Chapter 19

Numerical Methods

Almost every model in these notes was chosen for the same reason, and it was not realism. Affine dynamics give Riccati equations, Gaussian short rates give closed-form bonds, Heston gives a characteristic function, SABR gives an expansion — and each of those buys a price in microseconds, which is what made calibration possible on the hardware the models were designed for. That constraint shaped the canon, and it has largely lifted. This chapter is about what replaced it: what a lattice and a simulation actually cost, why early exercise is where the difficulty concentrates, and what becomes available once a model is no longer required to have a formula. We measure the costs rather than assert them, including two biases in the standard method for early exercise that pull in opposite directions and are usually discussed as one.

19.1 The Constraint That Chose the Models

Model	**What it was built to give in closed form**
Vasicek, Hull-White (ch. 8)	Bond prices, and hence swaptions by Jamshidian's decomposition
Affine models (ch. 14)	Bonds via Riccati equations
Heston (ch. 10)	A characteristic function, and prices by Fourier inversion
SABR (ch. 10)	An asymptotic expansion for implied volatility
Quasi-Gaussian (ch. 12)	A finite Markov state, so a low-dimensional grid

None of these was selected because the world behaves that way. Each was selected because a calibration needs thousands of model prices, and a model that takes a second to price cannot be calibrated at all if a second is what one has for the whole exercise.

Remark (What the constraint cost)**.** The earlier chapters have already itemised the bill without presenting it as one. Chapter 10's local volatility model fits every European price and predicts the wrong dynamics. Chapter 10's SABR fits one expiry and is not consistent across expiries. Chapter 12's quasi-Gaussian models buy a finite state by insisting the volatility structure decay exponentially, which nothing in the market requires. Chapter 18's Gaussian copula is the extreme

case: chosen because it is easy to simulate and calibrate, and structurally unable to represent the one quantity it was used for.

In each case a functional form was adopted because it was solvable, and the market then declined to agree. That is not an argument against the models — they are still what one reaches for, and most of this chapter's alternatives are slower and less understood. It is an argument for knowing which of a model's properties are markets and which are convenience.

Structure (Which constraint is binding now)**.** The models of chapter 5 through chapter 12 were selected under a constraint: a price had to be computable by hand, or by a machine far weaker than the one on any desk today. That constraint chose affine dynamics, exponential volatility structures and one-factor curves, and it chose them for tractability rather than for realism — which is what chapter 14 spends a chapter making precise.

That constraint has lifted. What has not lifted, and what this chapter argues is now the binding one, is *identification*. A model can be solved numerically at almost any complexity; it cannot be told apart from its neighbours by the instruments the market quotes. These notes have measured that three times over — SABR's β against its correlation, local-stochastic volatility's mixing weight, and quasi-Gaussian's mean reversion — and in each case the flatness was a property of the quoted instruments rather than of the optimiser. Compute stopped binding and data started.

So the question is not whether a solvable model is still *necessary*. It is not. The question is what solvability was providing besides speed, and the answer turns out to be two things: prices that are arbitrage-free for structural rather than statistical reasons, and errors one can bound in advance. The next section is about the first and §19.7 about the second.

19.2 Two Ways to Compute, and What Each Costs

Everything that is not a closed form is one of two things, and their weaknesses are complementary in a way that decides most practical choices.

Lattices and grids

Backward induction on a tree, or a finite difference solution of the pricing equation of chapter 5. Chapter 9's forward equation can be run the other way for a whole strike surface at once.

The virtues are exactness and early exercise. There is no sampling error — refine the grid and the answer converges, at a rate that is second order in the spacing for a decent scheme — and early exercise is free, because backward induction already knows the continuation value when it needs to compare against intrinsic. That is why chapter 20's model selection table sends callables to short rate models: not because those models are more realistic, but because their small state fits on a grid.

The vice is dimension. A tensor grid with n points per axis in d dimensions has n^d points:

d	**points**	d	**points**
1	10^2	4	10^8
2	10^4	5	10^{10}
3	10^6	6	10^{12}

at a hundred points an axis. Three dimensions is comfortable, four is a project, and five is not happening. A basket over ten underlyings, or a market model with twenty forward rates, is not a large grid — it is no grid at all.

Simulation

Monte Carlo has exactly the opposite profile. Its error depends on the number of paths and not on the dimension, and it handles path dependence without any special effort because it has the path.

Calculation 19.1 (The rate, measured)**.** The standard error of a Monte Carlo estimate falls as $1/\sqrt{N}$. Pricing a one year European put in `quant/src/numerics.rs`:

Paths	**Standard error**	**Error** $\times\sqrt{N}$
10^3	0.349	11.02
10^4	0.107	10.67
10^5	0.0343	10.85
10^6	0.0109	10.89

The last column is flat across three decades, which is the rate confirmed rather than assumed.

The second half of that claim decides which method a problem gets. Doing it honestly needs an integral whose *dimension* can be varied while everything else is held still, so build one: drive a single lognormal with d independent normals combined into one,

$$Z = \frac{z_1 + \cdots + z_d}{\sqrt{d}}, \qquad F_T = F \exp\left(-\tfrac{1}{2}\sigma^2 T + \sigma\sqrt{T}\, Z\right). \tag{19.1}$$

Z is standard normal for every d, so a call on F_T has the same Black-76 price at every d, and its payoff has the same distribution — and therefore the same variance. Only the dimension of the integral changes. Anything that varies with d is then a property of the method and not of the problem.

Calculation 19.2 (The dimension, measured)**.** A one year at-the-money call, $F = K = 100$, $\sigma = 20\%$, worth 7.9656 at every dimension. Both methods are given the same budget of 10^5 payoff evaluations: the grid spends it as n^d nodes with n as large as that allows, the simulation as 10^5 paths.[1]

d	**nodes an axis**	**grid error**	**simulation error**
1	100000	0.00000	0.025
2	316	0.00001	0.033
3	46	0.026	0.039
4	17	0.007	0.039
5	10	0.294	0.012
6	6	6.134	0.016
7	5	5.942	0.028

[1] `numerics::DimensionTest`.

The grid is not merely better in one and two dimensions, it is in a different league — a deterministic rule converges in the mesh, not in the square root of a sample. By six dimensions it is wrong by three quarters of the price. The simulation column has no trend in it at all.

The middle rows wobble because n moves in integer steps and the payoff's kink lands between nodes differently each time; the collapse is the signal, not the ordering of $d = 3$ against $d = 4$.

The cliff has a location and a reason. The grid gets $n = \lfloor B^{1/d} \rfloor$ nodes an axis out of a budget B, so a budget spread over more axes leaves too few points on each to resolve anything: a hundred thousand evaluations is a hundred thousand nodes on one axis and five nodes on each of seven. Equivalently, to hold the mesh fixed the cost is n^d, which is the table at the top of this section read as an error rather than as a point count.

The simulation's flatness is exactly true here and only approximately true in practice. The standard error is $\sigma_{\text{payoff}}/\sqrt{N}$, and the dimension appears nowhere in it. Measured across $d = 1, 2, 4, 8, 16$ it is 0.0416 every time, to three figures. But (19.1) was built so that the payoff's own standard deviation is the same at every dimension, and that is the part a real problem does not give you: a basket over ten assets, or a market model with twenty rates, generally has a payoff whose variance grows with the number of factors. Dimension independence is a statement about the *exponent*. The constant is still the payoff's standard deviation, and that is a modelling quantity, not a numerical one.

Remark (What the rate means for a desk)**.** Four times the work to halve the error. A price good to a cent needs a hundred times the paths of one good to ten cents, and a Greek computed by bumping needs the difference of two such numbers, which is why Greeks are the expensive part of a simulation and why pathwise and adjoint methods exist.

Variance reduction — antithetics, control variates, importance sampling — improves the constant, sometimes by a large factor, and leaves the exponent alone. Quasi-Monte Carlo is the one technique that changes the rate, approaching $1/N$ when the effective dimension is low, which is a genuinely different proposition and worth reaching for when it applies.

Structure (Both methods are applying the same semigroup)**.** The two techniques look unrelated and are two ways of computing $P_t f = \mathbb{E}[f(X_t)]$, the semigroup of chapter 3.

A lattice or a grid discretises the *operator*: replace $\mathcal{L}$ by a matrix and $P_t = e^{t\mathcal{L}}$ by repeated multiplication. The error is a discretisation error, it is deterministic, and it shrinks with the mesh — and the matrix has one row per grid point, which is where the dimensional cost comes from.

A simulation samples the *measure* instead: draw from the law of X_t and average f. Nothing is discretised in the state, so nothing costs n^d, and the error is a sampling error that shrinks with the square root of the count.

Written that way the trade is not a matter of taste. One method pays for accuracy in dimensions and the other in variance, and early exercise is hard for simulation for a structural reason: the exercise decision needs $P_t f$ at an intermediate state, which is exactly the object a lattice computes and a simulation does not.

19.3 Arbitrage-Free Becomes a Property of the Estimator

A closed form has a property that is easy to overlook because it is never stated: its prices are mutually consistent by construction. Black-Scholes cannot produce a negative butterfly, because it

is an expectation under a measure, and an expectation of a convex function of a random variable is convex. There is nothing to check.

Once the expectation is estimated rather than evaluated, that guarantee has to be earned, and it is not automatic.

Calculation 19.3 (A butterfly priced two ways)**.** Take three strikes with K_2 midway between K_1 and K_3, and price each call by simulation. The payoff is convex in the strike, so for any single path

$$\big(S-K_1\big)^+ - 2\big(S-K_2\big)^+ + \big(S-K_3\big)^+ \;\geq\; 0. \tag{19.2}$$

Price the three on a *common* set of paths and (19.2) is inherited term by term: the estimated butterfly is an average of non-negative numbers and cannot be negative. Ten thousand paths — far too few for an accurate price — give exactly zero violations across every width tried.[2]

Price them on *independent* paths, which is what pricing each option separately amounts to, and nothing preserves it. Each price carries its own sampling error, the butterfly is a difference of nearly equal numbers, and at the same ten thousand paths over a quarter of narrow triples come out negative.[3] The price set admits a static arbitrage: one could buy the butterfly for a negative amount.

Remark (Why more paths is the wrong answer)**.** The instinct is to spend more. It does not work: sampling error falls as $N^{-1/2}$ while a butterfly's true value falls as the square of the strike spacing, so for any path budget there is a spacing at which the violations return.[4] Sixteen times the paths buys four times the resolution in strike, and a desk quoting a surface always wants more resolution than that.

Note also which direction the failure hides in. Widening the butterfly makes its value large against the noise and the violations disappear, so a check on widely spaced strikes passes while the surface a desk actually shows — adjacent strikes, tightly spaced — fails. A validation that samples coarsely will not find this.

Structure (Three disciplines in place of one guarantee)**.** Calculation 19.3 generalises.

Arbitrage-freeness in a solvable model is a *structural* property: it follows from the price being an expectation under a measure. In a numerical model it is an *estimator* property, and it survives only if three things are done deliberately.

- *Price from a model, not from prices.* A surface interpolated or learned directly from quotes has no measure behind it and no reason to be convex in strike or monotone in maturity. A simulation of an arbitrage-free process does, whatever else is wrong with it.

- *Share the randomness.* Common random numbers across strikes, maturities and scenarios is what carries pathwise relations like (19.2) through to the estimates. The same discipline is what makes a bumped Greek meaningful, since a delta from two independent simulations is mostly noise.

[2] `measured`.

[3] `measured`.

[4] `measured at two budgets and two widths`.

- *Correct the discretisation.* A discrete scheme does not preserve the martingale property exactly, so numeraire-relative assets drift. Chapter 8 measures the consequence: a Gaussian forward curve model simulated with the wrong drift stops repricing its own initial curve, by five basis points at a year and half a per cent at ten. The repair is to force the known quantities to reprice, which is what a control variate or a martingale correction does.

None of the three is expensive. All three are easy to omit, and each omission produces prices that look like prices.

19.4 Early Exercise, Where the Difficulty Concentrates

The two methods divide the world neatly except at one point, where most of the practical difficulty lives.

Early exercise needs the continuation value: at each exercise date, the holder compares what the option is worth if kept against what it pays if stopped. A lattice knows the continuation value exactly, because it has already solved the layer after. A simulation does not, because it runs forwards and the future of a path is not known when the decision is made — and using it anyway is not merely inaccurate, it is a different and much more valuable contract, one exercised with foresight.

The standard resolution is Longstaff and Schwartz's: estimate the continuation value by regressing realised discounted cashflows on functions of the current state, across paths, and exercise where intrinsic beats the estimate. It works, it is what essentially every desk uses for Bermudans, and it is biased.

Remark (Two biases, not one)**.** They are usually discussed as though there were one, and they have different causes, different signs, and different remedies.

Foresight bias, upwards. If the same paths are used to fit the regression and to value the option, the exercise rule has seen the outcomes it is deciding about. It is fitted to the noise as well as to the signal, and a rule fitted to noise flatters itself on the data it was fitted to. This is a finite sample effect and it decays as paths are added.

Policy bias, downwards. The estimated rule is not the optimal stopping rule — a quadratic in the spot cannot be — and any suboptimal rule undervalues the option. This has nothing to do with sample size. It is the basis's fault, and adding paths does not touch it.

The usual implementation contains both. Which one dominates depends on how many paths there are, and that is exactly what makes the situation confusing.

Example 19.1 (The numbers)**.** A one year Bermudan put, ten exercise dates, struck at the money, at a 25% volatility. The lattice answer is 7.915 against a European value of 7.459, so early exercise is worth about forty-six cents. Averaged over two hundred and forty runs:

Paths	Same paths	Fresh paths
500	+0.156	−0.096
2,000	+0.058	−0.046
8,000	+0.004	−0.031
16,000	−0.008	−0.024

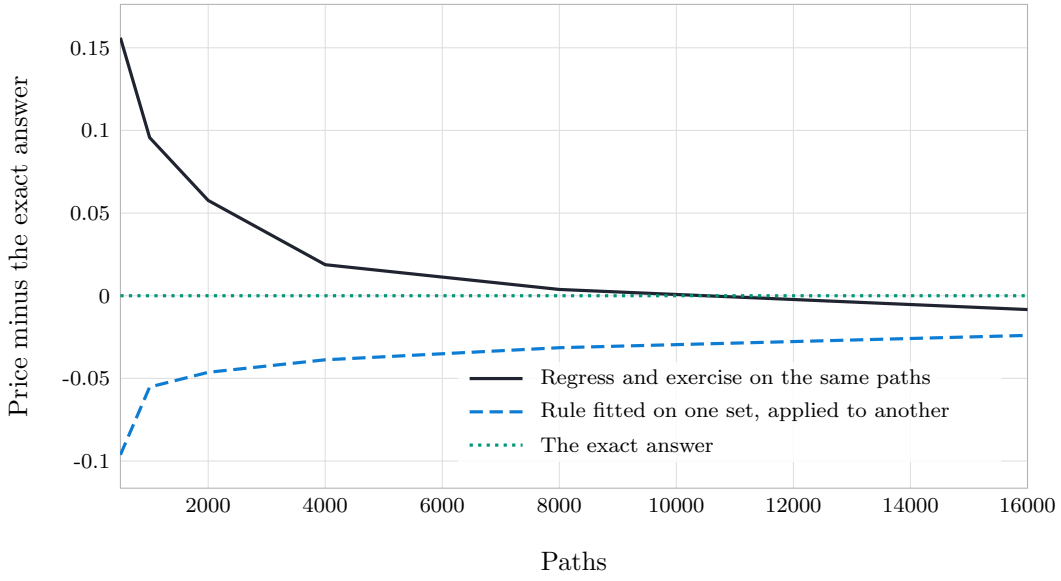

Figure 19.1: Longstaff-Schwartz against an exact lattice answer, averaged over two hundred and forty independent runs at each path count. The upper curve reuses its paths and starts fifteen cents high, falling through the exact answer as the foresight it enjoyed is priced out. The lower curve fits the rule on one set of paths and applies it to another, which removes the foresight and leaves only the loss from an imperfect exercise rule — so it stays below, and flattens rather than converging, because more paths cannot repair a basis that is too poor. The distance between the curves at any path count is the foresight; the distance from the lower curve to zero is the policy.
Drawn by `longstaff_schwartz` and `binomial_bermudan`.

Read the two columns against each other. The left one falls by a factor of twenty and changes sign; the right one falls by a factor of four and is levelling off. That is the two biases behaving as their causes predict.

Remark (One run cannot see its own bias)**.** The practical warning, and the reason the table above needed two hundred and forty runs per entry rather than one.

At a thousand paths the bias is about ten cents and the standard deviation of a single estimate is several times that. So a single Longstaff-Schwartz run compared against a lattice tells you nothing about which way the method leans — the answer will be above or below according to the seed, and both happen. A desk that validates its Bermudan pricer by running it once and eyeballing the difference has not validated anything, and `the test that pins this` exists because it is easy to believe otherwise.

Remark (The harder problem is choosing what to regress on)**.** The biases above are measurable, which makes them the easy part. The difficulty that actually decides whether a Bermudan price is any good is upstream of them: what should the continuation value be regressed *on*?

The method needs variables in which the continuation value is a function of the current state — Markovian in them — and it needs a basis in which that function is close to linear. For a Bermudan put on one asset both are easy: the state is the spot and a quadratic is adequate, which is why the example above behaves. For anything real neither is obvious.

A Bermudan swaption in the market model of chapter 13 has twenty forward rates as its state. One cannot regress on twenty variables with any hope of conditioning, so the modeller picks summary variables — the underlying swap rate, perhaps a slope, perhaps the first two principal components — and hopes the continuation value depends on the rest only weakly. A callable range accrual depends on the whole path of accruals so far, and the natural summary, the accrual accumulated to date, is an extra state variable that must be carried and regressed on. A callable with a barrier depends on whether the barrier has been touched, which is not a function of the current level at all.

And the error this causes is invisible from inside the run. If a relevant state variable has been left out, the regression still fits, the fit statistics still look reasonable, and the price comes out low by an amount nothing in the calculation reveals. It is the policy bias of the previous section with a cause that adding paths cannot touch and adding basis functions cannot touch either, because the missing information is not in the variables being regressed on.

The only real defences are a dual upper bound, which brackets whatever the policy lost, and pricing the same trade with a genuinely different state — for instance on a lattice in a low-factor model — and treating the gap as the model risk it is. Which is chapter 20's discipline again, arriving from a numerical direction rather than a statistical one.

Remark (Neither is an upper bound)**.** A tempting conclusion from the figure is that the two curves bracket the answer. They do here, and it is not a bracket one can rely on: the upper curve's position depends on the path count, and at sixteen thousand paths it has already crossed below.

A genuine upper bound needs a different idea. The dual formulation prices the option as a minimisation over martingales rather than a maximisation over stopping times, and any martingale one can write down gives a valid upper bound — so a rough one from the regression already in hand gives a true bracket. Andersen and Broadie made this practical. The bound converts "our Bermudan number is probably about right" into an interval.

19.5 What Lifting the Constraint Permits

Now the constructive half. If a model no longer needs a formula, what can it be?

Models with more parameters than quotes

Neural stochastic differential equations, market generators, and non-parametric local-stochastic volatility all abandon the idea that a model is a handful of interpretable numbers. The dynamics are specified by something with thousands or millions of parameters, fitted to whatever data is available, and priced by simulation because nothing else is possible.

Where the parameters are put matters more than how many there are, and chapter 2 settles the question. A continuous arbitrage-free price is an Itô diffusion with a drift fixed by no-arbitrage, so the only object left to parametrise is the diffusion coefficient — and a network placed there yields an arbitrage-free model at every parameter value, with the constraint carried by the form of the equation rather than by a penalty in the loss. That is the argument for learning coefficients instead of learning prices, and chapter 21 makes it in full. It also explains why these methods are simulation methods necessarily: a learned coefficient admits no closed form by construction, so the model is defined by the equation and priced by solving it.

The calibration problem then changes shape rather than disappearing. What is needed is not a closed form but a *fast approximation of the map from parameters to prices*, which can be learned once, offline, and then evaluated in microseconds — so the model is free to be slow, provided its price map is smooth enough to be learned. Chapter 21 discusses that technique under deep calibration; here the point is what it unlocks, which is the freedom to choose dynamics on their merits.

Learning the exercise rule instead of the continuation value

The regression above estimates a continuation value and derives a policy from it. Reinforcement learning inverts that: the policy is the object being optimised, and the value is whatever the policy achieves.

This is a genuine improvement in one respect — a policy can be represented by something far richer than a quadratic, so the policy bias above can be made small — and it changes nothing structural. A learned policy is still a policy, so the estimate is still biased low, and it still needs a dual bound to be turned into an interval. The bias moves; it does not go away.

Optimising the hedge instead of the price

The methods above still price first and hedge afterwards. Deep hedging drops that order, and it is the one item on this list that changes what is being computed rather than how.

Fix a risk measure ρ — expected shortfall, from chapter 24, is the usual choice and the one its axioms recommend. Parametrise the hedging strategy directly as a function of whatever is observable, hand it a simulated market, and choose its parameters to minimise ρ of the terminal profit and loss. The price falls out as the cash that makes the optimised risk acceptable, which is an indifference price rather than a replication cost.

What that buys is everything the replication argument had to assume away. Transaction costs, position limits, discrete rebalancing and an incomplete market are not obstructions to be handled

afterwards but terms in the objective, so the strategy is optimised against the market it will actually trade in. Chapter 5 measured what discrete rebalancing alone leaves behind — a residual whose spread falls only as the square root of the rebalancing count — and the delta it was rebalancing was optimal for a market with none of these features.

Two things it does not do, and both matter more than the machinery.

It does not produce a no-arbitrage price, and cannot. The answer depends on the risk measure and on how much risk is being tolerated, so two desks with different objectives get different numbers and neither is wrong. That is not a defect of the method; it is what an incomplete market means, and chapter 4's theorem said as much long before anyone trained a network. What the method supplies is a principled way to pick a point in the interval the theorem leaves open.

And it inherits the market it was trained on. A strategy optimised against simulated paths learns the measure that generated them, so every defect of the generator becomes a defect of the hedge — which is why the market generators of the first subsection are not a separate topic from this one. The calibration problem has been moved rather than removed, and it has been moved somewhere harder to inspect: a mis-specified parameter is visible, and a mis-specified simulator is a distribution nobody has plotted.

Solving high-dimensional equations directly

The dimensional wall is where the newer numerical methods have most clearly earned their place, and the leading one says something about pricing that the partial differential equation hides.

Definition 19.4 (The pricing problem as a backward equation)**.** A European claim's value Y_t and its hedge satisfy the backward stochastic differential equation

$$Y_t = g(X_T) + \int_t^T f\big(s, X_s, Y_s, Z_s\big)\, ds \;-\; \int_t^T Z_s \cdot dW_s, \tag{19.3}$$

where g is the payoff, f carries the discounting and any funding or default term, and Z_t is whatever integrand makes the equation hold.

Two features of (19.3) deserve emphasis. It runs *backwards* — the terminal condition is given and the initial value is the unknown — which is why it is the natural form of a pricing problem. And Z is not an auxiliary variable: by comparison with chapter 4's replication argument, $Z_t = \sigma^\top \nabla_x Y_t$ is the hedge. So (19.3) states the price and the hedging strategy as one object, which the pricing equation does only implicitly.

Remark (What a deep solver does with it)**.** The construction of E, Han and Jentzen turns (19.3) into a fitting problem, and does it by reading the equation forwards.

Parametrise the unknown initial value Y_0 as a number and the integrand $Z_t = \zeta_\theta(t, X_t)$ as a network. Then simulate X forward and propagate Y alongside it using (19.3) as a recursion,

$$Y_{t+\Delta} = Y_t - f\big(t, X_t, Y_t, Z_t\big)\Delta + Z_t \cdot \Delta W_t,$$

which requires no knowledge of the answer. At the horizon the equation demands $Y_T = g(X_T)$, and in general the propagated Y_T will not equal the payoff. So minimise

$$\mathbb{E}\Big[\big(Y_T - g(X_T)\big)^2\Big] \tag{19.4}$$

over Y_0 and θ. The loss is the terminal condition's failure, the fitted Y_0 is the price, and the fitted network is the hedge.

What has been bought is that nothing is discretised in the state. The cost is $O(\text{paths} \times \text{steps})$ regardless of dimension, which is why this works at fifty factors and a grid does not. What has been given up is the subject of the next section.

The honest test of such a solution is the same as for anything else in these notes: report the residual of the equation at points the fit never saw. A network that satisfies its equation only where it was trained has learned the training set.

States that are not vectors

Some problems have a state that is a graph rather than a list of numbers — a portfolio of counterparties with exposures between them, a limit order book, a network of related instruments. Graph neural networks are the natural representation, and this is the least mature of the strands here: the applications are real but the track record is short, and it is worth treating claims in this area with more suspicion than the rest.

19.6 Why This Matters for Trading and Not Only for Pricing

A natural objection to everything above is that a model which cannot be calibrated in milliseconds cannot be used. That is true for a market making desk quoting a screen, and it is not true generally.

Chapter 22's trades do not need a model that prices in microseconds. They need a model whose view of the future is better than the market's, because the entire proposition is that the market's price implies something implausible and the position collects when it stops doing so. For that, being right matters and being fast does not.

Remark (The division of labour)**.** This suggests a use of models that the earlier chapters have not needed. A desk can reasonably run two: a tractable one for marking, quoting and hedging, where consistency with the market and speed are everything, and a slower and more honest one for deciding what to hold, where realism is everything.

They will disagree, and the disagreement is not a bug to be reconciled. It is the trade. Chapter 20's discipline applies directly — price it in two models that disagree about the thing the payoff is sensitive to — with the difference that here the disagreement is being sought rather than reserved against.

19.7 What Computation Does Not Fix

Two limits, both of which the earlier chapters have already established and neither of which more compute touches.

The first is identifiability. Chapter 20 measured a SABR calibration in which the quotes fixed the level of the smile to one percent and left the correlation free to move across most of its range — with the model exactly right and the data noiseless. That flatness is a property of the instruments quoted, not of the fitting procedure, and a model with a million parameters fitted to forty quotes is that problem in its extreme form. A richer model does not extract information the market did not supply; it distributes the same information over more unknowns.

The second is that every estimate of an exercise policy is biased, by the argument of this chapter, and no amount of computation makes a policy optimal without knowing the optimal policy. Compute changes the size of the bias and the tightness of the bound. It does not remove either.

The third is the one this chapter's argument turns on, and it is not about accuracy but about *knowing* the accuracy.

Structure (Bounded and unbounded errors)**.** A grid's error can be bounded before it is run. A finite difference scheme of order p has error $O(h^p)$ with a constant depending on derivatives of the solution, so halving the mesh and watching the answer move by a factor of 2^p is a check that the bound is operative — and Richardson extrapolation turns two runs into an estimate of the error itself. The same is true of a Monte Carlo: the central limit theorem supplies a standard error from the sample, and it is a genuine confidence statement about the number produced.

A network fitted to (19.4) offers neither. The loss is a residual, and a small residual does not bound the error in the solution: the map from residual to error involves a stability constant for the equation that is not available in the high-dimensional problems the method exists to solve. Refining does not help, because there is no mesh to refine — one trains longer, or with more capacity, and the answer moves by an amount with no theory attached.

So the asymmetry is not that one method is more accurate. Over four dimensions the network is more accurate, sometimes by a wide margin. The asymmetry is that one method's error is a quantity and the other's is a hope, and that difference lands squarely on chapter 24: a valuation uncertainty has to be reserved against, and a reserve requires a number. An unbounded error cannot be reserved for; it can only be provisioned against by judgement, which is the same as saying it is carried as model risk.

That is the residual value of solvability. A solvable model is not more realistic. It is not faster in any way that matters now. What it has is that its answer comes with an error one can write down, and a desk that has to reserve against being wrong finds that this is worth something. The trade is realism for auditability, and stating it that way makes it a decision rather than a habit.

Remark (The rule that survives)**.** The methods in this chapter will date faster than anything else in these notes. What will not is the habit of asking, of any number produced by a numerical method, three questions: what is its standard error, which way does it lean, and what would it look like if the method were wrong.

References

- Longstaff, F. A., & Schwartz, E. S. (2001). Valuing American options by simulation: a simple least-squares approach. *Review of Financial Studies*, 14(1), 113–147.

- Glasserman, P. (2004). *Monte Carlo Methods in Financial Engineering*. Springer.

- Andersen, L., & Broadie, M. (2004). Primal-dual simulation algorithm for pricing multidimensional American options. *Management Science*, 50(9), 1222–1234.

- Rogers, L. C. G. (2002). Monte Carlo valuation of American options. *Mathematical Finance*, 12(3), 271–286.

- Tavella, D., & Randall, C. (2000). *Pricing Financial Instruments: The Finite Difference Method*. Wiley.

- Han, J., Jentzen, A., & E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *PNAS*, 115(34), 8505–8510.