

Rijeka Methodology

Curve construction, pricing, risk, XVA, digital-asset derivatives and on-chain confirmation

Version 1.0 · October 2026 · github.com/MikoDevedzic/rijeka · Apache-2.0

How to read this document. Every formula below is the one implemented in the open-source code, with the module named in the margin of each section. Where the code departs from textbook practice, or where a simplification is in force, it is stated as a *known limitation* rather than hidden. The Methodology tab of the application exports, for each product, an Excel workbook that rebuilds the default trade with live formulas from the user's own market data next to the engine's value; those workbooks are the executable companion to this text.

Contents

1	Scope and principles	2
1.1	Notation and conventions	2
2	Discount curve construction	2
2.1	Instruments	2
2.2	Algorithm	2
2.3	Interpolation and extrapolation	3
2.4	Risk bumps	3
3	Schedules and cashflows	3
4	Linear interest-rate products	4
5	European swaptions and the volatility surface	4
5.1	Annuity, forward and the Bachelier price	4
5.2	Volatility tiers	4
5.3	$\beta = 0$ normal SABR	5
6	Caps, floors and collars	5
7	Bermudan swaptions: Hull–White trinomial tree	5
8	Hull–White calibration for exposure simulation	6
9	Exposure, collateral and the credit support annex	6
9.1	Simulation	6
9.2	Variation margin and the margin period of risk	6
9.3	Initial margin	6
9.4	Residual exposure	6
9.5	Presets	7
9.6	SA-CCR	7
10	The XVA waterfall	7
11	P&L attribution	7
12	Currency and digital-asset forwards	7
13	On-chain bilateral confirmation	8

13.1 The canonical record (schema v2)	8
13.2 Signatures	9
13.3 The registry	9
13.4 Verification without Rijeka	9
14 Model governance and known limitations	9

1 Scope and principles

Rijeka is a front-to-back desk for over-the-counter derivatives: booking, pricing, cashflows, risk, XVA under a credit support annex, ISDA SIMM initial margin, and bilateral confirmation anchored to Ethereum. Phase 1 covers interest-rate products on overnight-index curves (USD SOFR, EUR €STR, GBP SONIA and others) and non-deliverable digital-asset forwards and options.

Three principles govern the implementation.

1. **No black boxes.** Each number is traceable to a formula in this document and a function in the repository; the Excel replications reproduce it outside the system.
2. **Risk taxonomy follows ISDA SIMM.** Sensitivities are IR01 (all curves bumped), IR01_DISC (discount curves only), THETA, GAMMA, VEGA; the terms PV01 and DV01 are not used.
3. **Market data is dated and sourced.** Every snapshot carries a valuation date and a source. Pricing off a stale snapshot is permitted but is always flagged with the snapshot date; it is never silent.

1.1 Notation and conventions

t_0	valuation date
$P(t, T)$	discount factor from T to t on the OIS curve; $P(t_0, T)$ written $P(T)$
$\tau(d_1, d_2)$	year fraction under a day-count convention: ACT/360, ACT/365F, 30/360 bond basis (ISDA 2006 §4.16(f)), ACT/ACT ISDA
N, K, F	notional, strike, forward
σ_N	normal (Bachelier) volatility, quoted in basis points per annum
Φ, φ	standard normal distribution and density

Business-day adjustment is Modified Following on the currency's calendar unless the trade says otherwise. Spot lag is two business days for USD, EUR, JPY and CHF, one for CAD, zero for GBP, AUD and NZD (`pricing/bootstrap.py`, `CCY_SPOT_LAG`).

2 Discount curve construction

Module: `pricing/bootstrap.py`, `pricing/curve.py`.

2.1 Instruments

A curve is built from an overnight deposit and a strip of OIS par swap rates (tenors in weeks, months and years). Futures, FRAs and IRS quote types are accepted by the engine but Phase 1 curves use deposit and OIS swaps only.

2.2 Algorithm

1. The overnight deposit gives $P(t_0 + 1d) = 1/(1 + r_{ON} \tau_{ACT/360})$.

2. The spot discount factor $P(t_s)$ is obtained by extrapolating the overnight zero rate to the spot date and is *frozen* as an explicit pillar. This is the step that makes the bootstrap and the pricer agree to the last cent: both read the same $P(t_s)$.
3. Annual OIS swaps are bootstrapped on a dense integer-year grid (1Y to the longest quote), with missing years filled by linear interpolation of par rates, so that every annual period end is a pillar.
4. Week- and month-tenor swaps are then bootstrapped in maturity order against the pillars already present.

For an OIS swap of par rate R with annual periods $(T_{i-1}, T_i]$, $i = 1 \dots n$, accrual factors τ_i (ACT/360 for USD), the terminal discount factor solves the par condition with the floating leg replaced by its telescoping value $P(t_s) - P(T_n)$:

$$P(T_n) = \frac{P(t_s) - R \sum_{i=1}^{n-1} \tau_i P(T_i)}{1 + R \tau_n}. \quad (1)$$

Because every period end is a pillar, a swap struck at its own par rate reprices to exactly zero.

2.3 Interpolation and extrapolation

Between pillar dates $d_0 < d < d_1$ the discount factor is log-linear in the calendar-day fraction $w = (d - d_0)/(d_1 - d_0)$:

$$P(d) = \exp [(1 - w) \ln P(d_0) + w \ln P(d_1)], \quad (2)$$

equivalent to a piecewise-constant forward rate. Beyond the last pillar the continuously-compounded zero rate is held flat. Before the first pillar $P = 1$. The Excel replications implement exactly this rule with MATCH/INDEX, so a reader can check any discount factor the engine produces.

2.4 Risk bumps

A parallel shift of b basis points is applied to zero rates: $P_b(T) = P(T) e^{-b \cdot 10^{-4} T}$ with T in ACT/365F years (`Curve.shifted`).

3 Schedules and cashflows

Module: `pricing/schedule.py`, `pricing/day_count.py`, `pricing/ir_swap.py`.

Coupon periods are rolled backward from the maturity date at the payment frequency (monthly, quarterly, semi-annual, annual, or zero-coupon), producing a short front stub by default; explicit first-period-start and last-period-end dates override this. Period end dates are unadjusted; payment dates apply the business-day convention and any payment lag. Floating fixings are two business days before the period start.

A fixed coupon over $(d_s, d_e]$ is $N r \tau(d_s, d_e)$. A floating coupon on a compounded overnight index is projected from the forecast curve as

$$r_{float} = \frac{1}{\tau(d_s, d_e)} \left(\frac{P_f(d_s)}{P_f(d_e)} - 1 \right) + s, \quad (3)$$

which equals the daily-compounded in-arrears rate implied by the curve; s is the spread. For a period already in progress, a realised fixing (when supplied) covers the elapsed part and the curve the remainder, weighted by accrual. Notional, fixed-rate and spread schedules (step-ups, amortisation) are resolved per period from their effective dates.

Each coupon is discounted at its business-day-adjusted payment date, the date the cash moves. (Engine versions before October 2026 discounted at the unadjusted period end; where a date rolled over a weekend the two differ by a day or two of discounting. Corrected in v0.5, with the Excel replications as the independent check.)

4 Linear interest-rate products

Module: `pricing/ir_swap.py`, `pricing/greeks.py`.

The value of a leg is the signed sum of discounted coupons, $PV = \pm \sum_k C_k P(T_k)$, with pay legs negative. A swap's NPV is the sum of its legs; a basis swap is two floating legs on different forecast curves; an OIS swap is the single-curve case where forecast and discount curves coincide. The par rate is $R^* = PV_{float} / (N \sum_i \tau_i P(T_i))$.

Sensitivities are computed by bump-and-reprice:

- **IR01**: all curves shifted +1bp;
- **IR01_DISC**: discount curves shifted +1bp with forecasting held fixed (in single-curve mode this requires a separate bumped copy for discounting, so that $IR01_DISC < IR01$; the difference is the forecasting sensitivity);
- **THETA**: $NPV(t_0 + 1d) - NPV(t_0)$ with curves held;
- **GAMMA**: second difference of $\pm 1bp$ shifts.

5 European swaptions and the volatility surface

Module: `pricing/swaption.py`, `pricing/sabr.py`.

5.1 Annuity, forward and the Bachelier price

For an option expiring at T_e into a swap paying annually at $T_1 < \dots < T_n$ (dates rolled backward from the swap's maturity, with the trade's spot lag and business-day adjustment when the trade dates are supplied),

$$A = \sum_{i=1}^n \tau_i P(T_i), \quad F = \frac{P(T_0) - P(T_n)}{A}. \quad (4)$$

Rates on overnight curves can be negative and are quoted in normal volatility, so the option is priced in the Bachelier model. With $v = \sigma_N \sqrt{T_e}$ and $d = (F - K)/v$,

$$V_{payer} = N A v [d \Phi(d) + \varphi(d)], \quad V_{receiver} = N A v [-d \Phi(-d) + \varphi(d)]. \quad (5)$$

Greeks are produced by bump-and-reprice: vega per 1bp of σ_N , IR01, theta per calendar day, volga, vanna, dollar gamma, and a delta-hedge notional. A Hull–White cross-check price is reported alongside when the model is calibrated (§7).

5.2 Volatility tiers

The volatility used is tagged **QUOTED** (an at-the-money surface point at the trade's expiry and tenor), **SABR** (an off-the-money strike read from the calibrated smile), **INTERPOLATED** (between grid points), or **MANUAL** (user supplied). The tier travels with the price so that a valuation can be challenged.

5.3 $\beta = 0$ normal SABR

The smile at each expiry–tenor point is the normal SABR of Hagan et al. (2002) with backbone $\beta = 0$:

$$\sigma_N(K) = \alpha C(F, K) \left\{ 1 + \left[\frac{2 - 3\rho^2}{24} \nu^2 + \frac{\rho \nu \alpha}{4} \right] T \right\}, \quad (6)$$

$$C = \frac{z}{\chi(z)}, \quad z = \frac{\nu}{\alpha}(F - K), \quad \chi(z) = \ln \frac{\sqrt{1 - 2\rho z + z^2} + z - \rho}{1 - \rho},$$

with $C = 1$ at the money. Parameters (α, ρ, ν) are fitted to quoted volatilities at strike offsets $\{-200, -100, -50, -25, 0, +25, +50, +100, +200\}$ bp by weighted least squares, with α constrained so the at-the-money volatility is matched exactly. The fit RMSE is stored with the parameters.

6 Caps, floors and collars

Module: *pricing/cap_floor.py*, *pricing/cap_vol_strip.py*, *pricing/cap_sabr.py*.

A cap on a compounded overnight index is a strip of quarterly caplets, ACT/360, settled $T + 2$, Modified Following, with each caplet expiring at the *end* of its accrual period (fixing in arrears). For caplet $(d_s, d_e]$ with forward F from (3), time to fixing T and normal volatility σ_N ,

$$\text{caplet} = N \tau P(d_e) \left[(F - K) \Phi(d) + \sigma_N \sqrt{T} \varphi(d) \right], \quad d = \frac{F - K}{\sigma_N \sqrt{T}}, \quad (7)$$

and a floorlet is the corresponding put. The at-the-money strike of a cap is the annuity-weighted average of its caplet forwards. A collar is a long cap and a short floor.

Caplet volatilities are stripped from flat cap volatilities one strike column at a time: the 1Y bucket is solved so that four Bachelier caplets reproduce the 1Y cap price; each later bucket (2Y, 3Y, 3–5Y, 5–7Y, 7–10Y) is solved after subtracting the value of the buckets already priced, piecewise flat within a bucket. A $\beta = 0$ SABR smile is then fitted per expiry bucket across strikes.

7 Bermudan swaptions: Hull–White trinomial tree

Module: *pricing/bermudan_swaption.py*.

The short rate follows $dr_t = [\theta(t) - a r_t] dt + \sigma dW_t$. Zero-coupon bonds are affine, $P(t, T) = A(t, T) e^{-B(t, T) r_t}$ with $B(t, T) = (1 - e^{-a(T-t)})/a$ (Brigo–Mercurio 2006, eq. 3.39). The tree is the Hull–White (1994) trinomial tree: a symmetric tree in $x_t = r_t - \alpha(t)$ with the standard branching rules, and a deterministic shift $\alpha(t)$ fitted at every grid time by Arrow–Debreu forward induction so that the tree reprices the market discount curve exactly. Backward induction values the underlying swap at each node by the single-curve telescoping identity $PV_{float}(t) = \sum_k N_k [P(t, T_{k-1}) - P(t, T_k)]$, and applies the early-exercise test at each exercise date.

Validation gate. Before any Bermudan value is trusted, the same tree must price European swaptions within 0.5% (at the money) and 1% (± 100 bp) of the closed-form Bachelier price on a grid of expiries and tenors. **Scope.** The model is single-curve and single-tenor: multi-curve underlyings, capped or floored underlyings and basis underlyings are refused rather than approximated, and a Bermudan whose underlying tenor is far from the 5Y calibration column carries a warning.

8 Hull–White calibration for exposure simulation

Module: *pricing/calibration.py*.

With constant (a, σ) , the model's at-the-money normal volatility for a swaption with expiry T_e on a swap with payment dates T_i , accruals α_i and annuity $A(0) = \sum_i \alpha_i P(0, T_i)$ is (Andersen–Piterbarg 2010, vol. 1 §4.4)

$$\sigma_N(T_e) = \sigma \frac{|w| \sqrt{V(T_e)}}{\sqrt{T_e}}, \quad w = \sum_i \frac{\alpha_i P(0, T_i)}{A(0)} B(T_e, T_i), \quad V(T_e) = \frac{1 - e^{-2aT_e}}{2a}, \quad (8)$$

where $P(0, T)$ is the *market* discount factor from the bootstrapped curve (a flat-rate approximation is the fallback only when no curve is supplied). A constant-parameter model cannot fit a whole surface; following standard XVA practice it is calibrated to the 5Y-tenor column (1Y×5Y, 2Y×5Y, 3Y×5Y, 5Y×5Y) by least squares, and the fit diagnostics are stored with the parameters. The calibration was verified against Jamshidian's decomposition to within 0.01%.

9 Exposure, collateral and the credit support annex

Module: *pricing/csa.py*, *pricing/xva_engine.py*, *api/routes/xva.py*.

9.1 Simulation

Short-rate paths are simulated under the calibrated Hull–White model on a monthly grid with the exact fit to the initial curve, and the trade is revalued on every path and date, giving mark-to-market paths $V(t)$ from our side.

9.2 Variation margin and the margin period of risk

Variation margin held at t was called against the exposure one margin period of risk δ earlier:

$$C(t) = \max(V(t - \delta) - TH_{cp}, 0) - \max(-V(t - \delta) - TH_{own}, 0), \quad (9)$$

zeroed when $|C|$ is below the minimum transfer amount. The margin period δ is shorter than a grid step, so $V(t - \delta)$ is sampled from the Brownian bridge between grid points,

$$V(t - \delta) | V(t - \Delta t), V(t) \sim \mathcal{N}(V(t - \Delta t) + w \Delta V, w(1 - w) s^2), \quad w = \frac{\Delta t - \delta}{\Delta t}, \quad (10)$$

with s^2 the cross-path variance of ΔV . This preserves the variance of the move over the margin period; linear interpolation would understate it by a factor $\delta/\Delta t$.

9.3 Initial margin

Initial margin follows ISDA SIMM interest-rate delta: $IM = RW(\text{tenor}) \times |\text{IR01}|$ with the published risk weights for regular-volatility currencies (109bp at two weeks falling to 61bp at 5–20 years and 64bp at 30 years). The weights are calibrated to a 10-business-day margin period; for another period they scale by $\sqrt{\delta/10\text{bd}}$. Margin is assumed exchanged symmetrically and amortises with residual maturity. Received IM reduces exposure; posted IM is funded and drives MVA.

9.4 Residual exposure

$$E^+(t) = \max(V(t) - C(t) - IM_{recv}(t), 0), \quad E^-(t) = \min(V(t) - C(t) + IM_{post}(t), 0). \quad (11)$$

Expected exposure $EE(t) = \mathbb{E}[E^+(t)]$, expected negative exposure $ENE(t) = \mathbb{E}[E^-(t)]$ and the 97.5% potential future exposure are reported per date.

9.5 Presets

Uncollateralised	no variation or initial margin.
Bilateral	daily VM, 10-business-day margin period (the BCBS–IOSCO floor for non-cleared trades), SIMM IM, thresholds and MTA from the CSA.
On-chain	atomic settlement: continuous VM, zero threshold and MTA, 5-business-day margin period (the cleared floor), SIMM IM scaled by $\sqrt{5/10}$.

The on-chain preset is the quantitative claim behind the confirmation layer: with continuous settlement the margin period halves, SIMM initial margin falls by $1 - \sqrt{1/2} \approx 29\%$, and the SA-CCR maturity factor falls in the same proportion.

9.6 SA-CCR

For margined netting sets the maturity factor is $MF = 1.5\sqrt{\max(\delta, \text{floor})/250}$ and replacement cost $RC = \max(V - C, TH + MTA - NICA, 0)$ with $NICA$ the IM received; unmargined sets use $MF = \sqrt{\min(M, 1)}$ and $RC = \max(V, 0)$. The interest-rate supervisory factor is 0.5% and $\alpha = 1.4$. Capital is 8% of risk-weighted exposure and feeds KVA.

10 The XVA waterfall

Module: *pricing/xva_engine.py*, function *run_waterfall*.

Default probabilities come from the credit triangle: hazard rate $h = s/LGD$ for a flat or term-structured spread s , survival $Q(t) = e^{-ht}$. With $D_i = P(t_i)$, Δt the grid step and $\Delta Q_i = Q(t_{i-1}) - Q(t_i)$,

$$CVA = LGD_{cp} \sum_i EE_i \Delta Q_i^{cp} D_i \omega_i, \quad DVA = -LGD_{own} \sum_i |ENE_i| \Delta Q_i^{own} D_i, \quad (12)$$

$$FVA = \sum_i s_i^{ftp} EE_i \Delta t D_i, \quad FBA = - \sum_i s_i^{fba} |ENE_i| \Delta t D_i, \quad (13)$$

$$KVA = \sum_i \kappa K_i \Delta t D_i, \quad MVA = - \sum_i IM_i \max(s_i^{ftp} - \pi, 0) \Delta t D_i, \quad (14)$$

where ω_i is an optional wrong-way-risk multiplier, κ the capital hurdle rate, K_i the SA-CCR capital on the exposure path, and π a yield pick-up on posted collateral (tokenised collateral that earns a return reduces MVA). Signs follow the convention that costs are positive. The waterfall is reported gross and under the CSA, so the benefit of collateral is visible term by term.

11 P&L attribution

Module: *pricing/pnl.py*, the scenario engine in the trade window.

Daily P&L is explained by a Taylor expansion in the taxonomy above: theta, $IR01 \times \Delta r$, $IR01_DISC \times \Delta r_{disc}$, $\frac{1}{2} GAMMA \times \Delta r^2$, vega $\times \Delta \sigma$, carry and roll-down, with the residual reported as *unexplained*, never absorbed. The joint rate-and-volatility scenario engine attributes a shock the same way per trade. A desk-level daily P&L with new-trade and amendment effects, and independent price verification against an external snapshot, are on the roadmap and depend on a market-data feed that moves day to day.

12 Currency and digital-asset forwards

Module: *pricing/fx_forward.py*, *chain/canonical.py* (*CRYPTO_FORWARD*, *CRYPTO_OPTION*).

FX spot, forwards and non-deliverable forwards on a currency pair, and forwards on BTC, ETH or SOL against USD, are one product priced by covered interest parity (cash-and-carry): borrow the quote currency, buy the base today and lend it, so the forward must repay the quote-currency interest. Each currency discounts off its own overnight-index curve (USD SOFR, EUR €STR, GBP SONIA, JPY TONAR, CHF SARON, AUD AONIA, CAD CORRA); a digital asset discounts at its funding rate r_c (lending or staking yield, continuously compounded). With spot S in quote per base, $P_q(T)$ the quote-currency discount factor and $P_b(T)$ the base-currency one, $F = S P_b(T)/P_q(T)$ and $NPV_q = \pm N_b (F - K) P_q(T)$; deliverable or non-deliverable changes the settlement mechanics, not the value. When a pair carries a quoted forward curve (forward points by tenor, saved like any other snapshot: Bloomberg, manual or import), the outright at a date is spot plus the points interpolated linearly in time, and the difference to the parity forward is reported as the cross-currency basis; the account chooses, in configuration, whether forwards are priced off quoted points or implied from the two curves, and every quote states which it used. For a digital asset against USD this reads

$$F = S \frac{e^{-r_c T}}{P_{USD}(T)} = S e^{(r_{USD} - r_c)T}, \quad NPV = (F - K) N P_{USD}(T). \quad (15)$$

A non-deliverable European option is priced with Black-76 on that forward with lognormal volatility σ :

$$C = P_{USD}(T) [F \Phi(d_1) - K \Phi(d_2)], \quad d_{1,2} = \frac{\ln(F/K) \pm \frac{1}{2} \sigma^2 T}{\sigma \sqrt{T}}, \quad (16)$$

with the put from parity. Settlement is in cash against a named settlement price source on a valuation date and time, and the canonical record carries exactly the fields the ISDA Digital Asset Derivatives Definitions name: digital asset, notional amount, strike or forward price and its currency, option type and exercise style, valuation and settlement dates, settlement currency and price source, premium amount, currency and payment date. **Status.** Implemented in the engine (`pricing/currency_forward.py`, with `pricing/crypto_forward.py` as the digital-asset case), priced from parameters and from booked trades on G10 pairs and BTC, ETH, SOL; spot from public reference rates (ECB) and a public exchange feed, the funding rate a desk input; the Excel replication rebuilds the digital-asset case cell by cell. The canonical on-chain record carries an FX product (pair, both notionals, rate, value date, settlement type, NDF fixing fields) beside the digital-asset one. Option pricing in the trade window is on the roadmap. A validation note: the first workbook had the discount-factor ratio inverted (forward below spot at positive USD rates); the engine's parity test caught it and both were corrected on 28 October 2026.

13 On-chain bilateral confirmation

*Modules: `chain/canonical.py`, `chain/signing.py`, `chain/attestation.py`;
the registry contract `chain/src/TradeConfirmationRegistry.sol`; the verifier `chain/verify/index.html`.*

13.1 The canonical record (schema v2)

Both parties sign one record that either can derive alone from its own booking. The record is keyed by the unique transaction identifier, names the two parties by LEI in sorted order, names the payer and receiver LEI of every leg, the buyer and seller of every embedded option and the payer and receiver of every custom cashflow, and orders legs by content. It contains only the economic terms: no internal references, no book or desk, no status, no timestamps, no party names. Serialisation rules, which a verifier in any language must reproduce exactly: JSON, UTF-8, keys sorted at every level, separators `,` and `:` with no whitespace; numbers as

decimal strings without exponent, trailing zeros or negative zero; dates ISO-8601; nulls and booleans preserved; strings unchanged. The trade hash is $h = \text{keccak256}(\text{bytes})$. The schema version is inside the payload so a confirmation is always re-derived under the version it was signed with.

13.2 Signatures

Each party signs an EIP-712 typed message in the domain `RijekaTradeConfirmation`, version 1, bound to the chain and the registry address:

```
TradeConfirmation(bytes32tradeHash,addresscounterparty)
TradeAmendment(bytes32prevHash,bytes32newHash,addresscounterparty)
TradeTermination(bytes32tradeHash,addresscounterparty)
```

Each party's message names the *other* party's address, so a signature cannot be replayed against a different counterparty, chain or registry. A firm's signing address is registered in Rijeka through a wallet challenge; once registered, Rijeka can verify that firm's signatures but can never produce them. On the hosted demo Rijeka holds keys for demo entities only, and any confirmation it signs for both sides is marked `bilateral:false`.

13.3 The registry

The registry contract has no owner and no administrative function. `confirm` verifies both signatures over the confirmation digest and records $(h, \text{partyA}, \text{partyB}, \text{timestamp})$ with status *Confirmed*; `amend` verifies both signatures over the amendment digest, marks the previous hash *Superseded* and records the new hash with the previous one as its parent; `terminate` verifies both signatures and sets *Terminated*. Signatures with high s values are rejected. Once a trade is anchored, Rijeka refuses off-chain edits to its terms: an amendment is itself a bilateral act.

13.4 Verification without Rijeka

A proof pack contains the canonical record, the hash, both signatures and the anchor. The public verifier, a single self-contained page, (i) re-serialises the record and recomputes the keccak256 hash, (ii) rebuilds both EIP-712 digests and recovers both signers, (iii) reads the registry on a public node and checks status and parties. No Rijeka server is contacted. The contract source is verified on Sourcify so the reader can audit what the registry does before trusting it. **What it proves:** that these exact terms were signed by these two keys and recorded at that block. **What it does not prove:** that a key belongs to a legal entity; that binding is a matter of the firms' own key management and registration.

14 Model governance and known limitations

- **Validation.** Versioned validation reports live in `model_validation/`; v0.4.2 covers IRS and OIS (38 tests against Bloomberg), later versions swaptions (delta of \$3 on a \$10m 5Y), caps (0.1% at matched volatility), SABR and the scenario engine.
- **Replications.** The Excel workbooks in the Methodology tab are regenerated on request from the user's own snapshot; the engine's value is written beside the formula result with a difference cell.
- **Open items, stated plainly.** Hull–White is single-curve and calibrated to one tenor column (§7–8). SABR is $\beta = 0$ only. The hosted demo prices off a dated snapshot with no live feed, and says so on every screen. The registry contract has had no external audit. SIMM covers interest-rate delta; vega and other risk classes are on the roadmap.
- **Change control.** Every change to a pricing module is a commit in a public repository with its tests; this document is versioned with the code.

References

- Andersen, L. and Piterbarg, V. (2010). *Interest Rate Modeling*, vols. I–III. Atlantic Financial Press.
- Brigo, D. and Mercurio, F. (2006). *Interest Rate Models: Theory and Practice*, 2nd ed. Springer.
- Hagan, P., Kumar, D., Lesniewski, A. and Woodward, D. (2002). Managing smile risk. *Wilmott*, September, 84–108.
- Hull, J. and White, A. (1994). Numerical procedures for implementing term structure models I. *Journal of Derivatives* 2(1), 7–16.
- Gregory, J. (2020). *The xVA Challenge*, 4th ed. Wiley.
- Green, A. (2015). *XVA: Credit, Funding and Capital Valuation Adjustments*. Wiley.
- BCBS (2014). *The standardised approach for measuring counterparty credit risk exposures (SA-CCR)*.
- BCBS–IOSCO (2015). *Margin requirements for non-centrally cleared derivatives*.
- ISDA (2024). *ISDA SIMM Methodology*, version 2.6; ISDA (2023). *Digital Asset Derivatives Definitions*.
- Ethereum Improvement Proposal 712: Typed structured data hashing and signing.