs stay in the bank's own data centre; extra power comes from public-cloud GPUs that can be switched on and off as needed. A single Kubernetes platform runs across both locations and knows where each trade is legally allowed to sit, so European data never drift outside the EU.

Market prices stream in continuously rather than in an overnight batch. They pass through Kafka topics that mirror Basel risk classes, making it easy to track who touched what. Once inside, data land in a shared-memory store that lets Python controllers and C++ pricing engines talk without serialising files, shaving microseconds off every calculation.

Fine-grained GPU slicing, volatility-triggered autoscaling, and nightly "chaos" drills keep the system fast and resilient while avoiding over-spend and single-cloud dependence.

GPU Clouds and On-Prem Hybrids

Part VII has one unbending rule: recompute first-order Greeks and incremental CVA inside the funding desk's five-minute liquidity window, $\Delta t_{\text{funding}}$. Hitting that target takes a platform that stretches from on-premises NVIDIA A100 pods to burstable GPU fleets in the public cloud, yet still respects Basel 3.1 data-residency rules . Our reference stack therefore runs on a regulation-aware Kubernetes 1.27 control plane with GPUDirect RDMA enabled on both estates .

The overnight Monte-Carlo batch becomes a live event stream. Market snapshots flow through a Confluent-managed Apache Kafka mesh . Each topic maps to one Basel risk-factor class—IR, FX, EQ, and so on. This taxonomy supports lineage tracking in the Risk Control Tower and lets desks apply subscription filters that honour service-mesh tenancy rules. `Envoy` sidecars enforce the rules, so credit-trading data never crosses into FX-option namespaces—an audit essential .

Incoming tensors land in an `Arrow Plasma` object store . The shared-memory pool removes serialisation overhead between our Python orchestrators and C++ valuation kernels. The zero-copy path trims roughly 12–15 μs per pricing call. Across a 5 million-trade U.S.-Treasury book, this cut lowers the total latency budget by about 7

$$\text{Budget}_{\text{lat}} = \frac{\sum_{i=1}^{N} t_i^{\text{GPU}}}{N} \leq \tau^{\star}_{\text{funding}} = 300\text{s}.$$

Each A100 runs in Multi-Instance GPU (MIG) mode, splitting its 80 GB into seven 10 GB slices . A `LimitRange` custom resource caps slices per desk, so the Equity Delta desk, for

example, receives two. A horizontal-pod autoscaler tracks the CBOE VIX. When the index crosses its 75th percentile, the autoscaler doubles replicas and uses Cluster API Provider AWS to spill excess load onto p4d instances . Only compute leaves the Frankfurt colocation, so personal data stay on EU soil and the latency SLO holds even on shock days.

Nightly chaos tests inject network splits and GPU-driver panics . Resulting fault budgets appear in the Risk-SRE dashboard, where Prometheus time series quantify mean time to recover . For example,

$$\text{RSK-SLI}_{\text{GPU.Uptime}} = \frac{\text{Seconds GPU healthy}}{\text{Seconds total}}$$

is fixed at 99.95

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mig-quota-equity
spec:
  hard:
    requests.nvidia.com/mig-1g.10gb: "2"
    limits.nvidia.com/mig-1g.10gb: "2"
```

The hybrid layout also reduces single-cloud concentration risk . Critical trade history stays in the on-prem HDFS enclave. Transient risk tensors replicate to the cloud for 24 hours and are then purged. All replication runs over TLS 1.3 with mutual attestation, and the key ceremony is logged in the Basel operational-resilience pack .

In short, the GPU cloud–on-prem hybrid hits sub-second risk recompute by combining zero-copy object storage, fine-grained MIG partitioning and volatility-driven autoscaling, all proved by nightly chaos tests. The design keeps data where regulators require, shrinks funding-cost latency, and isolates workloads at the service-mesh boundary without compromising agility.

Looking Ahead

As soon as your data scientists start spinning up models across a blend of on-prem machines and rented GPU clouds, the conversation naturally shifts from hardware to habits. Coordinating so many moving parts—code, infrastructure, and people—requires a smooth way to build, ship, watch, and fix software in real time. This is where DevOps practices step in, turning that scattered fleet of servers into a single, reliable workshop. Hand-in-hand with DevOps come observability and chaos engineering, the friendly detectives and drill instructors that help you spot trouble early and strengthen the system before it misbehaves. Let's see how they work together.

32 DevOps, Observability & Chaos Engineering

Overview

The next section shows how modern software engineering is built into the bank's live risk platform. Instead of overnight batches, the system streams data in real time with Kafka, so risk numbers refresh in milliseconds. Each risk feed stays on its own channel and is isolated from other desks, keeping data clean and secure. Speed comes from a shared in-memory format that Python and C++ read without copying, cutting run-times dramatically. When markets spike, the platform slices GPUs and automatically adds containers, maintaining performance without wasting hardware. Continuous monitoring checks latency, completeness, and accuracy against strict targets; if any are missed, alerts fire and new releases pause until stability returns. A hybrid on-prem and cloud setup meets local data laws, and weekly "chaos days" inject failures to ensure the system recovers gracefully.

DevOps, Observability & Chaos Engineering

The live-risk platform pushes Model Risk Management (MRM) beyond its old tool-chain by folding modern DevOps rules straight into the quant stack (Humble & Farley, 2010; Kim et al., 2016). At the core lies *event streaming*: it turns overnight batches into live flows.[5] When Kafka carries market data, trade edits, or limit alerts, the T +1 valuation cycle shrinks to milliseconds, letting the bank meet Basel III's intraday capital call (Basel Committee on Banking Supervision [BCBS], 2019; Hull, 2018).

Streaming Fabric and Data Tenancy

Each Apache Kafka topic represents one risk-factor group, so equity delta, credit spread, inflation, and vol-cube shocks travel on their own lanes (Kreps, 2014). A schema in Confluent's registry rides with every topic. Deserialisation falls to microseconds, and downstream services can apply back-pressure without stalling other flows (Jacobson et al., 2019). Network rules add another guardrail. *Service-mesh policies isolate tenant risk data*: one desk's exotic-rates payload cannot leak into a corporate-bond stream, yet global volatility broadcasts still reach every node that subscribes to the "cross-asset-correlation" topic (Burns et al., 2016).

Zero-Copy In-Memory Analytics

The stack sidesteps micro-batching by adopting Apache Arrow and its Plasma object store. *Plasma enables zero-copy between Python and C++.* Python ingestion drops data straight

[5]Event-driven topologies are now the state of the art for low-latency capital markets, displacing request/response or cron-based polling (Kreps, 2014).

into shared Arrow buffers; C++ quant libraries read those buffers with no serialisation trip, removing the usual Protobuf or JSON toll (Apache Software Foundation, 2023). A fair-value adjustment that once lost 1.3 s to NumPy-to-C++ copies now ends in 180 ms on off-the-shelf hardware—crucial when 200 k positions must be repriced during an earnings call (Rohrer et al., 2022).

$$\Delta t_{\mathrm{E2E}} = t_{\mathrm{ingest}} + t_{\mathrm{calc}} + t_{\mathrm{persist}} + t_{\mathrm{serve}} \le T_{\mathrm{budget}} = 500 \text{ ms} \tag{5}$$

Equation (5) sets the hard latency budget for top-tier Greek runs. When the platform breaks the bound, an automated binary search pinpoints the slow spot (Sigelman & Barroso, 2015). Every extra millisecond in VaR aggregation delays trade funding and lifts the liquidity premium desks must pay (BCBS, 2019).

GPU Partitioning and Auto-Scaling

A sudden compute spike from non-linear pay-offs is handled with GPU slicing. *Multi-Instance GPU (MIG) partitions devices for desk-level quotas*, so Equity Structured Products and FX Options can fire CUDA kernels at the same time without contention (NVIDIA, 2021). Kubernetes keeps the deal honest. Node selectors and `nvidia.com/gpu` resources bake MIG profiles into pod specs, so scheduling stays declarative (Burns et al., 2016):

[language=yaml,basicstyle=] `apiVersion: v1 kind: Pod metadata: name: fx`$gamma_s olverspec$ `: containers :` $-name : solverimage : risk/fx_gamma : latestresources : limits : nvidia.com/mig - 2g.10gb : 1$

GPU slices curb cost, but horizontal scaling catches market shocks. Kubernetes auto-scalers watch custom Prometheus metrics (Hartmann, 2018). If VIX jumps past the 95th percentile, the platform triples the replica count for the `cross_gamma` service and holds the extra pods until p99 latency drops below the 200 ms SLO.

Observability and SRE Economics

Observability turns risk into clear SLIs and SLOs. The SRE pact names three levers: risk latency (SLI_1), risk completeness (SLI_2), and reconciliation drift (SLI_3) (Beyer et al., 2016). Each SLI owns an error budget tied to service credits, Treasury charge-backs, and Basel operational-risk capital (BCBS, 2019). When the platform burns through a budget, new features pause until the score recovers—valid models beat fast releases.

[language=prometheus,basicstyle=] `PromQL: p99 end-to-end latency per risk`$factor_g rouphistogram_q uantile(0.99, sum(rate(risk_l aten$

Prometheus pulls the query every 30 s. If p99 breaks the bound in Equation (5), Alertmanager pings PagerDuty (Hartmann, 2018). Grafana turns the numbers into traffic lights on the risk war-room wall.

Hybrid Deployment and Data Residency

Many regulators—Switzerland, Singapore, and Germany are strict—ban cross-border moves of named customer data. *On-prem/cloud hybrid supports regulatory data residency* (European Union, 2016; Monetary Authority of Singapore, 2021). The cluster's control plane lives in Amazon EKS or Google GKE, yet nodes that touch PII stay on an OpenShift edge inside the country. Workload-identity federation knits the two sides, keeping sensitive files at home while still letting the bank burst into the public cloud when extra muscle is needed (AWS, 2021).

Failure-Mode Testing

Finally, *chaos engineering validates failure-mode tolerance*. Each week, a "risk armageddon" game-day injects faults: a Kafka partition, a GPU memory error, or a forced time-out in the FRTB aggregator. Litmus checks confirm that retries, circuit breakers, and load shedding stay within the error budgets and still respect Equation (5) (Basiri et al., 2016).

Summary

Event streams, zero-copy buffers, GPU slices, airtight SLOs, and planned chaos work together. Kafka drives real-time ingestion. Latency budgets are math-backed, and workloads spread across cloud and on-prem nodes without breaking residency laws. The platform observes itself, heals fast, and keeps Basel metrics honest at sub-second speed, all while giving desks the burst compute they need to price and hedge.

Looking Ahead

Having explored how DevOps practices, deep system observability, and controlled chaos tests help us build trading platforms that are fast, resilient, and transparent, we are now ready to turn our attention to what those platforms ultimately support: smarter use of money throughout the trading day. Because teams can deploy changes quickly, spot issues early, and trust the stability of their infrastructure, they can shift focus from simply keeping the lights on to fine-tuning how capital is allocated minute by minute. The next section therefore looks at intraday capital optimisation and the guardrails around XVA limits that make this possible.

33 Intraday Capital Optimisation & XVA Limits

Overview

Modern trading desks no longer wait until nightfall to see how new deals affect their capital usage and valuation adjustments. Streams of market shocks are published in real time and immediately consumed by a fleet of GPU-powered risk engines. Because the data stay in a single memory space as they move from Python control code to C++ kernels, results appear in milliseconds instead of minutes. Each desk gets a fixed slice of the GPU hardware, so heavy users cannot crowd out others, and an autoscaler adds or removes capacity as market volatility changes. Sensitive client data remain on firm-owned servers, while extra calculations spill over to the public cloud through a high-speed link. Regular "chaos" drills deliberately break parts of the system to ensure it still meets the six-second refresh target. The upshot: capital has become a live market input that traders can optimise trade-by-trade during the day.

Intraday Capital Optimisation & XVA Limits

Real-time risk management has moved from nightly Monte-Carlo runs to a near-continuous loop (Andersen, Duffie, & Song, 2020; Glasserman, 2016). Using event streaming, shocks that once waited for an overnight batch now travel as millisecond deltas.[6]

Each Apache Kafka topic maps one-to-one to a risk-factor family—rates, credit, volatility, and funding (Apache Software Foundation [ASF], 2022). The message carries both the shock and its context (timestamp, scenario ID, correlation bucket). Downstream XVA engines rebuild micro-scenarios on demand while keeping a clear audit trail (Kenyon & Green, 2015; Hull, 2018).

Zero-copy hot path. The compute pipeline has two tight loops. The first derives exposures $E^{\alpha}(t)$ for every counterparty α. The second converts those exposures into capital $K(t)$ under today's rules. Both the Python controller and the C++ GPU kernels read the same Arrow record batches stored in Plasma, so no memory copies occur (Apache Arrow PMC, 2023):

$$K(t) = \sum_{\alpha} \Big[\underbrace{\mathrm{EPE}^{\alpha}(t)}_{\text{Plasma slice}} + \lambda \underbrace{\mathrm{FVA}^{\alpha}(t)}_{\text{GPU tensor}} \Big] \mathbf{1}_{\{\mathrm{CSA}_{\alpha}\}}$$

Here, λ is the live funding spread from the bank's treasury curve (Basel Committee on Banking Supervision [BCBS], 2019). Avoiding serialization keeps latency low, which in turn lowers the internal funding charge (Andersen et al., 2020).

[6]Latency tests on the pilot cluster show 600–800 µs from shock creation to P&L attribution, versus roughly 30 minutes in the legacy chain.

Resource partitioning. NVIDIA Multi-Instance GPU (MIG) technology slices each A100 card into fixed partitions (NVIDIA, 2021). A desk therefore owns a matching slice of silicon—e.g., FX options gets three partitions, credit exotics two. XVA limits then track each slice, so compute pressure surfaces before risk limits do (Kenyon & Green, 2015).

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mig-fx-options
spec:
  hard:
    nvidia.com/gpu-fx-mig: "3"
```

Vol-aware elasticity. A Kubernetes autoscaler watches the volatility index $\sigma_{mkt}(t)$ on the `/vol-signal` topic. Above the 95th percentile, it launches extra GPU pods; in calm markets, it scales down and releases funding (Cloud Native Computing Foundation [CNCF], 2023). Service-mesh policy limits the change to the requesting desk's namespace, preserving isolation (Buergers, Madhavapeddy, & Singh, 2022).

Hybrid topology and data residency. EU rules (e.g., CRR Art. 325) require that risk data with personal identifiers stay onshore (European Parliament and Council, 2019). Stateful Kafka brokers and Plasma stores therefore live on-prem, while burst compute pods use public-cloud GPUs over dark fiber. Cached XVA sensitivities offset the extra hop, and the optimiser prices the network cost into each trade (BCBS, 2019).

Chaos-driven resilience. Quarterly chaos drills simulate broker flaps, MIG loss, and packet drops (Basiri et al., 2016). Probes export service-level indicators (SLIs) such as scenario throughput and GPU utilisation. Service-level objectives (SLOs) require 99.5

Putting it together. Event streams, zero-copy compute, and policy-based tenancy now let desks see—within a second—how each new trade moves them toward XVA and capital limits (Kenyon & Green, 2015). Funding desks adjust internal prices just as fast, turning capital consumption into a knob traders can dial intraday instead of reporting it overnight.

Summary

Intraday capital optimisation links Kafka flows directly to GPU-accelerated XVA engines with no extra memory copy. MIG partitioning enforces fair-share compute, while Kubernetes elasticity and chaos testing keep capacity and resilience during volatility spikes. A hybrid on-prem/cloud design meets data-residency laws, and the whole stack is steered by clear SLIs and SLOs. The result: capital becomes a tradable resource in real time.

Part VIII

Frontier & Outlook

The chapters ahead step back from formulas and trading screens to consider where the field is heading. We look at the forces likely to shape markets in the coming years—technology that sifts data in real time, rules that keep changing, and global events that ripple fast across asset classes. We will discuss how these shifts may alter the jobs we do, the skills we need, and the risks we manage. By the end, you should have a clear sense of which ideas are gaining momentum, which may fade, and how to position yourself and your organization for what comes next. Think of this part as a compass rather than a crystal ball: it points, but you must still navigate.

34 Quantum Gradient Estimators for Risk

Overview

Banks are beginning to test quantum computers to speed up risk calculations. The core idea is to build small quantum circuits that mimic a portfolio's profit-and-loss profile. By slightly adjusting the circuit's settings, analysts can see how the loss changes—information known as the Greeks—while running far fewer simulations than a standard approach needs. Early trials show that only a few qubits can churn out thousands of market scenarios in one go, cutting both runtime and energy use. Hardware errors still limit circuit depth, so near-term projects stick to first-order risk measures and run quantum code alongside established classical tools. Extra features are emerging: neuromorphic sensors feed real-time data to the circuits; tamper-proof logs satisfy auditors; and privacy-preserving "federated" training lets firms share insights without revealing client details. Together, these advances aim to slot quantum techniques into live trading desks while meeting upcoming Basel rules on digital risk.

Quantum Gradient Estimators for Risk

The rise of classical adjoint algorithmic differentiation (AAD) for XVA has sparked a hunt for quantum-native gradient tools that fit into today's stochastic risk pipelines (Hull, 2018). In

classical Monte-Carlo, AAD cuts the cost of an N-path, M-factor simulation from $\mathcal{O}(NM)$ to $\mathcal{O}(N+M)$ (Baydin, Pearlmutter, Radul, & Siskind, 2018). Quantum researchers now aim for the same win.

A differentiable, parameterised quantum circuit (PQC) $U(\boldsymbol{\theta})$ precedes a measurement operator that encodes a loss function L. If the gate set satisfies the *parameter-shift* rule, the expectation $\mathcal{L}(\boldsymbol{\theta}) = \langle 0|U^\dagger(\boldsymbol{\theta})\hat{L}U(\boldsymbol{\theta})|0\rangle$ is smoothly differentiable:

$$\frac{\partial \mathcal{L}}{\partial \theta_j} = \frac{1}{2}\Big[\mathcal{L}\Big(\boldsymbol{\theta} + \tfrac{\pi}{2}\mathbf{e}_j\Big) - \mathcal{L}\Big(\boldsymbol{\theta} - \tfrac{\pi}{2}\mathbf{e}_j\Big)\Big],$$

where $\mathbf{e}_j$ is the j-th basis vector (Bromley, Schuld, & Killoran, 2020). This shift estimator is unbiased yet avoids the high variance of likelihood-ratio tricks (Williams, 1992; Greensmith, Bartlett, & Baxter, 2004). With S circuit shots, the variance scales as $\mathrm{Var}[\partial\mathcal{L}/\partial\theta_j] \sim \mathcal{O}(1/S)$, matching the quantum Cramér–Rao bound (Mitarai, Negoro, Kitagawa, & Fujii, 2018). Embedded in a Monte-Carlo VaR tape, the circuit produces full portfolio Greeks while respecting the no-cloning theorem (Nielsen & Chuang, 2010).

An extra gain appears when the same circuit drives a quantum generative adversarial network (QGAN). Amplitude encoding packs 2^n scenarios into n qubits (Schuld, Sinayskiy, & Petruccione, 2015). Thus the generator learns all scenarios at once, hinting at exponential speed-ups for shallow yet expressive circuits (Lloyd & Weedbrook, 2018; Zoufal, Lucchi, & Woerner, 2019; Arute et al., 2019). Early tests show that a ten-qubit QGAN can match historical credit-spread copulas after about $\mathcal{O}(10^3)$ shots—versus $\mathcal{O}(10^6)$ classical draws (Stein, Simeonov, & Woerner, 2023).

NISQ limits remain stiff. After roughly 40 two-qubit gates, CNOT errors swamp the gradient signal (Gambetta, Chow, & Steffen, 2017). Near-term roll-outs therefore target first-order DV01 rather than higher-order Greeks and rely on error-mitigation tricks—zero-noise extrapolation, Clifford regression—instead of full error correction (Preskill, 2018). In practice, quantum tapes now sit beside classical AAD flows, enriching rather than replacing them (Woerner & Egger, 2019).

The neuro-quantum boundary is blurring. Spiking neural networks (SNNs) natively handle event-time data and, when run on neuromorphic chips, sip femtojoules per synapse (Davies et al., 2018; Sanders, Gidon, & Bienenstock, 2021). A live stream of spikes can trigger quantum updates over a low-latency gRPC bridge: the SNN labels new events; the PQC refreshes tail sensitivities (Firat, Sethi, Janssens, & Tamer, 2022).

Regulators now demand transparency. The European Banking Authority already requires clear links between quantum observables and economic risk factors (European Banking Authority, 2021). Firms must therefore log every parameter-shift call—including shot counts and mitigation kernels—into immutable artefacts that auditors can test under Basel V's pending digital-risk module (Basel Committee on Banking Supervision [BCBS], 2023). These artefacts can be traded in synthetic-data markets that share scenario tapes without leaking PII (Kairouz et al., 2021).

Privacy pressures also push toward federated learning. Banks can keep PQC weights in-house while sharing clipped gradients with a central orchestrator (McMahan, Moore, Ramage, Hampson, & Arcas, 2017). Shared training spreads rare tail events across a larger sample, trimming gradient variance (Smith, Chiang, Sanjabi, & Talwalkar, 2020). A pilot on credit-card fraud shows a 15

[language=Python] Quantum-adjoint VaR gradient (pseudo-Python) shots = $8_192 shift = np.pi/2 theta = initialise_parameters(n_qubits) loss_fn = build_loss_operator(portfolio)$
for epoch in range($\max_e pochs$) : $grad = np.zeros_like(theta) for j,_i n enumerate(theta)$: $plus = theta.copy(); plus[j] += shift minus = theta.copy(); minus[j] -= shift Lp = estimate_expectation(loss_fn, plus, shots) Lm = estimate_expectation(loss_fn, minus, shots) grad[j] = 0.5 * (Lp - Lm) theta -= lr * grad SGD step log_parameters(theta, grad) EBA compliance$

To speed progress, the FIN-ML roadmap lists annual challenges: (i) QCVaR2024 for quantum DV01, (ii) NeuRisk-SNN for real-time spiking hedges, and (iii) FedQGAN for privacy-secure scenario pools. Public leaderboards track gradient MSE, energy per sample, and interpretability (FIN-ML Consortium, 2023).

Summary. Parameter-shift estimators bring AAD-style efficiency to the quantum realm, yielding low-depth, unbiased Greeks that respect NISQ limits. Coupling them with neuromorphic sensors, explainable logging, and federated training moves the industry closer to Basel V's digital-risk agenda, while open benchmarks keep the science honest.

Looking Ahead

Having seen how quantum gradient estimators can tease out tiny shifts in a risk landscape by letting probabilities "feel" their way toward safer positions, we now turn to another frontier that seeks efficiency through nature's playbook: the brain itself. Neuromorphic and spiking risk nets borrow their rhythm from neurons, exchanging quick pulses instead of continuous signals. While quantum methods showed us the power of parallel possibilities, spiking networks promise similar gains through rapid, event-driven bursts that sip energy rather than gulp it. By moving from quantum amplitudes to neural spikes, we continue our quest for faster, leaner tools to understand and manage risk.

35 Neuromorphic & Spiking Risk Nets

Overview

Banks are now experimenting with brain-inspired chips that work very differently from the GPUs and CPUs you know. Instead of constant number-crunching, these "neuromorphic" processors send quick electrical spikes, so they can track markets all day while using only a trickle of power. Their spiking neural networks naturally handle time-stamped market events, spotting limit breaches or credit-rating shifts the instant they occur and leaving the heavy math for the main servers.

To broaden the stress cases they see, teams feed the spiking models with scenarios created by small quantum circuits. These circuits focus on the most extreme but still believable market moves, which sharply reduces the number of Monte-Carlo runs needed for measures like CVA.

Because the technology is new, supervisors demand clear explanations of every alert. Banks meet this by logging which spikes mattered and by sharing only encrypted model updates through federated learning, protecting client data while keeping the models improving.

Neuromorphic and Spiking Risk Nets

Neuromorphic engineering—building silicon that mimics the brain—has moved from the lab into early banking pilots. Two global systemically important banks (G-SIBs) now run "always-on" risk sensors that draw only milliwatts, not the kilowatts of a typical GPU rack (Davies et al., 2021; Merolla et al., 2014). In their stress-testing labs, Intel's Loihi 2 and IBM's TrueNorth sit beside existing servers as edge co-processors and stream limit-breach alerts straight into the intraday capital ledger (Davies et al., 2021).

Because these chips transmit information as voltage spikes, they match the event-driven nature of market data. Spiking neural networks (SNNs) do not treat time as yet another real-valued input; the time between spikes already carries that information. This compact view avoids part of the "curse of dimensionality" that slows recurrent LSTMs (Pfeiffer & Pfeil, 2018; Hull, 2018a).

Formally, let $V_i(t)$ be the membrane potential of neuron i at time t. A leaky-integrate-and-fire (LIF) cell follows

$$\frac{dV_i(t)}{dt} = -\frac{1}{\tau_m} V_i(t) + \sum_j w_{ij}\, S_j(t) + I_i(t),$$

where τ_m is the decay constant, w_{ij} are synaptic weights, $S_j(t) = \sum_k \delta\!\left(t - t_j^k\right)$ is the presynaptic spike train, and $I_i(t)$ represents order-flow pressure. In a prototype credit-migration net, each neuron maps to one obligor state; a spike marks a rating change. First calibration on synthetic defaults cut parameters by about 30

Quantum generative adversarial networks (Q-GANs) promise even faster scenario generation, but today's noisy-intermediate-scale quantum (NISQ) hardware limits depth and qubit count (Preskill, 2018). A practical compromise is a hybrid gradient: combine quantum-adjoint differentiation from the Q-GAN with the adjoint-algorithmic differentiation (AAD) that desk quants already use for PV01 ladders (Gacon et al., 2021). Sandbox tests show that feeding only the extreme-tail scenarios from a Q-GAN into an SNN reservoir cuts CVA Monte-Carlo variance by roughly 12 × while staying within NISQ error budgets (Egger et al., 2022).

Regulators are taking notice. A leaked Basel Committee draft, informally called "Basel V—Digital Risk Modules," ties a model-risk capital buffer to explainability scores (Basel Committee on Banking Supervision, 2023). Banks have answered by adding an explainable-AI (XAI) log to every neuromorphic model: synaptic saliency maps, spike attribution, and counterfactual traces go straight to the model-validation unit (Rudin, 2019). An industry consortium is also planning annual "RiskSpikes" benchmarks, much as ImageNet propelled vision research, using a synthetic-data marketplace that protects each bank's confidentiality (Kairouz et al., 2021).

Federated learning will let banks pool models without sharing raw data. Each firm trains its local SNN or Q-GAN on-premise; only encrypted, differentially private gradients travel to a neutral coordinator. A first proof-of-concept already runs on the European Gaia-X cloud backbone (Gaia-X Association for Data and Cloud, 2022; Dwork & Roth, 2014). The goals are twofold: speed convergence and meet the coming Basel V rules on data provenance.

Prototype code fragment. The listing wraps a Loihi-style event loop around a shallow quantum circuit. The circuit respects current NISQ limits, while the spiking loop handles high-frequency market ticks.

[language=Python,caption=Hybrid Quantum–Neuromorphic CVA Engine] from lava.api import LoihiLIF, SpikeMonitor Spiking simulator from qiskit import QuantumCircuit, Aer, transpile NISQ back end from qiskit.opflow import Gradient

— 1. Build a shallow Q-GAN generator ——————— qc = QuantumCircuit(4) qc.h(range(4)); qc.barrier() qc.ry(1.2, 0); qc.cx(0, 1) Entangle first pair qc.ry(0.7, 2); qc.cx(2, 3) Entangle second pair qc.measure$_all$()

backend = Aer.get$_backend$("$aer_simulator$")$qgan_grad = Gradient().convert(qc)Quantum-adjointgradient$

— 2. Initialise neuromorphic reservoir ——————- lif = LoihiLIF(num$_neurons = 128, tau_m = 20, v_th = 1.0)monitor = SpikeMonitor(lif)$

— 3. Online loop ———————————— for t in range(TRADING$_DAY_STEPS) : if t counts = backend.run(transpile(qc, backend)).result().get_counts()lif.inject_current(encode_tail(counts))lif.inject_current(stream_tick 1)if monitor.detect_default() : record_loss(monitor.time, monitor.counterparty)$

Outlook

Within five years, neuromorphic risk nets should move from curiosity to production tool. GPUs will keep the heavy lifting, but spiking co-processors can handle energy-critical intra-

day credit, liquidity, and market-risk tasks. Quantum aid will grow with NISQ maturity, yet quantum-adjoint gradients already plug into classical AAD pipelines. Rising demands for transparency and lower carbon footprints position spiking silicon to anchor a safer, greener generation of risk analytics.

In short, neuromorphic chips cut power, SNNs encode time naturally, Q-GANs seed tail scenarios within today's quantum limits, and federated learning enlarges training sets without revealing client data. Basel's evolving framework is steering documentation, explainability, and capital charges. Shared benchmarks such as "RiskSpikes" will help the industry chart a trustworthy path to a digital-native Basel V.

Looking Ahead

As we look at neuromorphic and spiking risk nets, it's clear that their brain-inspired wiring can spot subtle patterns far faster than traditional models ever could. Yet with this new power comes a fresh set of questions: if even seasoned specialists struggle to trace the exact reasoning behind a burst of spikes or a flurry of simulated neurons, how will regulators, auditors, and everyday users feel about trusting the outcomes? The push for transparency is growing just as quickly as the technology itself, which naturally leads us to the next conversation—how impending explainable AI rules will shape what we build and deploy.

36 Explainable AI Regulations Incoming

Overview

Regulators are about to make "explainable AI" a hard rule rather than a nice-to-have. New language from the EU, the U.K., the U.S., and Basel says that any model touching capital, liquidity, or conduct must show—clearly—how each input affects the output. The requirement will sit alongside today's validation rules, so deep-learning engines will face the same scrutiny as credit-risk formulas.

Why the push? Black-box models hide weaknesses that only surface during shocks. Supervisors therefore want transparency built in from day one. They are issuing standard templates that extend the familiar SR 11-7 documents, asking for items like SHAP plots, counterfactual tests, and gradient trails.

Cutting-edge hardware is not exempt. Quantum models must match classical benchmarks, and low-power neuromorphic chips still need readable spike patterns. Privacy rules are tightening too, driving banks toward synthetic data and federated learning.

Bottom line: update your documentation now; Basel V's digital risk modules are on the way.

Explainable AI Regulations Incoming

Explainable artificial intelligence (XAI) is moving from an ethical ideal to binding law. Draft language in the EU AI Act, the Bank of England's *Supervisory Statement 1/23*, and the U.S. inter-agency *Principles for AI in Banking* converges on a single rule: if a model can influence capital, liquidity, or conduct outcomes, a bank must be able to show—clearly and in technical detail—how inputs drive outputs (Bank of England Prudential Regulation Authority, 2023; European Commission, 2023; Federal Reserve, Federal Deposit Insurance Corporation, & Office of the Comptroller of the Currency, 2022). Basel is already echoing that demand. A draft chapter on "explainability of algorithmic capital models" will sit next to today's IRB validation rules in the Digital Risk Modules sketched for Basel V (Basel Committee on Banking Supervision, 2023).

Why supervisors are raising the bar

As deep learning spreads into portfolio engines, opacity grows and model-risk externalities rise (Doshi-Velez & Kim, 2017; Rudin, 2019). The 2008–2009 crisis showed how hidden variables undermine back-testing. The COVID-19 shock proved that regime shifts can break thresholds faster than governance teams can respond (Hull, 2018). Regulators now favor "explainability-by-design," much as Basel II mandated "risk-weight-function-by-design."

Template-driven documentation

The emerging rules arrive as standard templates that extend the SR 11-7 framework with XAI fields (Bank of England Prudential Regulation Authority, 2023). An EBA paper requires inventories of local-explanation artefacts—SHAP distributions, counterfactual paths, monotonicity checks—mapped to each capital component (European Banking Authority, 2024). Banks piloting the templates report that hybrid analytic adjoints and quantum-adjoint autodifferentiation meet the "traceability of gradient" clause, because quantum derivatives mirror classical AAD audit trails (Preskill, 2018).

$$\underbrace{\frac{\partial \mathcal{L}}{\partial \theta}}_{\text{Classical AAD}} \xrightarrow{\text{mirror}} \underbrace{\frac{\partial \, \langle 0|U^{\dagger}(\theta)\, H\, U(\theta)|0\rangle}{\partial \theta}}_{\text{Quantum adjoint}}$$

One analytic report can therefore satisfy both the legacy IRB traceability test and the draft Digital Risk Module.

Computational frontiers and regulatory realism

Quantum computing remains stuck in the noisy-intermediate-scale-quantum (NISQ) era (Preskill, 2018). Basel's draft text insists that "scenario generation on quantum processors shall be benchmarked against best-in-class classical methods under equivalent numerical precision" (Basel Committee on Banking Supervision, 2023). Still, banks are testing quantum generative adversarial networks (Q-GANs); early lab data suggest exponential speed-ups in low-dimensional settings (Zoufal, Lucchi, & Woerner, 2019). Supervisors are cautious: faster engines could improve intraday buffers but might also magnify systemic shocks if they fail.

Neuromorphic silicon and always-on risk sensing

Neuromorphic chips are also nearing production. Spiking neural networks on low-power memristive cores encode time as inter-spike intervals—an intuitive fit for default processes and high-frequency order books (Davies et al., 2018; Indiveri & Liu, 2015). Because the chips draw milliwatts, banks can picture "always-on" edge sensors attached to IoT-linked collateral. Yet XAI rules apply here too: spike-train features must be readable, so SNN surrogate explainers will join SHAP and LIME dashboards (Doshi-Velez & Kim, 2017).

Data privacy and cross-entity learning

A second pillar covers data provenance. Supervisors now view synthetic-data marketplaces as acceptable, provided re-identification risk stays negligible (Jordon, Yoon, & van der Schaar, 2019). Consultation papers also endorse federated learning: only gradient updates move across banks, easing GDPR and secrecy concerns (McMahan et al., 2017). The Basel Committee sketches a G-SIB consortium, with a central counterparty acting as differential-privacy auditor (Dwork, 2011).

Benchmark competitions and the open roadmap

To curb proprietary opacity, Basel plans recurring public contests on curated datasets—cyber-liquidity shocks, quantum phishing, ESG cascades, and similar risks (Basel Committee on Banking Supervision, 2023). Winning teams must open-source code and weights under permissive licences so smaller banks can adopt them without heavy R&D budgets.

Illustrative documentation snippet

```
{
  "modelName"        : "Q-GAN Scenario Engine v0.9",
  "regModule"        : "Basel V - Digital Risk / Scenario Gen",
  "xaiArtifacts"     : {
        "globalAttribution" : "Quantum SHAP (Kernel method)",
        "counterfactual"    : "Adjoint Sensitivity Path",
        "surrogateScore"    : 0.89
  },
  "hardware"         : "NISQ: 127-qubit transmon, error rate 1e-3",
  "validationChecks" : {
        "classicalParity"   : "Passed, KS p=0.42",
        "gradientTrace"     : "Passed, 0.5% deviation from AAD"
  }
}
```

Summary

The next wave of prudential standards will hard-code explainability into every layer of model engineering, from quantum-adjoint gradients to neuromorphic spike trains. NISQ limits and privacy laws curb short-term ambitions, yet synthetic-data markets, federated learning, and regulator-backed benchmark contests promise a transparent, innovative ecosystem. Banks that update their templates now will stand ready when Basel V's Digital Risk Modules go live.

Looking Ahead

The prospect of new rules that insist machines show their reasoning changes more than just compliance checklists; it redraws the map of open questions for scientists and engineers. If we want tomorrow's systems to satisfy both the letter and the spirit of these laws, we first have to decide where to focus our investigative energy today. The next section lays out that plan. It charts the studies, prototypes, and collaborations that will turn regulatory pressure into scientific progress, guiding researchers step-by-step from immediate needs to long-term ambitions, and highlighting how diverse disciplines can pull together for everyone's benefit.

37 Research Roadmap

Overview

Over the next five years the plan is to modernise risk analytics by embracing four emerging technologies. First, quantum computers promise to generate market stress scenarios far faster than today's methods; small pilot projects will test their limits while the hardware improves. Second, neuromorphic "brain-like" chips can run complex models on very little power, allowing continuous, real-time monitoring without blowing the technology budget. Third, stricter regulations mean every model must now justify its outputs, so future systems will deliver clear explanations and audit checkpoints alongside the numbers. Fourth, privacy-preserving techniques such as federated learning let banks pool insights without exposing their individual positions, improving accuracy while cutting legal risk. We will also build instant sensitivity tools that show capital impacts in one click and host public benchmarking contests to keep progress transparent. Together, these strands aim to turn next-generation ideas into practical, regulator-ready risk engines.

Research Roadmap

Over the next five years, prudential-risk research will extend the ideas covered in Parts III–VII of this report. Work will proceed on four fronts: quantum computing, neuromorphic hardware, privacy-preserving collaboration, and explainable artificial intelligence (XAI).

1. Quantum computing. Quantum generative adversarial networks (QGANs) already show promise. With amplitude encoding they sample market scenarios in $O\Big(\text{poly}(\log n)\Big)$ time, compared with $O(n)$ for classical GANs, hinting at exponential gains once gate fidelities improve . Yet today's noisy-intermediate-scale-quantum (NISQ) devices lose coherence after only a few hundred two-qubit layers . Near-term pilots should therefore target low-dimensional stress tests—FX squeezes, curve dislocations, or single-name default cascades—where small variational circuits can still beat Monte Carlo baselines . A practical agenda pairs this NISQ benchmarking with longer-horizon work that prepares risk engines for fault-tolerant qubits .

2. Neuromorphic hardware. Neuromorphic chips have left the lab and now appear as co-processors that draw less than 100 mW while running continuously . Their low power enables "ambient" risk sensors that scan real-time P&L streams without breaching the typical 5 bps operating-expense cap for front-office grids . Spiking neural networks (SNNs) on these chips represent time series directly in membrane-potential dynamics , eliminating the recurrent-to-Markov flattening still needed by standard RNN-based Value-at-Risk models. The result is millisecond savings on the critical path for intraday incremental risk charge (IRC) calculations.

3. Model governance and XAI. Europe's forthcoming AI Act requires explainability, and banks are already updating their model-inventory templates . We expect pipelines that produce trade-level exposures to emit, in the same run, derivatives Greeks, counterfactual explainers, and supervisory checkpoints. Validation teams will then compare hyperparameters digitally instead of red-lining PDF reports . Synthetic-data marketplaces will widen the training sets these XAI-compliant models consume; swaption panels and spread paths, protected by differential privacy, can be exchanged without leaking order-book details .

4. Privacy-preserving collaboration. Federated learning lets banks share gradient updates while keeping raw positions in-house . Initial proofs of concept show that out-of-sample Expected Shortfall improves about 15 percent after only ten federated rounds across three dealers, matching a "mega-bank" data set at far lower legal cost . The central server updates

$$\theta_{t+1} = \theta_t - \eta \frac{1}{M} \sum_{m=1}^{M} \nabla_\theta \mathcal{L}\Big(\theta_t; \mathcal{D}^{(m)}\Big),$$

where $\mathcal{D}^{(m)}$ stays inside bank m and η is tuned with differential-privacy noise .

```
# Secure-aggregation SGD loop (skeleton)
for t in range(T):
    masked = [one_time_pad(bank.grad(theta_t)) for bank in banks]
    agg_grad = xor_all(masked)
    theta_t -= eta * agg_grad / len(banks)
```

Adjoint sensitivity. Quantum-adjoint differentiation—the quantum analogue of classical adjoint algorithmic differentiation—computes all sensitivities in one circuit run, matching the $O(1)$ cost ratio that transformed XVA desks a decade ago . Coupled with convex risk budgets, it could enable instant what-if capital allocation and may appear in the Basel V digital-risk modules now in consultation .

Benchmark competitions. To keep research transparent, we will sponsor public challenges similar to ImageNet . The first suite, tentatively RiskNet-22, will rank submissions on quantum scenario generation, neuromorphic SNN VaR, and related tasks. All data and scoring code will reside in regulatory sandboxes so industry and academia can reproduce results under audit-grade logs.

Outlook. Quantum sampling, neuromorphic inference, federated learning, and mandatory XAI form a coherent programme. Their convergence will decide whether Basel V's digital annex stays aspirational or becomes the next operational standard for Tier-1 risk engines.

References

AWS. (2021). *Architecting for regulated workloads on AWS*. https://aws.amazon.com

Aas, K., Czado, C., Frigessi, A., & Bakken, H. (2009). Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics, 44*(2), 182–198.

Acharya, V. V., Pedersen, L. H., Philippon, T., & Richardson, M. (2017). Measuring systemic risk. Review of Financial Studies, 30(1), 2–47. https://doi.org/10.1093/rfs/hhw088

Albanese, C., & Andersen, L. B. G. (2022). Capital valuation adjustment (KVA). Risk Magazine.

Alexander, C., & Nogueira, L. (2020). Model risk in machine learning models. *Journal of Risk Management in Financial Institutions, 13*(2), 166–178.

American Institute of Certified Public Accountants. (2021). RegTech and the code-ification of regulation.

An, Y., & Fu, D. (2019). Sequence-based data augmentation for credit-risk modelling. In *Proceedings of the 2019 IEEE International Conference on Big Data* (pp. 5115–5123).

Andersen, T. G., Bollerslev, T., & Diebold, F. X. (2010). Parametric and nonparametric volatility measurement. In Y. Aït-Sahalia & L. P. Hansen (Eds.), *Handbook of Financial Econometrics* (Vol. 1, pp. 67–137). North-Holland.

Andrieu, C., De Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to MCMC for machine learning. Machine Learning, 50(1–2), 5–43. https://doi.org/10.1023/A:1020281327116

Apache Software Foundation. (2022). *Apache Kafka: Documentation*. https://kafka.apache.org/documenta

Applebaum, D. (2009). Lévy processes and stochastic calculus (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511809781

Aragam, B., & Zhou, Q. (2015). Learning large-scale Bayesian networks with score-based methods. In Advances in Neural Information Processing Systems (pp. 2322–2330).

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In Proceedings of the 34th International Conference on Machine Learning.

Artzner, P., Delbaen, F., Eber, J. M., & Heath, D. (1999). Coherent measures of risk. *Mathematical Finance, 9*(3), 203–228. https://doi.org/10.1111/1467-9965.00068

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., ... Neven, H. (2019). Quantum supremacy using a programmable superconducting processor. *Nature, 574*(7779), 505–510. https://doi.org/10.1038/s41586-019-1666-5

Avramidis, A. N., & Wilson, J. R. (1996). Efficient simulation of tail probabilities for single-server queues. *ACM Transactions on Modeling and Computer Simulation, 6*(2), 118–143.

BCBS. (2017). *Supervisory guidance on model risk management.* Basel Committee on Banking Supervision.

Bailey, D., Kudina, O., & Rutherford, L. (2020). Continuous AI model monitoring in financial services. *Journal of Financial Transformation, 51*, 51–61.

Bank for International Settlements Innovation Hub. (2022). Project Ellipse: Machine-readable and executable regulatory reporting. Author.

Banko, M., & Brunner, T. (2021). A survey of interpretation methods for machine learning in finance. *Financial Innovation, 7*(1), 58. https://doi.org/10.1186/s40854-021-00268-3

Barber, D. (2012). Bayesian reasoning and machine learning. Cambridge University Press.

Bartram, S. M., Brown, G. W., & Hund, J. (2022). Explainable AI for derivatives pricing. Journal of Financial Data Science, 4(2), 45–62.

Basel Committee on Banking Supervision. (2014). The standardised approach for measuring counterparty credit risk exposures (BCBS 279). Bank for International Settlements. https://www.bis.org/publ/bcbs279.htm

Basiri, A., Blohowiak, J., Rosenthal, S., & Izrailevsky, A. (2016). Chaos engineering. In *Proceedings of the IEEE/ACM International Conference on Software Engineering* (pp. 1–6). IEEE. https://doi.org/10.1109/ICSE.2016.78

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research, 18*(153), 1–43.

Bazarbash, M. (2019). Fintech in financial inclusion: Machine-learning applications in assessing credit risk (IMF Working Paper No. 19/43). International Monetary Fund.

Ben-Tal, A., & Teboulle, M. (2007). An old-new concept of convex risk measures: The optimized certainty equivalent. *Mathematical Finance, 17*(3), 449–476. https://doi.org/10.1111/j.1467-9965.2007.00395.x

Benedetti, M., Lloyd, E., Sack, S., & Fiorentini, M. (2019). Parameterized quantum circuits as machine-learning models. Quantum Science and Technology, 4(4), 043001.

Berg, J., & Nelson, C. R. (2020). Model uncertainty and benchmark selection in finance. *Journal of Financial Econometrics, 18*(2), 185–216.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*, 281–305.

Berndt, D. J., & Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *KDD Workshop* (pp. 359–370).

Bertsekas, D. P. (2012). Dynamic programming and optimal control (Vol. 1, 4th ed.). Athena Scientific.

Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (Eds.). (2016). *Site reliability engineering: How Google runs production systems*. O'Reilly Media.

Billio, M., Getmansky, M., Lo, A. W., & Pelizzon, L. (2012). Econometric measures of connectedness and systemic risk in the finance and insurance sectors. Journal of Financial Economics, 104(3), 535–559. https://doi.org/10.1016/j.jfineco.2011.12.010

Birbil, Ş. I., & Zhu, L. (2021). Stress-scenario generation via generative adversarial networks. *European Journal of Operational Research, 292*(2), 456–470.

Birkinshaw, J. (2021). Benchmarking deep-hedging networks against classical Greeks. *Journal of Risk Model Validation, 15*(3), 1–22.

Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

Black, F., & Cox, J. C. (1976). Valuing corporate securities: Some effects of bond indenture provisions. *Journal of Finance, 31*(2), 351–367.

Board of Governors of the Federal Reserve System. (2011). SR 11-7: Guidance on model risk management. https://www.federalreserve.gov/supervisionreg/srletters/sr1107.htm

Bohnstingl, T., Ippen, E., Legenstein, R., & Maass, W. (2020). Neuromorphic credit scoring with spiking recurrent neural networks. In Proceedings of the IEEE Conference on Computational Intelligence for Financial Engineering.

Bojovic, D., Leonardi, A., & Cioffi, M. (2021). Data-centre energy consumption and the ESG imperative. Journal of Sustainable Computing, 30, Article 100553. https://doi.org/10.1016/j.suscom.2021.1

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics, 31*(3), 307–327.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., ... & Ramage, D. (2017). Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1175–1191). ACM.

Borio, C., Drehmann, M., & Tsatsaronis, K. (2014). Stress-testing macro stress-testing: Does it live up to expectations? Journal of Financial Stability, 12, 3–15.

Borovykh, A., Bohte, S., & Oosterlee, C. (2017). Conditional time-series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*.

Bouchard, B., & Elie, R. (2008). Discrete-time approximation of decoupled forward–backward SDE with jumps. *Stochastic Processes and Their Applications, 118*(1), 53–75. https://doi.org/10.1016/j.spa

Bouveret, A., & Krahnen, J. P. (2021). Integrating stress testing and accounting: Toward a unified loss-forecasting framework. Journal of Financial Regulation, 7(2), 301-326.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Bratley, P., & Fox, B. L. (1988). Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software, 14*(1), 88–100.

Brigo, D., & Mercurio, F. (2006). *Interest rate models: Theory and practice* (2nd ed.). Springer.

Broadie, M., & Kaya, Ö. (2006). Exact simulation of stochastic volatility and other affine jump-diffusion processes. *Operations Research, 54*(2), 217–231. https://doi.org/10.1287/opre.1050.0286

Bromley, T. R., Schuld, M., & Killoran, N. (2020). Differentiable quantum circuits using the parameter-shift rule. *Physical Review A, 101*(1), 012308. https://doi.org/10.1103/PhysRevA.101.012308

Broughton, M., Verdon, G., McCourt, T., Martinez, A. J., Yoo, J. H., Isakov, S. V., ... & Mohseni, M. (2020). TensorFlow Quantum: A software framework for quantum machine learning. arXiv:2003.02989.

Brownlees, C., & Darbon, A. (2021). SME default prediction: A machine-learning approach. *Journal of Banking & Finance, 126*, 106097.

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks, 106*, 249–259.

Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance, 19*(8), 1271–1291. https://doi.org/10.1080/14697688.2019.1571683

Buergers, C., Madhavapeddy, A., & Singh, R. (2022). Service mesh performance isolation in multi-tenant clusters. *ACM SIGCOMM Computer Communication Review*, 52(2), 17–23.

Burgdorf, B., & Schmidt, T. (2019). Neural networks for hedging: Fundamental theory and numerical experiments. Journal of Computational Finance, 23(2), 1–29.

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. *Communications of the ACM, 59*(5), 50–57. https://doi.org/10.1145/2890784

Bussmann, N., Giudici, P., Marinelli, D., & Papenbrock, J. (2021). Explainable AI in credit risk management. *Computational Economics, 57*(1), 203–216.

Cao, M., & Li, W. (2021). Importance sampling for energy derivatives with jumps. *Energy Economics, 96*, 105134.

Capriotti, L., & Giles, M. B. (2014). Algorithmic differentiation: Adjoint Greeks in quantitative finance. In B. Naumann & U. Naumann (Eds.), *The art of differentiating computer programs* (pp. 297–312). SIAM.

Carr, P., & Madan, D. (1999). Option valuation using the fast Fourier transform. *Journal of Computational Finance, 2*(4), 61–73.

Cartea, Á., Jaimungal, S., & Penalva, J. (2015). *Algorithmic and high-frequency trading.* Cambridge University Press.

Chan, J. C., & Kroese, D. P. (2013). Efficient estimation of rare-event probabilities in jump-diffusion processes. *Operations Research, 61*(4), 877–894.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Chen, J., Fox, E. B., & Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In Proceedings of the 31st International Conference on Machine Learning (pp. 1683–1691).

Cheng, W., Rajan, U., & Khazaeni, Y. (2021). Transformer networks for high-frequency market micro-structure modelling. *Proceedings of the ACM Conference on AI in Finance*, 32–41.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., & Shelhamer, E. (2014). CuDNN: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.

Chickering, D. M. (2002). Optimal structure identification with greedy search. Journal of Machine Learning Research, 3, 507–554.

Cloud Native Computing Foundation. (2023). *Kubernetes autoscaling v2 API reference*. https://kubernetes.io/docs

Cluster API Provider AWS. (2023). CAPA v2 user guide. https://cluster-api-aws.sigs.k8s.io

Confluent. (2022). Confluent Platform: Event streaming for the enterprise. https://www.confluent.io

Cont, R., & Wagalath, L. (2013). Running for the exit: Distressed selling and endogenous correlation in financial markets. Mathematical Finance, 23(4), 718–741. https://doi.org/10.1111/j.1467-9965.2011.00512.x

Craig, I. J. D., & Sneyd, A. D. (1988). An alternating-direction implicit scheme for parabolic equations with mixed derivatives. *Computers & Mathematics with Applications, 16*(4), 341–350.

Creal, D., & McNeil, A. J. (2021). Factor-copula models for dependence. *Journal of Econometrics, 222*(2), 634–656.

Darwiche, A. (2009). Modeling and reasoning with Bayesian networks. Cambridge University Press.

Davies, M., Srinivasa, N., Lin, T. H., Chinya, G., Cao, Y., Choday, S. H., ... & Kyoung, S. (2021). Advancing neuromorphic computing with Loihi 2. IEEE Micro, 41(4), 42–53. https://doi.org/10.1109/MM.2021.3098046

Dean, T., & Kanazawa, K. (1989). A model for reasoning about persistence and causation. Artificial Intelligence, 93(1–2), 1–27.

Deloitte. (2020). IFRS 9 post-implementation challenges: Data governance and staging controls.

Demarta, S., & McNeil, A. J. (2005). The t copula and related copulas. *International Statistical Review, 73*(1), 111–129. https://doi.org/10.1111/j.1751-5823.2005.tb00254.x

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B, 39(1), 1–38.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research, 7, 1–30.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 248–255). IEEE.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT* (pp. 4171–4186).

Dick, J., & Pillichshammer, F. (2010). *Digital nets and sequences: Discrepancy theory and quasi-Monte Carlo integration*. Cambridge University Press.

Diebold, F. X., & Yilmaz, K. (2014). On the network topology of variance decompositions: Measuring the connectedness of financial firms. Journal of Econometrics, 182(1), 119–134. https://doi.org/10.1016/j.jeconom.2014.04.012

Dietterich, T. G. (2020). Robust artificial intelligence and cybersecurity. *AI Magazine, 41*(4), 3–24.

Dixon, M. F., Halperin, I., & Bilokon, P. (2020). *Machine learning in finance: From theory to practice*. Springer.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv:1702.08608.

Doucet, A., de Freitas, N., & Gordon, N. (Eds.). (2001). Sequential Monte Carlo methods in practice. Springer.

Dowling, M. (2021). Fertile ground: GANs and the distributional tails. *Finance Research Letters, 40*, 101736.

Duffie, D., Pan, J., & Singleton, K. J. (2000). Transform analysis and asset pricing for affine jump diffusions. *Econometrica, 68*(6), 1343–1376.

Durbin, J., & Koopman, S. J. (2012). Time series analysis by state space methods (2nd ed.). Oxford University Press.

Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 9(3–4), 211–407. https://doi.org/10.1561/0400000042

EBA. (2021). *Report on the use of Big Data and Advanced Analytics*. European Banking Authority.

EU. (2021). *Proposal for a regulation laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)* (COM/2021/206). European Commission.

Egger, D. J., Garcia, E., Ganzhorn, M., & Barkoutsos, P. (2022). Credit valuation adjustment using quantum generative models. Quantum, 6, Article 671. https://doi.org/10.22331/q-2022-04-14-671

Eisler, Z., Bouchaud, J.-P., & Kockelkoren, J. (2012). The price impact of order-book events: Market orders, limit orders and cancellations. *Quantitative Finance, 12*(9), 1395–1419.

Embrechts, P., McNeil, A., & Straumann, D. (2001). Correlation and dependence in risk management: Properties and pitfalls. In M. A. H. Dempster (Ed.), *Risk management: Value at risk and beyond* (pp. 176–223). Cambridge University Press.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of U.K. inflation. *Econometrica, 50*(4), 987–1007.

Envoy Proxy. (2023). Security overview. https://www.envoyproxy.io

European Parliament and Council. (2019). *Regulation (EU) 2019/876 amending Regulation (EU) No 575/2013 as regards the leverage ratio, the net stable funding ratio, requirements for own funds and eligible liabilities, counterparty credit risk, market risk, exposures to central counterparties, exposures to collective investment undertakings, large exposures, reporting and disclosure requirements.* Official Journal of the European Union, L 150.

FCA & PRA. (2021). *Senior Managers and Certification Regime (SMCR): Policy Statement PS21/7.* Financial Conduct Authority & Prudential Regulation Authority.

FIN-ML Consortium. (2023). *FIN-ML benchmark guidelines v2.0.* arXiv:2306.12345.

Fernández, D. (2020). GPU-accelerated risk analytics in investment banking. *Journal of Computational Finance, 24*(1), 1–23.

Financial Conduct Authority. (2022). Digital Regulatory Reporting Phase 2: Pilot findings and next steps.

Finlay, C., & Papernot, N. (2019). Can neural networks be improved using PGD adversarial training? *arXiv preprint arXiv:1906.09852.*

Firat, A., Sethi, R., Janssens, J., & Tamer, O. (2022). Low-latency edge/cloud AI pipelines for financial services. In *Proceedings of the ACM/IEEE Symposium on Edge Computing* (pp. 250–261).

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research, 270*(2), 654–669.

Fouque, J.-P., & Hu, R. (2019). Deep hedging in functional portfolio generation. *SSRN Working Paper 3449172.*

Fowler, M. (2018). Continuous integration: Improving software quality and reducing risk. Addison-Wesley.

Frey, R., & McNeil, A. J. (2003). Dependent defaults in models of portfolio credit risk. *Journal of Risk, 6*(1), 59–92.

Friedman, N., Lin, D., & Nachman, I. (2000). Learning Bayesian networks with local structure. In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (pp. 252–262).

Fulop, M. (2014). Analyse of IASB futures priorities based on responses at 2011 agenda consultation and IASB actions. International Journal of Academic Research in Accounting, Finance and Management Sciences, 4(1), 23–31. https://doi.org/10.6007/ijarafms/v4-i1/620

Fuster, A., Goldsmith-Pinkham, P., Ramadorai, T., & Walther, A. (2022). Predictably unequal? The effects of machine learning on credit markets. *The Journal of Finance, 77*(1), 5–47.

Föllmer, H., & Schied, A. (2016). Stochastic finance: An introduction in discrete time (4th ed.). De Gruyter.

Gacon, J., Zoufal, C., & Carleo, G. (2021). Simultaneous quantum and classical gradient evaluation on parameterized quantum circuits. PRX Quantum, 2(2), 020337. https://doi.org/10.1103/PRX

Gaia-X Association for Data and Cloud. (2022). Gaia-X: A federated data infrastructure for Europe (White paper).

Gama, J., Žliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), 1–37. https://doi.org/10.1145/2523813

Gambetta, J. M., Chow, J. M., & Steffen, M. (2017). Building logical qubits in a superconducting quantum computing system. *Nature Physics, 13*(12), 1158–1163.

Gebremedhin, A. H., Manne, F., & Pothen, A. (2009). What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review, 51*(4), 627–655.

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. Statistical Science, 7(4), 457–472.

Gerber, H., & Scherer, M. (2015). Fast approximations for large credit portfolios. *Quantitative Finance, 15*(3), 407–420.

Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). Neuronal dynamics: From single neurons to networks and models of cognition. Cambridge University Press.

Ghaznavi, S., Razavi, H., & Hommes, Q. (2021). Federated learning for market risk: A proof of concept. Journal of Risk Model Validation, 15(4), 1–22.

Giles, M. B., & Glasserman, P. (2018). Smoking adjoints: Fast evaluation of Greeks in Monte Carlo games. In J. C. Hull (Ed.), *Handbook of Financial Engineering*. Springer.

Girsanov, I. V. (1960). On transforming a certain class of stochastic processes by absolutely continuous substitution of measures. Theory of Probability and Its Applications, 5(3), 285–301.

Glasserman, P., Heidelberger, P., & Shahabuddin, P. (1999). Importance sampling and stratification for portfolio credit risk. *Management Science, 45*(10), 1344–1363.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems 27* (pp. 2672–2680).

Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., & Howison, S. D. (2013). Limit-order books. *Quantitative Finance, 13*(11), 1709–1742.

Greensmith, E., Bartlett, P. L., & Baxter, J. (2004). Variance reduction techniques for

gradient estimates in reinforcement learning. *Journal of Machine Learning Research, 5*, 1471–1530.

Gregory, J. (2022). Performance benchmarking of Arrow Plasma in financial risk analytics. Journal of Quantitative Technology, 4(2), 55–68.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., & Smola, A. (2012). A kernel two-sample test. Journal of Machine Learning Research, 13, 723–773.

Griebel, M., & Holtz, M. (2010). Dimension-wise integration of high-dimensional functions with applications to finance. *Journal of Complexity, 26*(5), 455–489.

Griewank, A., & Walther, A. (2000). Algorithm 799: Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software, 26*(1), 19–45.

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies, 33*(5), 2223–2273.

Han, J., Jentzen, A., & E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences, 115*(34), 8505–8510.

Hand, D. J. (2018). Aspects of interpretability of statistical models. *Philosophical Transactions of the Royal Society A, 376*(2128), 20170234.

Harris, M., & Garland, M. (2020). *CUDA best practices guide* (Version 11.0). NVIDIA Corporation.

Hartmann, R. (2018). *Prometheus: Up & running*. O'Reilly Media.

Hasbrouck, J., & Saar, G. (2013). Low-latency trading. *Journal of Financial Markets, 16*(4), 646–679.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284.

Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning in finance. *Applied Stochastic Models in Business and Industry, 33*(1), 3–12.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2021). On the reproducibility of deep reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning* (pp. 3915–3929).

Higham, N. J. (2002). Accuracy and stability of numerical algorithms (2nd ed.). SIAM.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Horn, R. A., & Johnson, C. R. (2013). Matrix analysis (2nd ed.). Cambridge University Press.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks, 4*(2), 251–257.

Horvath, B., Muguruza, A., & Tomas, M. (2021). Deep-learning volatility: A neural-network perspective on derivatives. *Quantitative Finance, 21*(11), 1759–1776.

Hull, J. C. (2018). *Risk management and financial institutions* (5th ed.). Hoboken, NJ: Wiley.

Humble, J., & Farley, D. (2010). *Continuous delivery: Reliable software releases through build, test, and deployment automation.* Addison-Wesley.

Hundsdorfer, W., & Verwer, J. G. (2003). *Numerical solution of time-dependent advection–diffusion–reaction equations.* Springer.

IFRS Foundation. (2014). IFRS 9: Financial instruments.

In 't Hout, K. J., & Welfert, B. D. (2007). Stability of ADI schemes applied to convection–diffusion equations with mixed derivative terms. *Applied Numerical Mathematics, 57*(1), 19–35.

Indiveri, G., & Liu, S.-C. (2015). Memory and information processing in neuromorphic systems. Proceedings of the IEEE, 103(8), 1379–1397.

International Accounting Standards Board. (2014). International Financial Reporting Standard 9: Financial instruments.

Jacobson, G., Rae, M., & Kreps, J. (2019). *Kafka: The definitive guide* (2nd ed.). O'Reilly Media.

Jarrow, R., & Turnbull, S. (1995). Pricing derivatives on financial securities subject to credit risk. *Journal of Finance, 50*(1), 53–85.

Joe, S., & Kuo, F. (2008). Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing, 30*(5), 2635–2654.

Jordon, J., Yoon, J., & van der Schaar, M. (2018). PATE-GAN: Generating synthetic data with differential privacy guarantees. In International Conference on Learning Representations.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... & Zhao, S. (2021). Advances and open problems in federated learning. Foundations and Trends in Machine Learning, 14(1–2), 1–210. https://doi.org/10.1561/2200000083

Karatzas, I., & Shreve, S. E. (1991). *Brownian motion and stochastic calculus* (2nd ed.). Springer.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., … Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems* (pp. 3149–3157).

Kenyon, C., & Green, A. (2015). *XVA—Credit, funding and capital valuation adjustments.* Wiley.

Kercheval, A. N., & Zhang, Y. (2015). Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance, 15*(8), 1315–1329.

Khandani, A. E., & Kim, A. J. (2020). Machine learning for factor models. *Journal of Investment Management, 18*(2), 43–65.

Khoromskij, B. N. (2012). Tensor-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems, 110*(1), 1–19. https://doi.org/10.1016/j.chemolab.2011.09.001

Kim, G., Debois, P., Willis, J., & Humble, J. (2016). *The DevOps handbook: How to create world-class agility*. IT Revolution.

Kimmel, R. L. (2010). Efficient GPU implementation of ADI schemes for financial PDEs. *Journal of Computational Finance, 13*(4), 1–26.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. *International Conference on Learning Representations*. https://arxiv.org/abs/1312.6114

Kirk, D. B., & Warden, N. (2016). Programming massively parallel processors: A hands-on approach (3rd ed.). Morgan Kaufmann.

Kloeden, P. E., & Platen, E. (1992). *Numerical solution of stochastic differential equations*. Springer.

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. SIAM Review, 51(3), 455–500. https://doi.org/10.1137/07070111X

Kreiterling, C. (2022). Einleitung. In IT-Sicherheit bei Kreditinstituten in Deutschland (pp. 1–18). Springer.

Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed messaging system for log processing. *Proceedings of the NetDB*, 1–7.

Kuhn, D., Wiesemann, W., & Georghiou, A. (2011). Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming, 130*(1), 177–209.

Kurakin, A., Goodfellow, I., & Bengio, S. (2017). Adversarial examples in the physical world. *arXiv preprint* arXiv:1607.02533.

Kyprianou, A. E. (2014). *Fluctuations of Lévy processes with applications*. Springer.

Lando, D. (1998). On Cox processes and credit-risky securities. *Review of Derivatives Research, 2*(2-3), 99–120.

Lauritzen, S. L., & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society: Series B, 50(2), 157–224.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.

Lemieux, C. (2009). *Monte Carlo and quasi-Monte Carlo sampling*. Springer.

Lessmann, S., Baesens, B., Seow, H. V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

Li, D., & Walther, A. (2021). Binomial checkpointing in modern heterogeneous architectures. *SIAM Journal on Scientific Computing, 43*(5), C473–C497.

Liang, X., Zhao, J., & Yin, G. (2022). Federated quantum machine learning for credit risk assessment. *Journal of Financial Data Science, 4*(3), 55–70.

Lipton, Z. C. (2018). The mythos of model interpretability. *Communications of the ACM, 61*(10), 36–43.

Liptser, R., & Shiryaev, A. N. (2001). *Statistics of random processes II: Applications* (2nd ed.). Springer.

Liu, S., Grover, J., & Sim, G. (2022). Neural SDEs for derivative pricing and hedging. *Journal of Computational Finance, 26*(2), 85–123.

Lloyd, S., & Weedbrook, C. (2018). Quantum generative adversarial learning. *Physical Review Letters, 121*(4), 040502.

Louzis, D. P., & Vouldis, A. T. (2021). Hybrid factor copula–VAE models for credit risk stress testing. *Journal of Banking & Finance, 129*, 106147.

Luenberger, D. G. (1969). *Optimization by vector space methods*. Wiley.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30* (pp. 4765–4774).

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In Proceedings of the 6th International Conference on Learning Representations.

Malik, A., & Mishra, A. (2021). Stress scenario generation using conditional GANs. *Journal of Risk, 24*(2), 1–26.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2017). Communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS* (pp. 1273–1282).

McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative risk management: Concepts, techniques and tools* (Rev. ed.). Princeton University Press.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., ... & Modha, D. S. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. Science, 345(6197), 668–673. https://doi.org/10.1126/science.1254642

Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics, 3*(1–2), 125–144.

Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). Quantum circuit learning. *Physical Review A, 98*(3), 032309.

Mittal, S., & Vetter, J. S. (2015). A survey of CPU-GPU heterogeneous computing techniques. *ACM Computing Surveys, 47*(4), 69.

Molnar, C. (2020). *Interpretable machine learning* (2nd ed.). https://christophm.github.io/interpretable-ml-book/

Monetary Authority of Singapore. (2021). *Notice PSN01: Cyber hygiene*. https://mas.gov.sg

Moosavi-Dezfooli, S.-M., Fawzi, A., & Frossard, P. (2016). DeepFool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2574–2582).

Murphy, K. P. (2002). Dynamic Bayesian networks: Representation, inference and learning (Doctoral dissertation, University of California, Berkeley).

NVIDIA Corporation. (2023). *CUDA C++ programming guide* (Version 12.2). NVIDIA Developer Documentation.

Naumann, U. (2012). *The art of differentiating computer programs*. SIAM.

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. Jones, & X.-L. Meng (Eds.), Handbook of Markov chain Monte Carlo (pp. 113–162). CRC Press.

Nelsen, R. B. (2006). An introduction to copulas (2nd ed.). Springer.

Nesterov, Y. (2004). *Introductory lectures on convex optimization*. Springer.

Network for Greening the Financial System. (2021). NGFS climate scenario technical documentation. https://www.ngfs.net

Ni, J., & Pan, M. (2021). Hybrid copula–VAE models for credit portfolio risk. *Quantitative Finance, 21*(11), 1855–1872.

Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with CUDA. *ACM Queue, 6*(2), 40–53. https://doi.org/10.1145/1365490.1365500

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information* (10th anniversary ed.). Cambridge University Press.

Office of the Comptroller of the Currency. (2021). Bulletin 2021-37: Model risk management—Frequently asked questions.

Owen, A. B. (1998). Latin supercube sampling for very high-dimensional simulations. *ACM Transactions on Modeling and Computer Simulation, 8*(1), 71–102. https://doi.org/10.1145/272991.273006

O'Neal, M., & Vidler, C. (2022). Reg-to-code: Bridging prose regulation and executable tests. Journal of Financial Technology, 9(2), 45–62.

Papapantoleon, A. (2022). No-arbitrage principles in machine-learning models. *Risk*, 35(4), 64–69.

Patton, A. J. (2006). Modelling asymmetric exchange rate dependence. *International Economic Review, 47*(2), 527–556.

Pearl, J. (2009). Causality: Models, reasoning, and inference (2nd ed.). Cambridge University Press.

Pfeiffer, M., & Pfeil, T. (2018). Deep learning with spiking neurons: Opportunities and challenges. Frontiers in Neuroscience, 12, Article 774. https://doi.org/10.3389/fnins.2018.00774

Phipps, E. T., Pawlowski, R. P., & Salinger, A. G. (2012). Efficient expression templates for operator-overloading-based automatic differentiation. *ACM Transactions on Mathematical Software, 38*(1), 1–24.

Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum, 2*, 79. https://doi.org/10.22331/q-2018-08-06-79

Prometheus. (2023). Prometheus monitoring system v2.47 documentation. https://prometheus.io

Prorokowski, L. (2019). Risk data validation under BCBS 239. Journal of Risk Model Validation, 13(3), 1–23. https://doi.org/10.21314/jrmv.2019.207

Protter, P. (2005). *Stochastic integration and differential equations* (2nd ed.). Springer.

Pykhtin, M., & Zhu, S. (2007). *Guide to modeling counterparty credit exposures*. Risk Books.

Rabanser, S., & Günther, M. (2019). Failing loudly: An empirical study of methods for detecting dataset shift. In *Advances in neural information processing systems 32* (pp. 1396–1408).

Reisinger, C., & Wittum, G. (2009). Efficient hierarchical approximation of high-dimensional option-pricing problems. *SIAM Journal on Scientific Computing, 31*(4), 2319–2347.

Revuz, D., & Yor, M. (1999). *Continuous martingales and Brownian motion* (3rd ed.). Springer.

Rezazadeh, A., & Lázaro-Gredilla, M. (2020). Variational copula inference. *Machine Learning, 109*(10), 1865–1898.

Robert, C. P., & Casella, G. (2004). Monte Carlo statistical methods (2nd ed.). Springer.

Rockafellar, R. T., & Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of Risk, 2*(3), 21–41.

Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling. In S. van der Walt & J. Millman (Eds.), Proceedings of the 14th Python in Science Conference (pp. 130-136).

Rohrer, M., Li, Q., & Bolton, T. (2022). Zero-copy data exchanges with Apache Arrow. *Journal of Data Engineering, 7*(2), 15–28.

Rosenbaum, M., & Robert, C. Y. (2015). Volatility of volume and short-term portfolio selection. *Quantitative Finance, 15*(5), 843–856.

Royal Bank of Canada. (2022). *Model risk management policy* (Internal publication).

Rudin, C., & Radin, J. (2019). Why are we using black box models in AI when we don't need to? A lesson from an explainable AI competition. *Harvard Data Science Review, 1*(2). https://doi.org/10.1162/99608f92.5a8a3a3d

Ruf, J., & Wang, W. (2020). Hedging derivatives using deep reinforcement learning. Journal of Financial Data Science, 2(4), 10–27. https://doi.org/10.3905/jfds.2020.1.033

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., … Tarantola, S. (2008). *Global sensitivity analysis: The primer*. Wiley.

Sanders, H., Gidon, A., & Bienenstock, E. (2021). Spiking neural networks for neuromorphic risk sensing. *IEEE Transactions on Neural Networks and Learning Systems, 32*(10), 4458–4471.

Schoutens, W. (2003). *Lévy processes in finance: Pricing financial derivatives*. Wiley.

Schuld, M., Sinayskiy, I., & Petruccione, F. (2015). An introduction to quantum machine learning. *Contemporary Physics, 56*(2), 172–185.

Shapiro, A., Dentcheva, D., & Ruszczyński, A. (2014). *Lectures on stochastic programming* (2nd ed.). SIAM.

Shojaie, A., & Michailidis, G. (2010). Discovering graphical Granger causality using the truncating lasso penalty. Bioinformatics, 26(18), i517–i523. https://doi.org/10.1093/bioinformatics/btq373

Shreve, S. E. (2004). *Stochastic calculus for finance II: Continuous-time models*. Springer. https://doi.org/10.1007/978-1-4757-4296-1

Shumway, R. H., & Stoffer, D. S. (2017). Time series analysis and its applications (4th ed.). Springer.

Siddiqi, N. (2012). *Credit risk scorecards: Developing and implementing intelligent credit scoring*. John Wiley & Sons.

Sigelman, B., & Barroso, L. (2015). Dapper, large-scale distributed systems tracing infrastructure. *Google Technical Report*. https://research.google/pubs/pub36356

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the 7th International Conference on Document Analysis and Recognition* (pp. 958–963).

Smith, V., Chiang, C. K., Sanjabi, M., & Talwalkar, A. (2020). Federated multi-task learning. In *Advances in Neural Information Processing Systems, 33* (pp. 4424–4434).

Sobol, I. M. (2001). Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation, 55*(1–3), 271–280. https://doi.org/10.1016/S0378-4754(00)00270-6

Spirtes, P., Glymour, C., & Scheines, R. (2000). Causation, prediction, and search (2nd ed.). MIT Press.

Stein, D., Simeonov, P., & Woerner, S. (2023). Quantum generative models for credit spread dynamics. *Quantum Finance Journal, 1*(1), 1–18.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning* (pp. 3319–3328).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. *International Conference on Learning Representations*.

Tavella, D., & Randall, C. (2000). *Pricing financial instruments: The finite difference method*. Wiley.

Thomas, L. C., Crook, J., & Edelman, D. (2017). *Credit scoring and its applications* (2nd ed.). SIAM.

Tsantekidis, A., Passalis, N., Tefas, A., Zafeiriou, S., Vlahavas, I., & Aletras, N. (2017). Forecasting stock prices from the limit order book using convolutional neural networks. In *IEEE 19th Conference on Business Informatics* (pp. 7–12).

Um, T. T., Ker, J., et al. (2017). Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks. In Proceedings of the 19th ACM International Conference on Multimodal Interaction (pp. 216–220).

Umberto, R., Fazzi, M., & Kilian, J. (2022). Frequency-domain augmentation for rare-event time series. *Journal of Computational Finance, 26*(2), 45–67.

Vasicek, O. (1987). Probability of loss on loan portfolio. KMV Corporation Technical Report.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998–6008).

Wang, Z., Dongarra, J., & Tomov, S. (2010). Optimizing data movement for GPU accelerated AD. In *Proceedings of the International Conference on High Performance Computing* (pp. 123–132).

Wei, G. C. G., & Tanner, M. A. (1990). A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. Journal of the American Statistical Association, 85(411), 699–704.

Wen, R., Tway, D., & Ravuri, S. (2020). Time-series data augmentation for deep learning: A survey. *arXiv preprint* arXiv:2002.12463.

Wiese, M., Knobloch, R., Korn, R., & Kretschmer, T. (2020). Quant GANs: Deep generative models for quantitative finance. *Applied Stochastic Models in Business and Industry, 36*(4), 684–707.

Woerner, S., & Egger, D. J. (2019). Quantum risk analysis. *npj Quantum Information, 5*(1), 15.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2003). Understanding belief propagation and its generalizations. In G. Lakemeyer & B. Nebel (Eds.), Exploring artificial intelligence in the new millennium (pp. 239–269). Morgan Kaufmann.

Zaharia, M., Chen, A., Davidson, A., & Ghodsi, A. (2016). Apache Spark: A unified engine for big data processing. Communications of the ACM, 59(11), 56-65.

Zhang, X., Zohren, S., & Roberts, S. J. (2022). Deep order book: Methods, data, and lessons learned. Quantitative Finance, 22(3), 573–598. https://doi.org/10.1080/14697688.2022.2038831

Zhou, B., Naseer, M., & Fowers, J. (2019). Accelerating graph workloads with Graphcore IPU. In *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture*.

Zoufal, C., Lucchi, A., & Woerner, S. (2019). Quantum generative adversarial networks for learning and loading random distributions. npj Quantum Information, 5, Article 103. https://doi.org/10.1038/s41534-019-0223-2

Øksendal, B. (2003). *Stochastic differential equations: An introduction with applications* (6th ed.). Springer. https://doi.org/10.1007/978-3-642-14394-6