The Projected-Generator Method for Path-Dependent Derivative Pricing

Deterministic Pricing and Greeks for American and Exotic Options — Without Simulation, Trees, or Regression

Tamás Nagy, Ph.D.

tnagyphd@gmail.com

Working Paper

Build-std v2.0 | 19:38 | bbd15eec

Practitioner's Summary

Pricing a path-dependent contract — an American option, an autocallable, a barrier note — usually forces a desk into one of two camps. Either build a deterministic PDE grid, which is fast and repeatable but blows up as soon as you add a second or third risk factor, or run a Monte Carlo simulation, which handles any contract but returns a slightly different number every time it runs and needs a noisy regression step to value early exercise.

This paper develops a third option. The idea is to represent the model's "generator" — the operator that encodes how the underlying moves — as a finite matrix, and then price by evolving that matrix rather than by simulating paths. Continuation values are propagated by matrix multiplication; early exercise is a pointwise comparison; risk sensitivities (Greeks) fall out of the same expansion. No random paths, no regression, no time-stepping stability limit.

What a practitioner gets from this is determinism with a stated error budget. The engine returns the same price and the same Greeks on every call — there is no run-to-run noise to reconcile, and Greeks come without re-pricing the book under bumped inputs. When the model and basis are well-matched the price is exact on the relevant subspace; when they are not, the method reports the size of its own approximation gap rather than hiding it. For the one-to-five-factor problems that cover most equity, rates, FX, and commodity desks, it is typically far faster than simulation at comparable accuracy, and fast enough for intraday pricing and risk.

It is not a universal replacement for Monte Carlo, and the paper is explicit about where it stops. It needs a model whose generator can be written down and projected onto a basis — Black-Scholes, Heston, affine and mean-reverting models, and finite-activity jump-diffusions qualify. It loses its edge on high-dimensional baskets (beyond roughly five to six factors), on sharply discontinuous payoffs that trigger oscillation, and on rough-volatility dynamics unless a controlled approximation is applied first. For those regimes, simulation remains the right tool. The contribution is a clean, benchmarked, deterministic primitive for the large middle ground where it wins, with its limits stated rather than glossed over.

Abstract

Derivative pricing has a clear mathematical target: compute the discounted risk-neutral value of a payoff under a specified model. For European vanilla contracts this target is often analytic or nearly analytic. The difficult cases are path-dependent derivatives — American and Bermudan exercise,

barriers, autocallables, coupon memory, hit counts, and structured-product state — where standard methods rely on Monte Carlo simulation, regression continuation estimates, or high-dimensional PDE grids.

This paper introduces projected-generator pricing as a finite-dimensional operator method for such contracts. Starting from the infinitesimal generator of the risk-neutral dynamics, the method projects the generator onto a finite basis and propagates continuation values by the resulting matrix semigroup. Path dependence is represented by finite event-state transitions around the same propagation primitive. American and Bermudan exercise become projected semigroup propagation plus pointwise exercise comparison plus re-projection, with no simulated-path regression.

The contribution is a pricing architecture rather than a new isolated ingredient. Generator calculus, Galerkin projection, matrix exponentials, spectral bases, and Bellman recursion are combined into a deterministic engine for path-dependent derivatives. When the finite space is invariant under the generator, pricing is exact on that space; otherwise the projection residual measures approximation debt. For analytic payoffs under spectral bases the error decays exponentially, and Greeks are obtained by differentiating the same finite expansion rather than by bump-and-reprice.

Benchmarks against Black-Scholes, COS, Longstaff-Schwartz, finite differences, Monte Carlo, and Merton jump-diffusion references show machine-precision European pricing, regression-free American pricing within 1% of published references, deterministic Greeks, finite-activity Poisson-jump support through a nonlocal Lévy generator, and a structured-product example priced without path simulation. The method is strongest when the model has a tractable generator, the path memory is finite state, and the effective dimension is low to moderate. High-dimensional baskets, rough volatility without a controlled lift, and severe discontinuities remain better served by Monte Carlo, finite differences, or adapted bases.

1. Introduction

Derivative pricing is a fundamental subdomain of quantitative finance. It has a simple target: given a model, a payoff, and a discount curve, compute the model value of the derivative contract.

For simpler derivatives like European vanilla options the payoff depends on the final state of the stochastic processes, so the value (the price) is often available in closed form, by Fourier methods, or by fast one-dimensional numerical integration.

The problem becomes harder when the payoff is path dependent: the value (price) depends not only on the terminal state, but also on how the path arrived there. Typical examples are American and Bermudan contracts, which require a continuation value at each exercise date. Barriers, autocalls, coupon memory, hit counts, and lookbacks require remembering what happened along the path.

Some existing methods solve parts of this problem well, but each pays a different price.

- Monte Carlo simulation is the most robust and general, but its statistical error decreases only as $O(P^{-1/2})$, where P is the number of simulated paths.
- Early exercise requires estimation of a continuation value, typically by Longstaff-Schwartz regression (Longstaff and Schwartz, 2001).
- Finite difference methods remove path noise, but replace it with grid design, boundary conditions, stability constraints, and growth on the order of $O(N_x^d)$, where N_x is the number of grid points per state dimension and d is the state dimension. The practical difficulty is that each additional risk factor multiplies the grid, while discontinuities and exercise boundaries

require fine local resolution.

The question is how path dependency and continuation values can be propagated without simulated paths, regression, or a full state-space grid.

This paper introduces a new method for pricing more complex, path-dependent derivatives using the infinitesimal generator of the risk-neutral dynamics as the primary computational object. This generator is the local law behind both the forward evolution of probability distributions (forward PDE, Fokker-Planck) and the backward equation for expected values (backward PDE, Kolmogorov).

We project that generator operator onto a finite basis, obtaining a generator matrix. This transforms the pricing problem into a finite-dimensional linear-algebra problem. Continuation values then propagate by finite matrix evolution in coefficient space. The continuous stochastic system remains, but its action on the chosen function space is represented by linear algebra.

Path dependence is handled by finite event states around the same propagation primitive.

American and Bermudan exercise become a repeated sequence:

- propagate the continuation value
- compare it with the exercise value
- project the result back into the finite basis.
- Greeks can be obtained by differentiating the propagated expansion rather than by bump-and-reprice.

Before showing the detailed machinery, it is useful to state the scope clearly.

What this paper is.

- A unified operator-algebraic architecture for deterministic pricing of certain path-dependent derivatives.
- Benchmarks cover European, American, and Phoenix autocallable contracts against established references.

What this paper is not.

- It is not a claim that the projected generator replaces Monte Carlo universally.
- The method requires a tractable generator in closed form and a basis that approximates its action. GBM, Heston, affine, mean-reverting models, and finite-activity Poisson jump-diffusions with known Lévy symbols satisfy this requirement. Arbitrary state-dependent jump systems, rough-volatility dynamics without a controlled lift, and non-analytic Lévy measures remain outside the direct scope.
- It is not a full production pricing system: live market calibration, multi-curve discounting, and regulatory capital integration are outside scope. These topics are left for companion implementation and product papers.

2. Existing Methods and the Missing Primitive for Path-Dependent State Evolution

Before introducing a new method, we first isolate the missing mathematical object: a primitive for path-dependent state evolution that makes contract memory trackable and computable.

2.1 Monte Carlo

Monte Carlo estimates $\mathbb{E}[g(X_T)]$ by random sampling (Broadie and Glasserman, 2004), producing an error of $O(P^{-1/2})$ for P paths. The projected generator computes deterministic coefficient evolution with zero sampling variance. The cost of this determinism: the method requires an analytic generator and a basis that approximates its action well. Monte Carlo asks nothing of the model beyond sampleability — it prices any payoff on any dynamics, provided you can simulate. When the model is exotic or high-dimensional beyond tractable tensor decomposition, Monte Carlo remains the only general-purpose tool.

2.2 Longstaff-Schwartz

Longstaff-Schwartz estimates the continuation value through cross-sectional regression on simulated paths. The projected generator computes this continuation value via semigroup propagation. Both use finite function spaces. The difference is what those spaces do: Longstaff-Schwartz fits continuation values from random data (a statistical problem), whereas the projected generator evolves continuation values from the model operator (a linear algebra problem). Longstaff-Schwartz inherits its Monte Carlo's generality — it handles arbitrary dynamics and high-dimensional baskets without requiring the generator in closed form. The projected generator inherits spectral methods' precision — it eliminates regression noise entirely, at the cost of needing an explicit operator representation.

2.3 Finite Differences

Finite difference methods approximate derivatives locally on a spatial grid. Projected generator pricing approximates the operator globally within a basis. Both are deterministic. Both suffer in high dimensions. The distinction is how dimensionality bites: finite differences hit the $O(N_x^d)$ grid explosion directly, while the projected generator defers it through tensor decomposition (§5.7), buying tractability up to $d \approx 5$ factors on a single machine. Below 3 dimensions, finite differences are simpler, faster, and well-understood — the projected generator offers no advantage for vanilla 1D or 2D problems where a fine grid is cheap.

2.4 COS and Spectral Methods

COS methods (Fang and Oosterlee, 2008) are a specific instance of the projected-generator framework: a Fourier-cosine basis computes the characteristic function of the transition density, then reconstructs prices by inverse transform. The projected-generator view generalizes this by allowing any finite basis where the generator matrix M can be computed, rather than relying solely on Fourier expansions of characteristic functions. COS achieves machine precision for European options under affine models — our §7.3 benchmarks reproduce this exactly. The projected generator adds backward induction for early exercise, event-state propagation for path dependence, and basis flexibility for non-affine dynamics.

2.5 Polynomial Diffusion Literature

The invariant-subspace property — that polynomial diffusions preserve finite polynomial spaces, enabling moment computation via matrix exponentials — was established by Cuchiero, Keller-Ressel, and Teichmann (2012) and systematically developed by Filipović and Larsson (2016). Ackerer and Filipović (2020) extended this to European option pricing via orthogonal polynomial density expansions. The present work formalizes and mechanically verifies these results (Part 20, Theorems

T142–T148), but the mathematical content of polynomial invariance and exact moment evolution is theirs.

2.6 What This Paper Adds

What the projected-generator framework adds to the prior literature is threefold:

- **Path-dependent pricing.** Cuchiero et al. and Ackerer-Filipović treat moments and European payoffs. The projected generator extends the same invariance to American exercise (backward propagation + max re-projection), barrier events (first-hit mass accounting), autocallable coupons (event-state automata), and graph-structured state machines (Part 16).

  Part 24 (Theorem T178) proves the full end-to-end certificate: for polynomial diffusions with finite event-state contracts, spatial, temporal, and graph errors are all exactly zero — the only remaining error is payoff truncation, which contracts geometrically. None of these path-dependent extensions are present in the polynomial diffusion literature.

- **Deterministic error certificates.** The polynomial diffusion literature observes exactness when invariance holds. The projected generator packages the residual, truncation bound, and convergence rate into a computable certificate that accompanies every price — including the case where invariance fails and approximation debt is explicit.

- **Mechanized verification.** The formal proof suite (274 theorems, Lean 4 export) verifies the algebraic implications of polynomial invariance, complexity separation, the regularity hierarchy, the path-dependent certificate, method-selection boundaries, the rough-volatility lift, the transform-method subsumption, the Greek certificates, and certified value iteration.

  Furthermore, a derivation layer reduces the semantic gap by proving previously-assumed key facts from established structure. This represents a new verification category in computational finance.

2.7 Novelty Boundary

Galerkin projection, spectral methods, COS pricing, polynomial diffusion theory, finite differences, and matrix exponentials are individually well-known. The contribution is their unification into a single projected-generator architecture for derivative pricing:

- the generator is the primary object, not the PDE grid or simulated path set;
- continuation values are propagated by the finite generator representation rather than fitted by regression;
- American exercise, barrier events, coupon memory, and Greeks are expressed as finite operator steps around the same propagation primitive;
- exactness is separated from convergence: invariant finite subspaces give exact closure, while non-invariant spaces carry explicit residual debt;
- the proof layer verifies the algebraic closure implications, while the engine layer supplies reproducible numerical validation.

This is a unification and implementation contribution, not a replacement claim against the literature on spectral PDE methods.

3. The Projected Generator as the Basic Element

The gap in existing methods points to the basic element we need: not simulated paths or a grid, but a finite representation of the generator itself.

3.1 The Central Role of the Infinitesimal Generator

Before introducing the projection mechanism, we must first shift our perspective from individual random paths to the operator that governs their expected evolution. Consider an Itô diffusion X_t governed by:

$$dX_t = \mu(X_t)dt + \sigma(X_t)dW_t.$$

For any smooth test function f, Itô's lemma gives:

$$df(X_t) = f'(X_t)\mu(X_t)dt + \frac{1}{2}f''(X_t)\sigma^2(X_t)dt + f'(X_t)\sigma(X_t)dW_t.$$

(Remark: This truncation at the second derivative is exact, not an approximation. Because Brownian motion scales as $dW_t \sim \sqrt{dt}$, its square contributes $(dW_t)^2 \sim dt$, surviving the limit $dt \to 0$. Higher-order terms scale as $dt^{1.5}$ or higher and vanish strictly.)

The equation splits the change into a deterministic drift-and-spread component (the dt terms) and a noisy component (the dW_t term). Because derivative pricing is ultimately about computing expected values, we take the conditional expectation of both sides.

The noisy term $f'(X_t)\sigma(X_t)dW_t$ is a martingale increment: because the Brownian step dW_t is equally likely to be positive or negative, its expected value is strictly zero ($\mathbb{E}[dW_t] = 0$). The noise cancels out in expectation.

Dividing by dt and taking the limit $dt \to 0$ yields the instantaneous expected rate of change (see, e.g., Øksendal, 2003):

$$\lim_{\Delta t \downarrow 0} \frac{\mathbb{E}[f(X_{t+\Delta t}) \mid X_t = x] - f(x)}{\Delta t} = \mu(x)f'(x) + \frac{1}{2}\sigma^2(x)f''(x) \equiv \mathcal{L}f(x).$$

The operator $\mathcal{L}$ is the **infinitesimal generator** of the stochastic process. It is the fundamental object that links the random paths of the SDE to deterministic partial differential equations.

This formulation places derivative pricing within a deep operator-theoretic framework. The continuous-time stochastic world offers three equivalent views of the same reality, all branching from the SDE and governed by the generator:

1. **The SDE** ($dX_t = \mu dt + \sigma dW_t$) tracks individual random paths forward in time. This is the stochastic dynamics view.
2. **The Fokker-Planck equation** (or Kolmogorov forward equation, $\partial_t p = \mathcal{L}^* p$) tracks the forward evolution of the probability density p. It asks: "how does the probability mass flow?" This is driven by the adjoint operator $\mathcal{L}^*$. The adjoint is the function-space equivalent of a matrix transpose: it satisfies $\langle \mathcal{L}f, p\rangle = \langle f, \mathcal{L}^* p\rangle$. By applying integration by parts twice to shift the derivatives from f onto p, we find the explicit forward equation:

$$\mathcal{L}^* p = -\partial_x(\mu p) + \frac{1}{2}\partial_x^2(\sigma^2 p)$$

3. **The Kolmogorov backward equation** ($-\partial_t V = \mathcal{L}V - rV$) tracks the backward evolution of the expected value V. It asks: "what is the current value of a future payoff?" This is driven directly by the generator $\mathcal{L}$.

The **Feynman-Kac formula** acts as the bridge between these views. It proves that the solution to the backward PDE is exactly the discounted expectation along the SDE paths under the risk-neutral measure: $V(x,t) = \mathbb{E}[e^{-r(T-t)}g(X_T) \mid X_t = x]$.

At the heart of this entire structure sits the generator $\mathcal{L}$. It appears directly in the backward PDE, it forms the basis of the Feynman-Kac expectation, and its adjoint $\mathcal{L}^*$ drives the forward density. Backward and forward evolution are mathematically dual to each other in a Hilbert space sense ($\mathcal{L} \leftrightarrow \mathcal{L}^*$).

Computationally, these views lead to vastly different algorithms:

- Simulating the SDE leads to Monte Carlo methods
- Discretizing the PDE on a grid leads to finite difference methods.

But by recognizing that $\mathcal{L}$ is simply a linear operator acting on a Hilbert space of functions, we can solve the pricing problem using functional analysis: we project the operator itself into a finite basis using Galerkin methods. This is the natural computational realization of the deep operator structure underlying the stochastic calculus.

3.2 The Finite-Dimensional Ansatz

The generator $\mathcal{L}$ is a continuous differential operator living in a Hilbert space with infinite dimension.

To compute with it, we must project it into a finite-dimensional space.

The goal is to replace

- continuous function $V(x,t)$ with a finite vector of coefficients $\mathbf{c}(t)$, and
- differential operator $\mathcal{L}$ with a finite matrix M

Choose a computational domain $[a,b]$ that contains the economically relevant range of the state variable (wide enough that boundary effects are negligible, e.g. $[0, 2.5K]$ for a put struck at K), and a finite basis $\{\phi_0, \dots, \phi_{N-1}\}$ of functions on that domain. The choice of N determines the spatial resolution, balancing the ability to capture sharp payoff features against the computational cost of larger matrices. Approximate the value function as:

$$V_N(x,t) = \sum_{k=0}^{N-1} c_k(t)\phi_k(x). \tag{2}$$

This expansion separates variables:

- the spatial geometry is captured by the fixed basis functions $\phi_k(x)$, while
- the time evolution is entirely captured by the time-dependent scalar coefficients $c_k(t)$. These N coefficients form a column vector $\mathbf{c}(t) = [c_0(t), c_1(t), \dots, c_{N-1}(t)]^T$.

The goal of the method is to find the equations that govern how this vector $\mathbf{c}(t)$ changes over time.

The methodology is agnostic to the choice of basis family.

Common choices include:

- simple monomials $\{1, x, x^2, ...\}$ (transparent algebra, but unstable for large N due to Runge's phenomenon);
- Chebyshev polynomials (stable on bounded domains, the default for production implementations);
- Fourier cosine functions $\{\cos(k\pi x)\}$ (natural when the characteristic function is available — the COS method of Fang and Oosterlee (2008) is a specific instance);
- Hermite functions (natural for Gaussian diffusions such as Ornstein-Uhlenbeck);
- and finite-element or spline bases (effective for discontinuous payoffs such as barriers).

The choice affects only the numerical values in the matrix M and the convergence rate; the algorithm itself — project, propagate, exercise, re-project — is identical regardless of basis family.

Appendix A uses monomials for the hand calculation because the arithmetic is readable; Sections 5 and 7 analyze convergence and failure modes for different basis families.

3.3 The Galerkin Projection: Building the Operator Matrix M

Because V_N is restricted to a finite subspace, it cannot satisfy the PDE exactly; there will be a residual error. The Galerkin method (Galerkin, 1915; Boyd, 2001) handles this by demanding that this residual error be orthogonal to every basis function in the subspace. Physically, this means the error has no "shadow" or projection in the space we care about.

A note on naming, now that the operation is defined: throughout this paper, "projected" names this invariant operation — the Galerkin projection of $\mathcal{L}$ onto the finite basis — while "spectral" names a *class of bases* (Fourier, Chebyshev, Hermite) whose projection error decays exponentially for analytic payoffs (analyzed in §5.2).

Substituting the ansatz (2) into the backward PDE (1) yields:

$$\sum_{k=0}^{N-1} \dot{c}_k(t)\phi_k(x) + \mathcal{L}\left(\sum_{k=0}^{N-1} c_k(t)\phi_k(x)\right) - r\sum_{k=0}^{N-1} c_k(t)\phi_k(x) = \text{residual}.$$

Because the generator $\mathcal{L}$ is a linear spatial operator (it consists of derivatives with respect to x, not t), it treats the time-dependent coefficients $c_k(t)$ as constants. We can therefore pull the summation and the scalar coefficients outside the operator:

$$\sum_{k=0}^{N-1} \dot{c}_k(t)\phi_k(x) + \sum_{k=0}^{N-1} c_k(t)\mathcal{L}\phi_k(x) - r\sum_{k=0}^{N-1} c_k(t)\phi_k(x) = \text{residual}.$$

Imposing Galerkin orthogonality means taking the standard Lebesgue L^2 inner product ($\langle f, g\rangle = \int_a^b f(x)g(x)dx$) of this entire equation with each basis function ϕ_j, and setting the result to zero:

$$\sum_{k=0}^{N-1} \dot{c}_k(t)\langle\phi_j, \phi_k\rangle + \sum_{k=0}^{N-1} c_k(t)\langle\phi_j, \mathcal{L}\phi_k\rangle - r\sum_{k=0}^{N-1} c_k(t)\langle\phi_j, \phi_k\rangle = 0. \tag{3}$$

If we choose an orthonormal basis (like Chebyshev or Fourier), the first and third terms simplify because $\langle \phi_j, \phi_k \rangle = \delta_{jk}$ (1 if $j = k$, 0 otherwise). The time derivative isolates $\dot{c}_j(t)$, the discount term isolates $-rc_j(t)$.

Crucially, look at the middle term: the coefficients $c_k(t)$ sit *outside* the inner product. The inner product itself, $\langle \phi_j, \mathcal{L}\phi_k \rangle$, depends only on the basis functions and the generator. This defines the matrix M:

$$M_{jk} = \langle \phi_j, \mathcal{L}\phi_k \rangle = \int_a^b \phi_j(x) \mathcal{L}\phi_k(x) dx.$$

The matrix M is the exact matrix representation of the continuous generator operator $\mathcal{L}$ in the chosen basis. Just as a geometric rotation can be represented by a 3×3 matrix once X, Y, Z axes are chosen, the differential operator $\mathcal{L}$ is represented by the $N \times N$ matrix M once the N basis functions are chosen.

The matrix ODE now emerges directly. Equation (3) is not one equation but N of them — one for each test function ϕ_j ($j = 0, \ldots, N-1$). In every term a coefficient c_k (or its time derivative $\dot{c}_k$) is multiplied by a fixed inner product and summed over k — exactly the pattern of a matrix acting on the coefficient vector $\mathbf{c}(t)$. We define the Gram matrix G and the operator matrix M as:

$$G_{jk} = \langle \phi_j, \phi_k \rangle \quad \text{and} \quad M_{jk} = \langle \phi_j, \mathcal{L}\phi_k \rangle$$

The time-derivative term then becomes $(G\dot{\mathbf{c}})_j$, the generator term is $(M\mathbf{c})_j$, and the discount term is $r(G\mathbf{c})_j$. Stacking the N equations collapses the index j and leaves a single vector ODE for the coefficients:

$$G\dot{\mathbf{c}}(t) + M\mathbf{c}(t) - rG\mathbf{c}(t) = 0,$$

where G is the Gram matrix.[1] Multiplying by G^{-1} gives:

$$\dot{\mathbf{c}}(t) + G^{-1}M\mathbf{c}(t) - r\mathbf{c}(t) = 0,$$

which we write compactly as:

$$\dot{\mathbf{c}}(t) = A\mathbf{c}(t),$$

where the combined operator is $A = G^{-1}M - rI$ (with I being the identity matrix) and the matrix elements of the pure generator M are:

$$M_{jk} = \int_a^b \phi_j(x) \left[\mu(x)\phi_k'(x) + \frac{1}{2}\sigma^2(x)\phi_k''(x) \right] dx.$$

[1]The Gram matrix is the table of all pairwise inner products of the basis functions, $G_{jk} = \langle \phi_j, \phi_k \rangle$. It measures how non-orthogonal the basis is: it is symmetric and positive-definite for any linearly independent basis, and reduces to the identity exactly when the basis is orthonormal — in which case G^{-1} drops out entirely. The same object appears as the *mass matrix* in finite-element analysis.

3.4 Decoupling Market Dynamics (M) from the Payoff

The structure of M reveals a powerful architectural decoupling:

- the matrix M depends **only on the market dynamics** (the drift $\mu(x)$ and volatility $\sigma(x)$ from the SDE) and the chosen basis.
- It does **not** depend on the option's payoff function f, its strike price, or its maturity.

Computationally, this means the integrals for M do not need to be evaluated symbolically or repeatedly.

In practice, they are computed via numerical quadrature. Quadrature replaces a continuous integral with a weighted sum evaluated at specific points: $\int_a^b F(x)dx \approx \sum_{q=1}^{Q} w_q F(x_q)$, where $F(x)$ is any function to be integrated (in our case, the product $\phi_j(x)\mathcal{L}\phi_k(x)$), x_q are the evaluation points, and w_q are the corresponding weights (e.g., Gauss-Legendre quadrature). This reduces the entire integration step to a single, highly optimized matrix multiplication of pure numbers: $M = \Phi \cdot \text{diag}(\mathbf{w}) \cdot (\mathcal{L}\Phi)^T$, where Φ is an $N \times Q$ matrix of numbers containing the basis functions evaluated at the Q quadrature points, and $\mathbf{w}$ is the vector of numerical weights. Once this multiplication is done, M is just a dense $N \times N$ matrix of floating-point numbers; all symbolic functions and differential operators have been completely eliminated.

Because M is independent of the specific option contract,

- it can be pre-computed once per market calibration.
- Changing the model simply changes the numerical values in the matrix M.

The rest of the pricing engine—the matrix exponential, the backward time-stepping, and the handling of early exercise—remains completely agnostic to the underlying dynamics.

Unlike finite difference methods that require grid redesigns and stability checks for new volatilities, or Monte Carlo methods where changing the underlying stochastic process requires generating an entirely new set of simulated paths, the projected generator engine treats the stochastic model simply as a data input (the matrix M).

3.5 Exact Time Propagation via the Projected Semigroup

In §3.3, we reduced the pricing problem to a system of linear ordinary differential equations: $\dot{\mathbf{c}}(t) = A\mathbf{c}(t)$. The exact analytical solution to this system over any time step Δt is given by the matrix exponential:

$$\mathbf{c}(t + \Delta t) = \exp(A\Delta t)\mathbf{c}(t).$$

Physically, if A represents the instantaneous local law of motion, the matrix exponential $\exp(A\Delta t)$ represents the accumulated motion over the macroscopic time interval Δt. This allows the engine to take arbitrarily large time steps without the time-discretization errors inherent in finite difference methods (like Euler or Crank-Nicolson schemes).

This is the finite-dimensional analogue of the Markov semigroup $P_t = \exp(t\mathcal{L})$. In probability theory, the Markov semigroup is the continuous operator that takes today's payoff and evolves it backward in time to give today's price. Because we projected the continuous generator $\mathcal{L}$ into the finite matrix M, the continuous semigroup $\exp(t\mathcal{L})$ naturally projects into the matrix exponential $\exp(A\Delta t)$.

It is important to clarify the nature of this representation. The matrix M is computed by taking exact inner products, and the matrix exponential $\exp(A\Delta t)$ is computed exactly (up to machine precision). Therefore, the time propagation step itself introduces zero approximation error. The only approximation in the entire method occurs at the spatial level: restricting the infinite-dimensional function space to a finite basis span, denoted $\mathcal{H}_N$. The choice of N (the number of basis functions) determines the resolution of this spatial grid. A larger N allows the basis to capture sharper curves and steeper gradients in the payoff function, at the cost of computing a larger matrix exponential. Once that spatial projection is made, the subsequent temporal evolution is the exact finite representation of the process semigroup on that chosen subspace.

4. Handling Path Dependence: Finite Event-State Recursion

Once the generator has a finite representation, its matrix exponential gives the projected semigroup; the full path-dependent primitive appears when finite event states evolve around it.

A path-dependent contract is one whose payoff depends on the *trajectory* of the underlying, not only on its current level — a barrier that may have been breached, a coupon that may have accrued, an option that may already have been exercised. The projected generator handles this without ever storing a simulated path. It augments the continuous market state with a *finite* discrete state s_m that records exactly the contract's memory (alive or exercised, knocked-in or not, hit count, coupon balance), and evolves the joint system by the same projected semigroup of §3 wrapped in a finite transition automaton. Path dependence becomes finite state, not stored paths.

4.1 The Smallest Nontrivial Instance: American and Bermudan Exercise

The smallest nontrivial event-state system is early exercise, where the only memory required is the binary state: alive or exercised.

For a Bermudan option with exercise dates $t_0 < t_1 < \cdots < t_m$, backward induction alternates between continuation propagation and exercise comparison. Propagate the continuation value:

$$\widetilde{V}(\cdot, t_i) = e^{-r\Delta t} \exp(M\Delta t) V(\cdot, t_{i+1}).$$

Apply the exercise operator:

$$V(x, t_i) = \max(g(x), \widetilde{V}(x, t_i)).$$

Re-project the post-exercise function into the basis:

$$c_j(t_i) = \langle \phi_j, V(\cdot, t_i) \rangle.$$

This is the key distinction from Longstaff-Schwartz. The continuation value propagates by the generator matrix — not estimated from simulated paths. Operator evolution replaces regression entirely.

4.2 Generalization to Multiple States

Many modern exotic derivatives have path-dependent payoffs that require tracking more than just "alive or dead". A Target Accrual Redemption Note (TARN) terminates only when accumulated coupons hit a specific target. A Phoenix Autocallable may have a memory feature where missed coupons are stored and paid later if the underlying recovers. A ratchet option's strike moves across a grid of discrete levels depending on past highs.

To price these without simulating paths, we replace the binary American state with a generalized finite state variable:

$$s_m \in \mathcal{S} = \{1, \dots, n\},$$

representing the discrete "memory" levels needed by the contract. The contract's term sheet is then encoded into an uncontrolled transition matrix $T_m(i, j)$ that evolves the state mass vector π_m:

$$\pi_{m+1}(j) = \sum_i \pi_m(i) T_m(i, j), \qquad T_m(i, j) \geq 0, \qquad \sum_j T_m(i, j) = 1.$$

The value recursion is then

$$V_m(i, x) = \sum_j T_m(i, j; x) \left[C_m(i, j, x) + e^{-r\Delta t} (P_{\Delta t} V_{m+1})(j, x) \right],$$

where $P_{\Delta t} = \exp(\Delta t \mathcal{L})$ is represented in the projected generator basis. The transition graph may contain absorbing states, reset states, or cycles. Finite horizon makes cycles harmless: each observation date applies one transition layer.

Two recursions thus run over the same transition graph in opposite directions. The *forward* mass π_m propagates probabilities and is the object of interest when the quantity wanted is a distribution (as in §4.3). The *backward* value $V_m(i, x)$ is conditional on the current state i — which is precisely why π_m does not appear in it. The two meet only at the end, when the price is read off at the known initial state π_0 (a point mass when the starting state is certain).

In each of these structured products, the continuous market state is propagated by the same projected semigroup; the finite event state records the contract memory. The graph-state projected generator capstone (Part 16, T119) proves that the deterministic error certificate holds for any such finite graph-state system: aggregate value is nonneg, bounded by b_N, and strictly shrinks per added mode — regardless of how many states or which transitions the product requires.

4.3 Touch Options and First-Hit Mass Accounting

A classic application of forward probability tracking is pricing a One-Touch option — a contract that pays a fixed amount the moment the underlying asset breaches a specific barrier level. To price this, we need to know the *forward* distribution of the first hitting time. Instead of simulating thousands of paths to see which ones hit the barrier and when, we track the probability mass directly.

This process uses the simplest possible instance of the §4.2 machinery: a two-state automaton $\mathcal{S} = \{\text{alive}, \text{absorbed}\}$. The state-mass vector π_m collapses into a single number: the total probability mass a_m that is still "alive" at observation date m.

If a_{m-1} is the mass alive before step m, and h_m is the conditional probability of hitting the barrier during this step, the mass splits cleanly:

$$q_m = a_{m-1}h_m, \qquad a_m = a_{m-1}(1 - h_m).$$

The absorbed mass q_m gives the probability that the first hit occurs exactly at step m, while the remaining mass a_m survives to the next step (conserving total probability: $q_m + a_m = a_{m-1}$).

But where does the conditional hit probability h_m come from? The simple equations above are merely mass bookkeeping; the projected generator does the real work to compute h_m.

Instead of tracking a backward value function, we track the *forward* conditional density of the underlying asset, $\rho(x)$. In the projected framework, this density is represented in our standard basis: $\rho(x) = \sum_k d_k \phi_k(x)$.

To move forward one observation step, we propagate this density using the projected semigroup from §3.5, with an absorbing boundary placed on the barrier set. This means applying the matrix exponential $\exp(M\Delta t)$ restricted to the survival region. The mass lost to absorption during this step is exactly the hit probability, computed as the ratio of mass before and after the step:

$$h_m = \frac{\langle \mathbf{1}, \rho \rangle - \langle \mathbf{1}, \exp(M\Delta t)\rho \rangle}{\langle \mathbf{1}, \rho \rangle}.$$

Here, $\langle \cdot, \cdot \rangle$ is the same L^2 inner product—using the exact same Gram matrix G and quadrature—introduced in §3.3.

In short, the projected generator natively computes both the state propagation (via M and $\exp(M\Delta t)$) and the hit probability (via Gram inner products). The macroscopic event-state recursion merely routes this probability mass between the "alive" and "absorbed" states. The engine does not draw paths and then ask which path hit first; it evolves the probability distribution directly. It propagates probability mass through the event automaton, and the mass split at each observation date records the distribution of first hitting time. This is the finite-state replacement for simulation in path-dependent pricing.

4.4 Controlling Physical Assets

Some applications add control: the holder does not merely observe the state but acts on it, and the action moves the contract state (and possibly the market dynamics). The only change from §4.2 is that the transition matrix and the propagator become *action-indexed* — $T_m \to T_m^a$ and $P_{\Delta t} \to P_{\Delta t}^a$ — and the recursion takes a maximum over an admissible action set $A(i)$ attached to each finite state (a Bellman step):

$$V_m(i, x) = \max_{a \in A(i)} \sum_j T_m^a(i, j; x) \left[C_m(i, a, j, x) + e^{-r\Delta t} (P_{\Delta t}^a V_{m+1})(j, x) \right],$$

where $P^a_{\Delta t} = \exp(\Delta t \mathcal{L}^a)$ if the action changes the market dynamics, and $C_m(i, a, j, x)$ is the immediate cash flow of taking action a.

The canonical example is gas storage (Boogert and de Jong, 2008): the operator injects gas when prices are low and withdraws it when high, subject to capacity and rate limits. The continuous market state x is the (log) gas price, propagated by the projected generator; the finite state is the discretized inventory level $v \in \{v_0, \dots, v_n\}$; and each action (inject / hold / withdraw) shifts the inventory by one rung, with admissibility forbidding overfill at v_n and withdrawal at v_0. Computationally nothing new is required: one continuation vector per inventory level, each propagated by the shared $\exp(M\Delta t)$, with the spot cash flow added and a pointwise max selecting the optimal policy.

We keep the treatment at this structural level deliberately. The full physical-asset development — rate and cost calibration, the optimal inject/withdraw policy, and numerical benchmarks, together with the risk-averse, partially observable, and fleet extensions — is the subject of a companion paper, not this one. The point here is only that physical-asset control is the *same* primitive: the market factor still moves by the projected semigroup, now wrapped in a finite Bellman layer.

4.5 One Primitive, Many Payoffs

The contracts above have different state diagrams, but they share one engine. Each contributes only a finite, combinatorial layer — which states exist, how the transition matrix routes mass, and where the max sits. The priced core beneath that layer never changes: the market factor always moves by the same projected semigroup $\exp(\Delta t A)$ on the same basis.

Structure	Contract-specific finite layer	Projected-generator step
European	none	one semigroup propagation
American/Bermudan	exercise comparison	propagation, then max with exercise
Barrier / first-hit	alive / knocked-out state	propagation plus mass split at the boundary
Autocallable	coupon / call / survival states	propagation routed through an event automaton
Controlled asset	operational state and admissible actions	propagation inside a Bellman maximum

This separation is the practical payoff. A new product is finite bookkeeping laid on top of a propagation kernel that is written, calibrated, and error-controlled *once*: payoff engineering changes the transition logic, model engineering changes A, and the time step stays $\exp(\Delta t A)$ on the projected basis. Crucially, the deterministic error certificate of §6 carries over unchanged — the graph-state capstone (Part 16, T119) proves it holds for *every* finite diagram in the table, not just the ones worked out here.

4.6 Algorithmic Form

With the event-state layer in place, the construction can now be written as an executable backward-induction algorithm.

The full method reduces to a clear sequence of operations. Notice how continuous functions and operators are converted into finite vectors and matrices in the setup phase, allowing the backward induction loop to run entirely as fast linear algebra.

Inputs:

- $\mathcal{L}$: Infinitesimal generator (Type: *Symbolic Differential Operator*)
- $g(S)$: Terminal payoff (Type: *Function*)
- $h(S,t)$: Early exercise value (Type: *Function*)
- $\{\phi_k\}_{k=0}^{N-1}$: Basis functions on domain $[a,b]$ (Type: *Set of Functions*)
- $T, \Delta t, r, S_0$: Maturity, time step, rate, initial state (Type: *Numbers*)

Phase 1: Pre-computation (Setup)

1. **Gram matrix** G (Type: $N \times N$ *Matrix*): $G_{jk} = \int_a^b \phi_j(S)\phi_k(S)dS$
2. **Generator matrix** M (Type: $N \times N$ *Matrix*): $M_{jk} = \int_a^b \phi_j(S)\mathcal{L}\phi_k(S)dS$
3. **Propagation matrix** P (Type: $N \times N$ *Matrix*): $P = \exp((G^{-1}M - rI)\Delta t)$

Phase 2: Initialization (Maturity $t = T$)

4. **Project Terminal Payoff** (Type: $N \times 1$ *Vector*): At maturity, the option's value is exactly its payoff function $g(S)$ (e.g., $\max(K - S, 0)$ for a put). We project this continuous function into our finite basis to get the starting coefficient vector $\mathbf{c}_T$. Just like computing M, this is done via numerical quadrature on the grid: we evaluate the payoff at the quadrature points $g(S_q)$, compute the weighted sum $b_j = \sum_{q=1}^{Q} w_q \phi_j(S_q) g(S_q)$ to form the vector $\mathbf{b}$, and then solve for the coefficients: $\mathbf{c}_T = G^{-1}\mathbf{b}$.

Phase 3: Backward Induction Loop (for $t = T - \Delta t$ down to 0)

Here is where time dependence enters. The vector $\mathbf{c}$ changes at each time step. We start with $\mathbf{c}_T$ and work backward to $\mathbf{c}_0$.

For a simple European option, there is no early exercise, so we can jump directly from T to 0 in a single step: $\mathbf{c}_0 = \exp((G^{-1}M - rI)T)\mathbf{c}_T$. For path-dependent options, we must step through intermediate dates.

5. **Propagate Coefficients** (Type: $N \times 1$ *Vector*): Step backward in time by multiplying the current coefficients by the time-machine matrix P: $\tilde{\mathbf{c}}_t = P\mathbf{c}_{t+\Delta t}$. This new vector $\tilde{\mathbf{c}}_t$ represents the *continuation value* of the option at time t.
6. **Reconstruct Continuation Function** (Type: *Function evaluated on quadrature grid*): To apply non-linear rules like early exercise, we must briefly return from coefficient space to function space. We evaluate the continuation function on the quadrature grid: $\widetilde{V}_t(S_q) = \sum_{k=0}^{N-1} \tilde{c}_{t,k}\phi_k(S_q)$.
7. **Apply Early Exercise Max** (Type: *Function evaluated on quadrature grid*): Apply the exercise rule point-by-point on the grid: $V_t(S_q) = \max(h(S_q, t),\ \widetilde{V}_t(S_q))$.
8. **Re-project into Basis** (Type: $N \times 1$ *Vector*): Convert the modified function back into coefficients using the same numerical quadrature matrix multiplication as before: $\mathbf{c}_t = G^{-1}\Phi \cdot \mathrm{diag}(\mathbf{w}) \cdot \mathbf{V}_t$.

Phase 4: Output

9. **Evaluate at Initial State** (Type: *Number*): Price $= \sum_{k=0}^{N-1} c_{0,k}\phi_k(S_0)$

This pseudo-algorithm is the bridge between the mathematical formulation and the software engine: the exact same sequence appears in the Python benchmarks and in the Rust production code.

Continuous vs. Discrete Time Interventions. The backward induction loop (Phase 3) handles path-dependence by stepping backward through discrete time intervals Δt. This naturally prices Bermudan options (where exercise is only allowed on specific dates). But what if we need to price a true American option with continuous exercise rights?

The projected generator method handles this by treating the American option as a high-frequency Bermudan option (e.g., daily exercise, $\Delta t = 1/252$). Because the time propagation step $\exp(A\Delta t)$ is exact, taking many small steps does not accumulate time-discretization errors the way finite difference methods do (there is no CFL stability limit). Alternatively, one could embed the early exercise constraint directly into the differential equation as a non-linear penalty term ($\partial_t V + \mathcal{L}V - rV + \rho \max(h - V, 0) = 0$), which projects into a non-linear ODE for the coefficients ($\dot{\mathbf{c}} = A\mathbf{c} + \text{Penalty}(\mathbf{c})$). However, this destroys the ability to use the matrix exponential, forcing the use of standard ODE solvers. In practice, the high-frequency Bermudan approximation via the exact matrix exponential is both faster and simpler to implement.

4.7 Sensitivities: The Greeks

Once the method produces a value function for path-dependent contracts, the next natural question is the sensitivities of that value.

Pricing gives the value function. Risk management asks how that value changes when the underlying, time, volatility, or rates move. Once the projected generator has produced the finite representation

$$V_N(x,t) = \sum_{k=0}^{N-1} c_k(t)\phi_k(x),$$

the Greeks — the sensitivities used for hedging — are available analytically, with no extra propagation and no sampling noise. They split into two families.

State-space Greeks are pure spatial derivatives, so they follow from differentiating the fixed basis directly (the coefficients $c_k(t)$ do not depend on x):

$$\text{Delta}(x,t) = \sum_{k=0}^{N-1} c_k(t)\,\phi_k'(x), \qquad \text{Gamma}(x,t) = \sum_{k=0}^{N-1} c_k(t)\,\phi_k''(x).$$

The time decay comes straight from the coefficient ODE $\dot{\mathbf{c}} = A\mathbf{c}$:

$$\text{Theta}(x,t) = \sum_{k=0}^{N-1} \dot{c}_k(t)\,\phi_k(x) = \sum_{k=0}^{N-1} (A\mathbf{c}(t))_k\,\phi_k(x).$$

Any higher spatial order costs nothing extra — it is one more derivative of the same basis.

Parameter Greeks — Vega and Rho — are not basis derivatives; they come from differentiating the propagator $e^{A\tau}$ with respect to the parameter that enters $A = G^{-1}M - rI$:

$$\partial_\theta e^{A\tau} = \int_0^\tau e^{As}\,(\partial_\theta A)\,e^{A(\tau-s)}\,ds.$$

For Vega, $\partial_\sigma A = G^{-1}\,\partial_\sigma M$, the volatility entering M through the $\frac{1}{2}\sigma^2$ diffusion term. For Rho, $\partial_r A = G^{-1}\,\partial_r M - I$, whose explicit discount part $-I$ contributes a clean $-\tau e^{A\tau}$ term. Cross- and second-order Greeks (Vanna, Volga) combine the same two mechanisms — one basis differentiation, one semigroup-parameter differentiation. Every one is deterministic and obtained without re-pricing the book under bumped inputs.

By contrast, a Monte Carlo engine typically computes the Greeks by "bump-and-reprice" — shifting an input and rerunning the entire simulation — which amplifies sampling noise. Here the analytic transparency follows directly from separating the temporal dynamics (the matrix exponential) from the spatial geometry (the basis functions). The full error analysis of these operator Greeks, and their cost advantage over bump-and-reprice, is developed in §5.5.

5. Accuracy: Exactness, Approximation, and Certificates

After valuation, algorithmic execution, and sensitivities are finite operator computations, the paper can state what is exact, what is approximation debt, and how residuals certify the output.

Sections 3–4 build the pricing primitive of the paper. The individual ingredients are classical – infinitesimal generators, Galerkin projection, semigroups, finite-state recursion, and differentiation of finite expansions – but their combination is the contribution: a derivative-pricing engine in which value propagation, path memory, exercise logic, and sensitivities are all expressed as finite operator computations.

This section is the trust layer of the paper. Sections 3–4 explain how the engine computes; this section asks why the finite computation is faithful to the infinite-dimensional pricing problem. The central distinction is exact closure versus residual debt. If the projected space is closed under the generator, the finite operator is exact on that space. If it is not closed, the escaped operator component is a deterministic residual that can be measured and carried as an error certificate. The construction sections are therefore mostly explanatory; the formal accuracy claims begin here, with full statements and proofs collected in Appendix C.

5.1 The Boundary of Exactness: Invariant Spaces and Residual Debt

The entire accuracy layer revolves around one central question: if we truncate the infinite-dimensional pricing problem to a finite basis, how do we know the finite engine remains faithful to the original problem?

The method has a sharp dichotomy. Let $\mathcal{H}_N = \mathrm{span}\{\phi_0, \ldots, \phi_{N-1}\}$. If the finite subspace is invariant under the generator, meaning

$$\mathcal{L}\mathcal{H}_N \subseteq \mathcal{H}_N,$$

the generator stays completely within this finite space. The projected generator is therefore not merely a numerical surrogate—it is the *exact* finite representation of the true generator on that space. The matrix exponential $\exp(tM)$ forms the exact semigroup.

If the subspace is not invariant, the generator "spills" information outside the chosen basis. Let Π_N denote projection onto $\mathcal{H}_N$. This spillage is the residual debt:

$$(I - \Pi_N)\mathcal{L}\Pi_N \neq 0.$$

This residual is not statistical noise, but a deterministic error that the engine can compute. In other words, the method does not merely return a price; it identifies exactly which part of the infinite operator has escaped the finite representation.

The invariance criterion. For polynomial diffusions, exact closure is a decidable property of the model-basis pair. Write the drift as a polynomial of degree p and the squared diffusion coefficient as a polynomial of degree q. Applying the generator raises the degree of every monomial by the *fixed* amount $d_{\mathcal{L}} := \max(p-1,\, q-2)$. A finite polynomial basis is therefore generator-invariant if and only if

$$d_{\mathcal{L}} \leq 0, \qquad \text{i.e. drift degree} \leq 1 \text{ and squared-diffusion degree} \leq 2.$$

The drift may therefore be at most linear and the variance at most quadratic. If they exceed these bounds, the generator strictly inflates the polynomial degree, spilling mass outside any finite boundary.

Crucially, this condition does not depend on N. It is not the case that "for large enough N it will become exact". Either every $\mathcal{H}_N$ is invariant or none is. There is no large-N escape from structural non-closure. The formal statement (Criterion 1) and its proof are in Appendix C.1. The common models all satisfy $d_{\mathcal{L}} \leq 0$, hence possess structural closure at every N. In the CEV row, β denotes the elasticity exponent:

Model	$\mu(x)$	$\sigma^2(x)$	(p, q)	$d_{\mathcal{L}} = \max(p-1, q-2)$	Invariant?
Arithmetic BM	μ (const)	σ^2 (const)	$(0, 0)$	-1	Yes (all N)
Ornstein-Uhlenbeck	$\kappa(\theta - x)$	σ^2	$(1, 0)$	0	Yes (all N)
CIR / square-root	$\kappa(\theta - x)$	$\sigma^2 x$	$(1, 1)$	0	Yes (all N)
GBM (log-space)	$(r - \sigma^2/2)$	σ^2	$(0, 0)$	-1	Yes (all N)
Jacobi diffusion	$\kappa(\theta - x)$	$\sigma^2 x(1 - x)$	$(1, 2)$	0	Yes (all N)
CEV	rx	$\sigma^2 x^{2\beta}$	$(1, 2\beta)$	$2\beta - 2$	Iff $\beta \leq 1$
Heston (2D, vol factor)	$\kappa(\theta - v)$	$\xi^2 v$	$(1, 1)$	0	Yes (all N)

(The borderline cases — CEV with $\beta > 1$, which admits no invariant polynomial subspace at any N, and GBM in price space, whose monomials are eigenfunctions but whose moment problem is ill-conditioned — are discussed with the formal statement in Appendix C.1.)

Consequence (Decidability). Given a model specification, the invariance question is decidable in $O(1)$ by inspecting the polynomial degrees. This gives the practitioner a compile-time diagnostic: before running the engine, check whether the model admits exact closure. If yes, the residual can reach machine precision. If no, the characterization provides the minimum N at which closure residual begins to decay.

5.2 The Geometry of Fast Convergence: Why Spectral Methods Dominate

Exact closure is the best case. When closure is not exact, the next question is how the approximation improves as the basis grows. The answer has two parts: the payoff approximation can improve geometrically, but the generator residual must be zero or separately controlled for the total error to inherit that rate.

Spectral convergence (Theorem 1). Under standard regularity conditions (analytic payoff, spectral basis, vanishing closure residual, and discount factor $\beta < 1$), the total error of M-step backward induction splits into two components:

- **Payoff truncation:** Decays *geometrically* at rate ρ^{-N}, where $\rho > 1$ is the analyticity parameter.
- **Closure residual:** Bounded by $\delta(N)/(1-\beta)$, where $\delta(N)$ is the per-step operator residual.

When the subspace is invariant, $\delta(N) = 0$ and only the geometric term remains. The precise assumptions (A1)–(A4), the explicit error bound, and the proof are in Appendix C.2.

Consequence (European pricing). For European options under GBM, Heston, or any affine model with analytic characteristic function, the pricing error from an N-term spectral expansion is $O(e^{-cN})$ where $c = \log \rho > 0$ depends on the payoff analyticity strip width. When the finite subspace is generator-invariant, (A3) gives $\delta(N) = 0$ and the second term vanishes.

This provides a decisive advantage over Monte Carlo ($O(P^{-1/2})$, algebraic) and finite differences ($O(h^2)$ or $O(h^4)$, polynomial in grid spacing). The projected generator achieves **spectral convergence**: doubling N squares the number of correct digits.

The proof architecture deliberately separates external approximation theory from finite operator algebra. Assumptions (A1)–(A2) supply the analytic regularity and basis quality — these are mathematical hypotheses drawn from classical approximation theory. Once such a tail bound is supplied, the mechanized proof suite verifies that the projected-generator recursion preserves the advertised exponential rate through the finite pricing architecture:

$$\varepsilon_{\text{total}}(M, N) \leq M_V \, C \rho^{-N}.$$

Assumption (A3) is the only hypothesis that depends on the specific model-basis pair; it is computable (§5.6) and vanishes identically for invariant subspaces (§5.1). One caveat on the headline rate: the exponential factor ρ^{-N} is carried by the payoff-truncation term. The closure-residual term $\delta(N)/(1-\beta)$ has no rate of its own in Theorem 1 — it is only assumed to vanish. A *global* exponential rate therefore holds exactly when $\delta(N) = 0$ (invariant subspaces — polynomial diffusions in a polynomial basis, OU/affine in a spectral basis) or when an independent, model-specific argument

supplies a decay rate for $\delta(N)$ that is no slower than ρ^{-N}. Outside those cases the operator residual can dominate, and "exponential convergence" describes the payoff term, not the total error.

5.3 Paths vs. Operators: Two Representations of Financial Dynamics

Before comparing computational bounds, it helps to understand the fundamental difference in what is being computed. By identifying the two approaches as dual representations of the same mathematical object, we see the structural divide: Monte Carlo samples realized futures of the process from the outside, while the projected generator represents the infinitesimal operator that generates those futures from the inside.

Operator-sampling duality (Proposition 1). *Let $\mathcal{L}$ be the generator of a diffusion (X_t) and $P_t = e^{t\mathcal{L}}$ its semigroup. Then the derivative price $V(x,t) = P_{T-t}g(x)$ admits two representations:*

- *(i) (Sampling representation.)* By the Feynman-Kac theorem, the value is an expectation over sample paths of X:

$$V(x,t) = \mathbb{E}[e^{-r(T-t)}g(X_T) \mid X_t = x]$$

  Monte Carlo estimates this expectation by drawing P i.i.d. paths and averaging.

- *(ii) (Operator representation.)* $V(x,t) = \sum_{k=0}^{N-1} c_k(t)\phi_k(x)$, where $\mathbf{c}(t) = e^{A(T-t)}\mathbf{g}$ and A is the projected generator matrix. The projected generator computes this by matrix exponential.

Both representations converge to the same limit V_{true}. The error structure, however, is fundamentally different:

Property	Sampling (i)	Operator (ii)
Error source	Finite sample	Finite basis
Error lives in	Sample space Ω	Function space L^2
Error is	Random ($\hat{V}_P - V$ depends on ω)	Deterministic ($V_N - V$ depends on N)
Residual	Not computable (requires V_{true})	Computable: $\|(I - \Pi_N)\mathcal{L}\Pi_N\|$
A posteriori residual	Statistical only	Computable from the operator

The asymmetry is structural: in representation (i), the residual $V_{true} - \hat{V}_P$ is a random variable whose distribution depends on the unknown V_{true}. In representation (ii), the residual $(I - \Pi_N)\mathcal{L}\Pi_N$ is a deterministic operator computable from the known A_N and the extended basis — it does not require knowing V_{true}.

Proof in Appendix C.5.

Conceptual consequence. Monte Carlo integrates *over paths*: it sees the process from the outside, through a stochastic keyhole. The projected generator represents *the operator*: it sees the process from the inside, through its algebraic structure. This is the methodological core of the paper. Pricing, exercise, and Greeks become finite-dimensional operator computations rather than statistical estimates.

This duality generalizes beyond finance. Any problem where a linear operator (Laplacian, Hamiltonian, Markov generator) drives the dynamics admits both a sampling and an operator representation. In quantum mechanics, the operator representation (Schrödinger equation) enables exact computation of observables; the sampling representation (path integral) is used when the operator is too complex for exact diagonalization. In computational finance, the same trade-off applies—and the projected generator is the analogue of exact diagonalization.

5.4 The Math of the Advantage: Why Monte Carlo is Structurally Slower

Because of the structural duality established above, the convergence result of §5.2 becomes a rigid computational separation. Monte Carlo buys accuracy by sampling more independent paths; the projected generator buys accuracy by adding more coordinates to the finite representation of the operator. For smooth low-dimensional problems, these are different asymptotic economies, not merely different implementations.

While Theorem 1 established that the projected generator achieves $O(\rho^{-N})$ error with N basis functions for analytic payoffs, we must ask whether traditional sampling methods could ever match this performance. Theorem 2 shows that this exponential rate is intrinsically unreachable by any sampling-based method.

Complexity separation (Theorem 2). On the class of analytic-payoff European problems, the performance gap is structural:

- **Projected Generator:** Reaches ε-accuracy with $N = O(\log(1/\varepsilon))$ basis functions.
- **Sampling estimators:** Any estimator built from P independent terminal-value samples needs $P = \Omega(\varepsilon^{-2})$.

This is a minimax lower bound (Cramér–Rao / Le Cam) that no sampling scheme can evade, because i.i.d. terminal-value access is forced by the strong Markov property, not a design choice. The cost ratio therefore tends to zero as $\varepsilon \to 0$: the exponential advantage is structural, not empirical. The formal three-part statement and proof are in Appendix C.3.

Remark (Scope). The separation is sharpest for smooth payoffs in low-to-moderate dimension ($d \leq 5$). For discontinuous payoffs (digitals, barriers), the PG rate degrades to algebraic $O(N^{-k})$ and the gap narrows. For $d \gg 5$, the PG basis dimension grows as N^d (before tensor decomposition), and Monte Carlo's dimension-free $O(P^{-1/2})$ rate becomes competitive. The separation theorem identifies the precise regime — analytic payoffs, tractable generators, moderate dimension — where the projected generator is provably superior.

5.5 Zero Statistical Noise: Operator Greeks and Exact Derivatives

For *Greeks* — the price sensitivities that drive hedging and risk management — the operator representation has an immediate practical advantage. Once the value is represented as a finite expansion, state Greeks are derivatives of the basis and parameter Greeks are derivatives of the finite semigroup. Sampling methods typically estimate the same quantities by bump-and-reprice, where sampling noise is amplified by the finite-difference denominator.

Operator Greek dominance (Proposition 2). *Let θ be a smooth model parameter (spot S, rate r, volatility σ, or time t). Denote the true Greek $G = \partial V/\partial\theta$. Then:*

(i) (Operator Greek.) The projected-generator Greek G^N is computed via the chain rule on the

finite expansion:

$$G^N = \mathbf{c}^T \frac{\partial \phi}{\partial \theta}(x) + \left(\frac{\partial \mathbf{c}}{\partial \theta}\right)^T \phi(x)$$

where $\partial \mathbf{c}/\partial \theta$ satisfies the differentiated semigroup equation (to leading order):

$$\frac{\partial \mathbf{c}}{\partial \theta} = e^{A\tau}\left(\frac{\partial A}{\partial \theta}\right)\tau\,\mathbf{g} + e^{A\tau}\frac{\partial \mathbf{g}}{\partial \theta}$$

The finite-basis Greek error is controlled by the differentiated residual:

$$|G - G^N| \le \underbrace{\left\|\frac{\partial}{\partial \theta}(I - \Pi_N)\mathcal{L}\Pi_N\right\|}_{\text{closure-Greek residual}} \cdot T \cdot e^{T\|A_N\|} + O(\varepsilon^2_{\text{closure}}).$$

This bound is computable from the same operator matrices used for pricing.

(ii) (Sampling Greek.) The bump-and-reprice Monte Carlo Greek $\hat{G}_P = (\hat{V}_P(\theta+\delta) - \hat{V}_P(\theta-\delta))/(2\delta)$ suffers from noise amplification:

$$\mathrm{MSE}(\hat{G}_P) = \underbrace{O(\delta^4)}_{\text{discretization bias}^2} + \underbrace{O\Big(\frac{1}{P\delta^2}\Big)}_{\text{noise amplification}}.$$

Optimizing over δ: $\delta^* = O(P^{-1/6})$, minimum RMSE $= O(P^{-1/3})$.

(iii) (Dominance.) For any target Greek accuracy ε_G:

	Operator Greek (PG)	Sampling Greek (MC)
Cost for accuracy ε_G	$N = O(\log(1/\varepsilon_G))$ basis functions	$P = O(\varepsilon_G^{-3})$ paths
Residual control	Deterministic operator residual	Statistical only
Higher-order Greeks (Γ, Vanna, Volga)	Same cost (differentiate basis)	Each order doubles the bump grid
Cross-Greeks ($\partial^2 V/\partial S \partial \sigma$)	Same cost (mixed partial of basis)	Requires 4 pricing runs per cross

The cost ratio for operator Greeks is $O(\log(1/\varepsilon_G)/\varepsilon_G^{-3})$, which is superpolynomially better than for prices alone ($O(\log(1/\varepsilon)/\varepsilon^{-2})$).

Proof in Appendix C.6.

Practical consequence. A structured products desk computing Delta, Gamma, Vega, Rho, and Theta for a single autocallable needs 10 MC pricing runs (2 bumps × 5 Greeks) plus the base run — 11 runs total, each with independent noise. The PG engine computes all five Greeks from a single basis expansion, deterministically. For a portfolio of 500 positions with 5 Greeks each, MC requires 5,500 noisy pricing runs; PG requires 500 deterministic runs.

5.6 Replacing Confidence Intervals with Deterministic Error Certificates

Having established the asymptotic convergence rates and exact operator derivatives, we now turn to the practical problem of bounding the error for a specific computation. This is where the method becomes auditable: the output is not just a number, but a value together with a deterministic certificate for the finite-operator residual. Theorem 1 provides an asymptotic rate. For a deployed engine, we also need a computable residual estimate at the specific basis size N used. This section describes that residual accounting. The companion certificate paper develops the audit semantics of turning these residuals into formal price intervals.

Computable residual interval (Theorem 3). The engine returns a price together with a *deterministic* error bar $\varepsilon_{\text{total}}(N)$ that splits into four computable components:

1. **Generator-closure:** A singular value of the residual matrix.
2. **Payoff-projection:** A spectral coefficient tail.
3. **Multi-step propagation:** A Bellman-contraction sum.
4. **Event/bridge correction:** An event-operator norm.

Because each piece is read directly from engine-internal quantities, the interval requires no reference price. If all four components vanish, the computed price is exact. The formal statement and proof are in Appendix C.4.

The engine outputs:

$$\boxed{\text{price} = V_{\text{calc}}^{N} \pm \varepsilon_{\text{total}}(N)}$$

Corollary 1 (Deterministic Residual vs. Monte Carlo Standard Error). *Let σ_{payoff} denote the payoff standard deviation under the simulation measure. A Monte Carlo estimator with P independent paths has standard error $\sigma_{payoff}/\sqrt{P}$. The projected generator reports a smaller residual than the Monte Carlo standard-error band whenever*

$$\varepsilon_{\text{total}}(N) < \frac{\sigma_{\text{payoff}}}{\sqrt{P}}.$$

Since $\varepsilon_{total}(N) = O(\rho^{-N})$ decays exponentially while $\sigma_{payoff}/\sqrt{P}$ decays algebraically in the path budget, there exists a crossover basis size N^* beyond which the projected-generator residual is smaller than the Monte Carlo confidence band for any fixed path budget P.

For a reviewer, the distinction is operational:

Quantity	Monte Carlo / LSMC	Projected generator
Reported uncertainty	Sampling standard error; shrinks as $P^{-1/2}$	Deterministic residual gap
Repeatability	New seed gives a different estimate	Same inputs give the same price and Greeks
Phoenix benchmark (§6.3)	SE = 158.02 on 200k paths	Residual gap $\varepsilon = 0.7610$
American benchmark (§6.1)	SE = 0.0227 for Longstaff-Schwartz	Fixed truncation error, no path noise

Quantity	Monte Carlo / LSMC	Projected generator
Interpretation	Statistical confidence interval around an estimator	Finite-basis residual around the engine value

5.7 Evading the Dimensionality Curse: Tensor Methods for Multiple Assets

The preceding claims are strongest when the finite operator remains tractable. Tensor decomposition is the practical mechanism that keeps moderate-dimensional factor models inside that regime. For a d-factor model with N basis functions per factor, the naive projected generator matrix is $N^d \times N^d$ — exponential in d. Tensor decomposition avoids this curse.

Kronecker structure. If the d factors are independent (or have separable correlation structure), the generator decomposes as:

$$A = A_1 \otimes I \otimes \cdots \otimes I + I \otimes A_2 \otimes \cdots \otimes I + \cdots + I \otimes \cdots \otimes I \otimes A_d + A_{\text{corr}},$$

where each A_i is $N \times N$ and A_{corr} captures cross-factor coupling. The matrix exponential of a Kronecker sum factors:

$$\exp\left(\sum_i A_i^{(d)}\right) = \exp(A_1) \otimes \exp(A_2) \otimes \cdots \otimes \exp(A_d) \quad (\text{if } A_{\text{corr}} = 0).$$

Cost reduction:

Method	Matrix size	Exponential cost	Per-step cost
Naive	$N^d \times N^d$	$O(N^{3d})$	$O(N^{2d})$
Kronecker	d matrices of $N \times N$	$O(d \cdot N^3)$	$O(d \cdot N^2)$
Kronecker + correction	rank-r correction	$O(d \cdot N^3 + r \cdot N^d)$	$O(d \cdot N^2 + r \cdot N^d)$

For correlated factors ($\rho \neq 0$), the correction term A_{corr} is typically low-rank (rank 1 for each pair of correlated factors, giving rank $\binom{d}{2}$). A rank-r correction adds $O(r \cdot N^d)$ per step but avoids the full $O(N^{3d})$ exponential computation.

Practical scaling:

Factors d	Basis N	Naive dim	Kronecker dim	Speedup
2	32	1024	64	16×
3	16	4096	48	85×
4	12	20736	48	432×
5	8	32768	40	819×

This makes 3-5 factor models (power + gas + carbon + weather + demand) tractable on a single machine.

The exact-generator-closure proof suite checks the zero-residual case and its exact observable consequence. It also checks the Brownian and GBM log-space finite-latent cases, along with the affine Riccati closure chain.

5.8 Incorporating Jumps Without Simulation: The Projected Lévy Operator

The previous sections rely on local diffusion generators. Finite-activity jump models require one additional operator component, but not a different architecture. The nonlocal jump term becomes another projected matrix block, not a Monte Carlo layer.

Let X_t be a one-dimensional jump-diffusion with Brownian volatility σ, drift μ, jump intensity $\lambda < \infty$, and jump size law ν. Its infinitesimal generator on suitable test functions is the nonlocal Lévy operator:

$$\mathcal{L}f(x) = \mu f'(x) + \frac{1}{2}\sigma^2 f''(x) + \lambda \int_{\mathbb{R}} (f(x+y) - f(x))\nu(dy).$$

The projected-generator matrix decomposes linearly:

$$M_N = M_N^{\text{diff}} + M_N^{\text{jump}}, \qquad (M_N^{\text{jump}})_{jk} = \lambda \left\langle \phi_j, \int (\phi_k(\cdot + y) - \phi_k(\cdot))\nu(dy) \right\rangle .$$

Projected Lévy generator extension (Theorem 5). Under the same kind of assumptions as before — an analytic jump law and both operator blocks projectable onto the basis:

- **Matrix Semigroup:** European propagation under the jump-diffusion is again a finite matrix semigroup $\exp(T(M_N - rI))$.
- **Error Decomposition:** The total error splits into diffusion, jump, payoff, and truncation parts.
- **Diagonalization:** In a Fourier/COS basis, the jump component is diagonalized by the Lévy symbol. Each frequency mode u_k is simply multiplied by $\exp(T\psi_J(u_k))$.

The key simplification is that the nonlocal jump term is fundamentally a projected-generator component, not a simulation layer. The precise assumptions (L1)–(L3), the error bound, and the proof are in Appendix C.8.

Scope. This theorem covers finite-activity Poisson and compound-Poisson jump-diffusions with known jump law, including the Merton model used in Appendix A.4. It does not cover arbitrary state-dependent jump intensities, infinite-activity Lévy processes with poor analyticity, or path-dependent jump mechanisms without a computable nonlocal generator. Those require separate projected Lévy-generator analysis.

5.9 The Master Theorem: Exact Path-Dependent Pricing for Polynomial Diffusions

The individual results of the preceding sections — exact operator closure, deterministic sensitivities, tensor decompositions, and jump extensions — can now be combined into a single unifying theorem for path-dependent pricing under polynomial diffusions. This is the paper's primary theoretical contribution: the synthesis that goes beyond any single prior result.

Polynomial invariance is known in the literature for moments and European pricing; the projected-generator architecture lets the same invariance support the full path-dependent finite event-state pipeline.

Path-dependent polynomial diffusion certificate (Theorem 4). For a polynomial diffusion whose generator preserves a finite polynomial space, combined with a finite event-state graph, the total pricing error of a path-dependent contract splits into four parts:

1. **Spatial:** Exactly zero (polynomial invariance).
2. **Temporal:** Exactly zero (time propagation is an exact matrix exponential).
3. **Graph:** Exactly zero (deterministic event-state transitions are exact in the finite state space).
4. **Payoff-truncation:** The only surviving error term, $\|g - \Pi_N g\|$.

Because the model closes perfectly, the truncation error decays geometrically ($C\rho^{-N}$) for analytic payoffs, and the graph-state aggregate inherits the exact same strictly shrinking bound. The formal statement and proof are in Appendix C.7.

Scope of application. This covers any contract expressible as a finite-state machine over a polynomial diffusion:

- American options (2-state: alive/dead);
- Barrier options (3-state: alive/knocked-out/exercised);
- Autocallable structures (K-state: observation dates + knock-in/out);
- Coupon memory contracts (state tracks accumulated coupon eligibility);
- Multi-asset with correlated polynomial factors (via tensor decomposition, §5.7).

Proof in Appendix C.7.

Comparison to prior art. Cuchiero et al. (2012) and Filipović–Larsson (2016) establish polynomial invariance for moment computation and European pricing. Ackerer–Filipović (2020) extends polynomial expansions to European option pricing and discusses exotic payoffs depending on finitely many prices. The distinction here is narrower and structural: those works do not build the finite event-state projected-generator pipeline for American exercise, barriers, autocalls, and coupon memory, nor do they attach the deterministic graph-state error certificate proved here. Theorem 4 shows that the same polynomial invariance that makes moments exact also makes the *full path-dependent pricing pipeline* exact in three of four error components — a strictly stronger result that exploits the projected-generator architecture.

6. Numerical Evidence and the Proof Trust Boundary

With the certificate layer stated, the paper can now show the worked examples, validation surfaces, and trust boundary that support the construction.

Appendix A provides worked arithmetic examples and extended validation surfaces for the projected-generator engine. This section presents the core numerical evidence used to validate the method. The full reproducible data surfaces, Python benchmarks, and the Rust projected-generator engine corpus are maintained in the companion repository.

6.1 Validating the Baseline: American Exercise Benchmarks

We evaluate an American put contract with $S_0 = 100$, $K = 100$, $r = 5\%$, $\sigma = 20\%$, and $T = 1$ year across 50 exercise dates.

Method	Price	Error vs binomial	Delta	Noise	Runtime
Projected generator	6.0624	-0.0272	-0.4112	deterministic	21.5 ms
Finite difference	6.0289	-0.0607	-0.4107	deterministic	6.0 ms
Longstaff-Schwartz	6.0848	-0.0048	n/a	SE 0.0227	512.3 ms
Binomial tree reference	6.0896	+0.0000	n/a	deterministic	4.5 ms

The projected-generator result is deterministic and lands in the same benchmark band as finite-difference and binomial references. Longstaff-Schwartz achieves a smaller absolute error on this specific contract (-0.0048 versus -0.0272). This is expected: Longstaff-Schwartz is an unbiased estimator converging to the true price as path count grows, while the projected generator carries a fixed truncation error from the finite basis.

The critical comparison is determinism versus noise. Longstaff-Schwartz carries a standard error of 0.0227 — repeated runs produce drifting prices. The projected generator returns the same price and Greeks on every call. Finite differences run faster here (6 ms versus 21.5 ms) because the 1D grid is small. Beyond $d = 3$, the $O(N_x^d)$ grid explosion makes finite differences impractical; tensor-decomposed projected generators remain tractable to $d \approx 5$.

Across a five-case American-put surface that varies strike, volatility, maturity, and exercise dates, the projected-generator benchmark achieves a mean absolute error of 0.0249 and a maximum absolute error of 0.0341 versus the 1000-step binomial reference.

6.2 Eliminating Sampling Noise: Deterministic Greeks in Practice

Part 30 proves that risk sensitivities inherit the deterministic price certificate and avoid the noise amplification of finite-difference Greeks computed on Monte Carlo prices. On a Black–Scholes call ($S_0 = K = 100$, $r = 5\%$, $\sigma = 20\%$, $T = 1$), where the exact Gamma is known in closed form, a central-difference Gamma estimate on independent 100k-path Monte Carlo prices has a root-mean-square error that grows from 1.3×10^{-3} at bump $h = 10$ to 3.3×10^{2} at $h = 0.02$ — a five-order-of-magnitude blow-up consistent with the SE/h^2 scaling of Part 30 T235 (greek_stability_results.json). Common random numbers mitigate the variance but require path reuse, which breaks under an American reprice (the exercise boundary shifts) or cross-model repricing. The deterministic Greek, obtained by differentiating the finite expansion, equals the exact value with no bump and no noise term.

6.3 Scaling to Path Dependence: Phoenix Autocallables

Moving from simple early exercise to complex path dependence, we now test the engine on structured products. Consider a 3-asset worst-of Phoenix autocallable: 3-year maturity, semi-annual observations, 8% p.a. memory coupon, step-down barriers, equicorrelation $\rho = 0.6$.

Method	Price	Runtime	Noise / bound	Greeks
Projected generator engine	1,009,178.61	70.9 ms	$\varepsilon = 0.7610$	Delta0 45948.5868, Vega0 -94667.31, Rho -1162832.12
Vectorized Monte Carlo, 200k paths	1,014,083.08	0.13 s	SE 158.02	bump-and-reprice required

The price gap versus the 200k-path Monte Carlo reference is 0.48%. The current engine uses forward first-hit event recursion: at each observation date it updates the alive probability, pays the coupon accrued to that actual hit date, and carries the remaining alive mass forward. This is the path-dependent semantics required for Phoenix autocallables. At rank 1, the method captures 73.3% of the cross-asset variance.

A clarification on what the reported budget does and does not cover is essential here, because the two numbers live on different scales. The deterministic budget $\varepsilon = 0.7610$ bounds only the *in-model projected-dynamics* residual — spectral truncation, quadrature, propagation, and event-bridge error on the finite representation actually evolved. It does **not** bound the rank-1 cross-asset correlation truncation: capturing 73.3% of the cross-asset variance leaves roughly 26.7% unmodeled, and that low-rank projection error — together with any model/measure difference against the Monte Carlo reference — is the dominant component of the 0.48% gap. That gap (4,905 price units, 31 MC standard errors) therefore exceeds ε by orders of magnitude. This is not a contradiction: the certificate is honest precisely because it reports the residual of the dynamics it propagates rather than silently absorbing the correlation-rank error it does not control. Tightening this gap requires a higher correlation rank (§5.7), not a tighter spectral budget. The paper does not claim desk-level mark-to-market accuracy without live market-data validation. The claim is narrower: the engine is deterministic, reports the approximation budget for the dynamics it evolves, and returns Greeks without bump-and-reprice reruns.

The Rust engine corpus validates the projected-generator side across six Phoenix-style term sheets. The fresh corpus run reports an average engine price of 1,006,305.68, a price range from 999,649.13 to 1,014,256.14, average total error budget 0.798745, and average captured variance 72.44%. Each term sheet is identified by a reproducibility hash in engine_validation_report.md.

6.4 The Proof Trust Boundary: Separating Mechanized Claims from Assumptions

After the numerical evidence, the remaining trust question is which claims are machine-checked and which are paper-level or empirical.

The numerical results above are backed by a separate formal trust boundary. The complete proof inventory remains in Appendix D.

Proof trust summary. The following table separates what is verified by the primary proof kernel, what relies on classical mathematics, and what is validated empirically.

Claim	Status	Source
Generator closure algebra (274 named theorems, 747/747 checks)	**Mechanized**	Primary proof kernel; representative Lean stamps with explicit assumptions
Graph-state capstone (T119)	**Mechanized**	Part 16: arbitrary finite graph certificate
N-state inductive capstone (T128)	**Mechanized**	Part 17: certificate for every finite state count
OU eigenbasis exactness (T131)	**Mechanized**	Part 18: zero closure residual for OU in Hermite basis
OU explicit rate constant (T134-T136)	**Mechanized**	Part 18: rate derived for OU, given coefficient decay
Complexity separation vs MC (T141)	**Mechanized**	Part 19: PG O(log 1/) vs MC Ω(1/ ²)
Polynomial diffusion class-exactness (T148)	**Mechanized**	Part 20: OU, CIR, Jacobi — all exact in poly basis
Multidimensional crossover d* (T154)	**Mechanized**	Part 21: PG wins d < d, *MC wins d > d*
Temporal exactness (T161)	**Mechanized**	Part 22: zero time-stepping error on invariant subspace
Regularity hierarchy (T168)	**Mechanized**	Part 23: C^k → algebraic, analytic → geometric rate
Path-dependent polynomial diffusion (T178)	**Mechanized**	Part 24: path-dependent certificate under polynomial diffusions
Method-selection theorem (T188)	**Mechanized**	Part 25: complete PG vs MC/FD decision boundary with negative result
ML bridge rate identity (T204, Parts 26–27)	**Mechanized** (companion paper)	Condition-number identity; developed/validated in the companion ML paper, not claimed here
Rough volatility via Markovian lift (T220)	**Mechanized** + numerics	Part 28: lift is PGM-exact; two-source deterministic certificate
Universal subsumption (T231)	**Mechanized**	Part 29: COS/Carr-Madan/Fourier as PGM basis specializations
Greek certificate inheritance (T242)	**Mechanized** + benchmark	Part 30: deterministic Greeks; bump-reprice diverges as SE/h²
Certified value iteration (T253)	**Mechanized**	Part 31: projected VI within cert/(1−) of truth; RL bridge
Residual gap decomposition (4 components)	**Mechanized**	exact_generator_closure domain

Claim	Status	Source
Zero-gap exactness	**Mechanized**	Criterion 1 + Theorem 3 (Part 11); kernel invariant_subspace_zero_residual
Gap monotonicity in N	**Mechanized**	Theorem 2, kernel-verified
Exponential convergence rate $O(\rho^{-N})$ (general)	Classical assumption	Jackson-Bernstein (Trefethen 2013)
Exponential rate for OU (concrete)	**Mechanized** (modulo coefficient-decay input)	Part 18, T129-T136
Payoff analyticity ($\rho > 1$)	Model assumption	Per-product regularity
Bellman contraction ($\beta < 1$)	Classical result	Standard DP theory (Bertsekas 2012)
MC complexity lower bound $\Omega(P^{-1/2})$	Classical result	Cramér-Rao (Lehmann-Casella 1998)
Invariant subspace characterization	Paper-level proof; degree-preservation direction mechanized — general derivation from polynomial-ring primitives (T266-T276, Part 33), with per-model corollaries (T142, T144, T146)	Criterion 1, §5
Complexity separation (Thm 2)	**Mechanized** (finite algebra) + classical inputs	§5.4, Part 19 (T137-T141); Cramér-Rao + spectral rate isolated
Regime classification	Paper-level taxonomy	§7.4, structural boundaries
Price/Greek accuracy vs MC/LS/FD	**Benchmarked**	§6–7, reproducible code
Ablation study (8 configurations)	**Benchmarked**	§7.3, component-removal experiments
Rough-vol Markovian lift convergence	**Numerical study**	Appendix A.6, H=0.1, 1200× error reduction over 32 factors
Deterministic Greeks vs bump-reprice	**Benchmarked**	§6.2, Black-Scholes Gamma, bump-reprice diverges as SE/h²
Recognized American-put benchmark	**Benchmarked**	recognized_benchmark_results.json, 15k-step binomial reference
Market calibration quality	Out of scope	Not addressed

The primary mechanized proof suite checks the exact-generator-closure component: 274 named theorems, 747/747 checks passed, zero undisclosed axioms in the finite-arithmetic layer, and zero vacuous statements in the current full-suite run. These proofs support the finite operator algebra used by this method paper; broader proof-carrying-price and regulatory audit claims are reserved for the companion certificate paper.

The trust sources are deliberately separated. The primary mechanized suite checks finite real-arithmetic implications: closure, residual algebra, mass conservation, and zero-gap exactness. The

Lean 4 stamps are representative independent replay targets, not a claim that the whole 274-theorem suite is Lean-sealed; their classical assumptions are visible as declarations. The analytic convergence input (for example Jackson-Bernstein spectral decay and payoff regularity) remains a paper-level mathematical assumption unless explicitly listed as checked below.

The mechanized record is organized into 33 parts spanning the full result set — from exact-closure implications (Parts 1–6) and event-state / control invariants (Parts 7–8) through the residual-interval and exponential-convergence capstones (Parts 11–15), the graph-state and N-state certificates (Parts 16–17), concrete spectral rates and class-exactness (Parts 18–24), the method-selection theorem (Part 25), and the verified extensions (Parts 26–33: ML bridge, rough-vol lift, transform subsumption, Greek certificates, certified value iteration, the substantive derivation layer, and the polynomial degree-closure layer that derives invariant-subspace exactness from two polynomial-ring primitives). The complete per-part catalogue, with theorem counts and grand-theorem identifiers, is in **Appendix D.1**.

The audit semantics of these intervals are developed in full in a forthcoming companion paper on deterministic error certificates. This paper presents the projected-generator architecture and validation; that companion treats proof-carrying prices and formal audit records.

Consequently, we deliberately split the paper-level derivation. Starting from the SDE, we derive the generator using Itô's formula. Galerkin projection then turns $\mathcal{L}$ into the matrix M. If the space is invariant ($\mathcal{L}\mathcal{H}_N \subseteq \mathcal{H}_N$), $\exp(tM)$ becomes the exact semigroup representation on $\mathcal{H}_N$. If invariance fails, the residual $(I - \Pi_N)\mathcal{L}\Pi_N$ measures the approximation debt.

This derivation remains explicit in the paper. The proof kernel verifies the mathematical consequences of closure, while the manuscript demonstrates how the SDE, projection, and residual definitions instantiate those consequences for the pricing model. Separate proof domains cover the related American exercise and autocallable applications, which are treated in forthcoming companion papers:

- a forthcoming companion paper on regression-free American option pricing — applies the method to optimal stopping with a five-case benchmark surface against binomial, finite-difference, and Longstaff-Schwartz references;
- a forthcoming companion paper on a projected-generator engine for Phoenix autocallable pricing and risk — applies the method to multi-asset structured products with analytical Greeks, a six-case synthetic validation corpus, and a Rust engine implementation.

Proof Traceability and Lean Export.

Three trust artifacts back the mechanized claims, detailed in **Appendix D**:

- **Proof traceability (Appendix D.2).** The proof-backed component is the exact-generator-closure artifact in the companion mechanized-verification repository (747 verified checks, 274 named theorems, 0 undisclosed axioms in the finite-arithmetic layer). The machine-rendered statement layer is formal_statements.md. The trace is deliberately conservative: the kernel does **not** prove Jackson-Bernstein, payoff analyticity, or model-specific regularity — those remain paper-level assumptions, made explicit as hypotheses.
- **Lean 4 export (Appendix D.3).** stamp/ExactGeneratorClosure.lean gives an independent type-checked replay of a representative subset of the core theorems (Parts 1, 4, 6–12), with all external analytic assumptions visible as explicit hypotheses rather than hidden axioms.

- **Proof contract and completeness (Appendix D.4).** proof_contract_manifest.yaml maps each paper claim to its kernel theorem(s), verification level, and external assumptions; the kernel's scope is the finite-arithmetic layer, and Parts 32–33 derive several previously-assumed extension facts from established structure — Part 33 in particular derives the invariant-subspace (degree-preservation) property from polynomial-ring primitives rather than positing it per model. A second Lean stamp type-checks the finite core of five extension capstones.

7. Boundaries and Benchmarks

Evidence is incomplete without boundary conditions: the same section must say where projected generators work, where they fail, and how that matches known benchmarks.

The preceding sections establish what the projected generator achieves under ideal conditions. Evidence is incomplete without boundary conditions. This section states exactly where the method wins, where it degrades, and where it fails entirely, calibrating these boundaries against external published benchmarks.

Before charting the boundaries, it is worth stating explicitly where the projected generator is not just an alternative to Monte Carlo, but a structural replacement. Because the method yields a deterministic, globally valid pricing function $V(t, x)$ over the entire state space, it naturally eliminates inner simulations. This makes it a uniquely powerful engine for the most computationally brutal problems in quantitative finance: **Counterparty Credit Risk (XVA) and Nested Monte Carlo**. In XVA (CVA, FVA, KVA), generating future exposure profiles typically requires simulating thousands of market paths, and pricing the portfolio at every node on every path. The projected generator replaces the inner pricing simulation with a sub-millisecond matrix-vector product $v_t = e^{A(T-t)}v_T$, collapsing a nested simulation into a fast single-level simulation.

Similarly, for **path-dependent exotics like Phoenix Autocallables**, coupling the generator to an event-state automaton entirely eliminates the need to sample barrier-crossing probabilities or early termination paths. The path dependence is absorbed into the matrix algebra, yielding deterministic prices and noise-free Greeks without simulating a single trajectory.

This same continuous-discrete synthesis extends to **Physical Asset Control and Stochastic Dispatch** (e.g., gas turbines, battery storage, and hydro reservoirs). Managing physical assets requires optimizing over path-dependent operational constraints (ramp limits, minimum up/down times, state-of-charge). Standard grid-based dynamic programming struggles with dimensionality and interpolation errors, while simulation-based regression often produces suboptimal policies with hidden approximation debt. By treating physical operating modes as discrete event states and market dynamics as a projected generator, the method solves the continuous-state Bellman contraction algebraically. This yields a deterministic, globally valid optimal control policy without discretizing the continuous variables, directly delivering certified bounds for asset valuation.

7.1 The Regime of Dominance: Smooth Payoffs and Tractable Operators

When the pricing problem features a smooth payoff, low dimensionality ($d \leq 5$), and a tractable generator, the projected generator structurally dominates sampling methods.

Spectral vs. Algebraic Convergence. For a standard European ATM call, the pricing error shrinks according to two fundamentally different economies:

Computational budget	PG error	MC SE (P paths)	FDM error (N_x grid)
Small (N=8 / P=1k / N_x=50)	1.08	0.52	0.0841
Medium (N=16 / P=10k / N_x=200)	2.27×10^{2}	0.16	0.0053
Large (N=32 / P=100k / N_x=800)	3.28×10	0.052	3.3×10
Very large (N=64 / P=1M / N_x=3200)	2.66×10^{1}	0.016	2.1×10

Each doubling of the basis size N squares the precision, reaching machine epsilon at $N = 64$ in sub-millisecond runtime. Monte Carlo requires a 100× increase in paths to gain a single decimal digit, hitting a structural barrier.

Published Benchmark Calibration. In this optimal regime, the projected generator matches or exceeds the precision of established methods in the literature:

- **European Options:** At $N = 64$, the engine matches Black-Scholes exact values to 15 decimal places, matching the COS method of Fang and Oosterlee (2008).
- **Heston Dynamics:** With a 2D tensor basis ($N = 512$), it matches the COS Heston benchmarks (Fang-Oosterlee 2008, Table 2) to 10^{-8} precision with a deterministic residual interval that is $10^5\times$ tighter than a 1M-path Monte Carlo standard error.
- **American Exercise:** For American puts, the engine matches the classic Longstaff-Schwartz (2001) values within 1% but runs about 24× faster (21.5 ms versus 512.3 ms, §6.1) and entirely deterministically.

7.2 The Boundaries: Dimensionality, Discontinuities, and Stiffness

Three specific regimes degrade performance or break the method entirely.

1. High-Dimensional State Augmentation (The Dimensionality Wall). If the contract demands more than ~ 5 continuous factors, the tensor basis (N^d) explodes. For a 10-asset basket option, the projected generator requires N^{10} terms, which is strictly infeasible. This is confirmed against the Broadie-Glasserman (2004) stochastic mesh benchmark: the PG matches BG up to $d = 3$, strains at $d = 5$, and breaks at $d = 10$. In this high-d regime, Monte Carlo remains the only practical tool.

2. Non-smooth Payoffs and Barriers (The Regularity Boundary). If the payoff has a sharp discontinuity (e.g., a Digital option $g(x) = \mathbf{1}_{x>K}$), a global polynomial basis suffers from Gibbs oscillations. Convergence slows from exponential $O(\rho^{-N})$ to algebraic $O(N^{-1})$. At $N = 500$, the PG error is 0.0018, which takes 52ms. A simple finite-difference scheme achieves 0.0004 error in 3.2ms. For discontinuous payoffs in $d = 1$, finite differences or domain-adapted bases are strictly superior.

3. Stiff Generators (The Resolution Boundary). For models with extreme mean-reversion (e.g., CIR with $\kappa = 500$ and $T = 0.01$), the generator matrix develops large eigenvalues that inflate the matrix-exponential error bound. While the actual computed price remains accurate at small N, the *certificate* becomes catastrophically loose ($10^6\times$ too wide) until $N > 3\kappa/\sigma$. This requires higher resolution to re-establish tight residual bounds.

7.3 Ablation Study: Why Every Component is Necessary

The projected-generator engine is a composition of architectural decisions. Systematically ablating components on a separate 3-asset Phoenix autocallable instance — notional normalized to 100, distinct from the headline contract of §6.3, so its absolute price, error, and residual are not directly comparable — demonstrates that each layer exists because a specific failure mode requires it:

Configuration	Price	Error vs MC	Residual ε	Verdict
Full engine	100.72	0.43%	0.89	Baseline (70 ms)
No correlation correction ($A_{\text{corr}} = 0$)	94.31	6.76%	N/A	**Broken** — systematic 7% bias on multi-asset.
No temporal conditioning ($K_t = 0$)	97.82	3.29%	3.41	Residual valid but 4× wider.
No event-state automaton	100.72	N/A	0.12	Prices a European — misses autocall/barrier completely.
No residual computation	100.72	0.43%	**None**	Price OK, but approximation debt is hidden.
No Bellman contraction	100.72	0.43%	14.7	Residual 16× too loose.

No single component can be removed without breaking correctness, losing residual control, or degrading the product to a simpler class.

7.4 Regime Classification: The Decision Rule

These boundaries can be codified into a sharp decision rule for choosing between the projected generator and its alternatives.

Theorem (Regime Classification). *Consider a pricing problem characterized by payoff regularity s ($s = \infty$ for analytic), state dimension d, and generator tractability τ ($\tau = 1$ if closed-form, 0 otherwise). The optimal method depends strictly on the regime:*

Regime	s	d	τ	Best method	Error control	Rate
Smooth, low-dim, tractable	∞	≤ 5	1	**Projected generator**	Deterministic residual $O(\rho^{-N})$	Exponential
Smooth, low-dim, intractable	∞	≤ 3	0	Finite differences	Grid convergence study	$O(h^2)$

Regime	s	d	τ	Best method	Error control	Rate
Smooth, high-dim, tractable	∞	> 5	1	Hybrid MC-PG	Per-factor residual	$O(P^{-1/2})$
Non-smooth, low-dim, tractable	$< \infty$	≤ 5	1	PG with adapted basis	Algebraic residual decay	Algebraic
High-dim (any s, τ)	any	> 5	any	**Monte Carlo**	Sampling standard error	$O(P^{-1/2})$

A practitioner facing a new pricing problem should ask three questions:

1. *Can I write down the generator in closed form?* If no → Monte Carlo.
2. *Is the effective dimension* ≤ 5*?* If no → Monte Carlo.
3. *Is the payoff smooth?* If no → PG works but loses exponential speed. Compare against Finite Differences.

If all three answers are favorable, the projected generator dominates: deterministic, exponentially convergent, and equipped with noise-free analytical Greeks. The honest claim of this paper is not "PG replaces MC," but "PG structurally dominates in a precisely characterized regime that covers most equity, rates, FX, and commodity derivatives on 1–5 underlyings."

8. Conclusion

The boundaries leave the final question: what has the projected-generator primitive delivered, and what should be specialized next?

The generator is the natural object for derivative pricing. It encodes the local law of the stochastic process; projecting it onto a finite basis converts that law into a matrix; and matrix exponential propagation replaces both path simulation and PDE grid discretization. The result is a deterministic pricing method that returns the same price and Greeks on every call, with exactness and approximation debt visible through finite residuals.

This paper establishes the base method: the projected generator, finite event-state path dependence, operator Greeks, exact closure on invariant subspaces, and residual accounting when invariance fails. The central methodological claim is that any problem providing a tractable generator and a finite decision structure can be priced by the same project–propagate–decide primitive, with errors tracked rather than hidden.

The verification architecture reflects deliberate separation of concerns. The primary mechanized proof kernel (274 named theorems, 747/747 checks, zero undisclosed axioms in the finite-arithmetic layer) verifies the finite-operator algebra: if the closure residual is zero, the price is exact; if nonzero, the residual marks the approximation debt. A representative Lean 4 stamp type-checks selected arguments with explicit assumptions. The benchmark suite validates numerical accuracy against established methods (Black-Scholes, Longstaff-Schwartz, Fang-Oosterlee COS, binomial trees, finite differences, and Monte Carlo).

The immediate open question is scaling. Tensor decomposition buys tractability to $d \approx 5$ factors; beyond that, either low-rank approximation of the correlation correction or hybrid Monte Carlo–spectral schemes are needed. For the 1–5 factor regime that covers many equity, rates, FX, and commodity derivatives, the projected generator offers a concrete alternative: deterministic, regression-free, and fast enough for intraday pricing and risk.

8.1 Contributions

The core paper has ten primary contributions, followed by six verified extensions included for traceability and scientific context but not depended on by the base method. This keeps the main claim narrow: projected generators are a deterministic operator engine for path-dependent pricing, with an explicit certificate layer.

1. **Computational Primitive:** It frames American and exotic derivative pricing directly as finite-dimensional generator propagation, decoupling the mathematical method from any specific spectral or local basis.
2. **Exactness & Certificates:** It formalizes and mechanically verifies the exactness criterion of polynomial diffusions, extending it to deterministic pricing error certificates. It separates exactness from approximation: invariant finite spaces give exact closure, while non-invariant spaces expose computable residuals.
3. **Convergence Rates:** It formalizes and verifies exponential convergence: for analytic payoffs and spectral bases, pricing error decays as $O(ho^{-N})$. The formal proof chain establishes the complexity separation against Monte Carlo ($O(P^{-1/2})$) and finite differences ($O(h^2)$) with explicit constants.
4. **Path Dependence & Automata:** It reformulates American exercise as regression-free backward propagation. It demonstrates that barriers, autocalls, and coupon memory reduce to event-state propagation around the same projected-generator primitive.
5. **Graph-State Certificates:** It proves (Part 16) that the deterministic error certificate holds for *any* finite graph-state system coupled to the projected generator: the aggregate certificate is bounded by the per-state bound, regardless of graph topology.
6. **The Master Theorem:** It proves (Part 24) the *path-dependent polynomial diffusion theorem*: for polynomial diffusions coupled to finite event-state contracts, the total pricing error decomposes into four independent sources (spatial, temporal, graph-residual, truncation), of which the first three are exactly zero under polynomial invariance.
7. **Regime Classification:** It proves (Part 25) the *method-selection theorem*: a complete characterization of when projected-generator pricing is optimal versus Monte Carlo and finite differences, identifying the sharp boundaries where the method's dominance ends (e.g., discontinuous payoffs, high-dimensional state spaces).

Beyond these core claims, the mechanized kernel carries **six verified extensions** (Parts 26–32), recorded for traceability but not relied on by the base method: the machine-learning bridge (the shared spectral condition-number identity with in-context gradient descent, deferred to a companion paper); the rough-volatility Markovian lift (§A.6); transform-method subsumption (COS, Carr–Madan, Fourier inversion as projected-generator specializations); Greek certificates (deterministic sensitivities versus bump-and-reprice); certified value iteration (the reinforcement learning bridge);

and the substantive derivation layer that discharges several previously assumed extension facts. The full per-part account is in Appendix D.

8.2 Future Work

The same project–propagate–decide primitive is intended to be specialized to several application domains. These are planned directions rather than completed results: at the time of writing none of the companion papers below exists, and this paper claims none of their specific findings. They are listed to indicate the architecture's intended reach.

- **American and Bermudan exercise.** A dedicated treatment of regression-free Snell-envelope backward induction, with an early-exercise-premium analysis and multi-step error accumulation bounds.
- **Structured products and autocallables.** Event-state automata for Phoenix notes, barriers, and coupon-memory contracts, with deterministic Greeks and scenario-level validation against simulation references.
- **Deterministic error certificates.** A standalone audit language that turns the residual, truncation, and propagation components into a portfolio-level error record — the formalization deferred from §5 and §6.
- **Physical asset dispatch.** Power, battery, and hydro control via a projected generator coupled to an automaton compiler, targeting certified dual bounds.
- **Mean-field fleet coordination.** Composition of single-asset pricing across large fleets through a mean-field limit, with risk-measure monotonicity.
- **Spectral importance sampling.** Extending the framework from pricing into tail-risk estimation via eigenvalue-conditioned measure changes.
- **Machine-learning bridge.** The condition-number identity linking the projected-generator spectral rate to in-context gradient descent, developed for a machine-learning audience.

Each direction reuses Layers 1–3 of the present method — generator projection, semigroup propagation, and residual control — and adds only a domain-specific decision structure on top. The base primitive proven and benchmarked here is the shared foundation; the applications are future work.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author used Cursor and its integrated AI models (Claude and GPT families) in order to assist with formal proof mechanization, structural editing, and language polishing. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the publication.

References

- Ackerer, D. and Filipović, D. (2020). Option pricing with orthogonal polynomial expansions. *Mathematical Finance*, 30(1), 47–84.

- Ahn, K., Cheng, X., Daneshmand, H. and Sra, S. (2023). Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 36.
- Bertsekas, D. P. (2012). *Dynamic Programming and Optimal Control, Vol. II: Approximate Dynamic Programming*. 4th ed. Athena Scientific.
- Black, F. and Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3), 637–654.
- Brown, D. B., Smith, J. E. and Sun, P. (2010). Information relaxations and duality in stochastic dynamic programs. *Operations Research*, 58(4), 785–801.
- Broadie, M. and Glasserman, P. (2004). A stochastic mesh method for pricing high-dimensional American options. *Journal of Computational Finance*, 7(4), 35–72.
- Boyd, J. P. (2001). Chebyshev and Fourier Spectral Methods. *Dover Publications*.
- Cuchiero, C., Keller-Ressel, M. and Teichmann, J. (2012). Polynomial processes and their applications to mathematical finance. *Finance and Stochastics*, 16(4), 711–740.
- Cont, R. and Tankov, P. (2004). *Financial Modelling with Jump Processes*. Chapman & Hall/CRC.
- Fang, F. and Oosterlee, C. W. (2008). A novel pricing method for European options based on Fourier-cosine series expansions. *SIAM Journal on Scientific Computing*, 31(2), 826–848.
- Feynman, R. P. (1948). Space-time approach to non-relativistic quantum mechanics. *Reviews of Modern Physics*, 20(2), 367–387.
- Filipović, D. and Larsson, M. (2016). Polynomial diffusions and applications in finance. *Finance and Stochastics*, 20(4), 931–972.
- Galerkin, B. G. (1915). Rods and plates. Series in some problems of elastic equilibrium of rods and plates. *Vestnik Inzhenerov i Tekhnikov*, 19, 897-908.
- Glasserman, P. (2003). *Monte Carlo Methods in Financial Engineering*. Springer.
- Kac, M. (1949). On distributions of certain Wiener functionals. *Transactions of the AMS*, 65(1), 1–13.
- Kolmogorov, A. N. (1931). On analytical methods in probability theory. *Mathematische Annalen*, 104(1), 415–458.
- Lehmann, E. L. and Casella, G. (1998). *Theory of Point Estimation*. 2nd ed. Springer.
- Longstaff, F. A. and Schwartz, E. S. (2001). Valuing American options by simulation: a simple least-squares approach. *Review of Financial Studies*, 14(1), 113–147.
- Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1-2), 125–144.
- Nagy, T. (2026). The Latent: Finite Sufficient Representations of Smooth Systems. *Zenodo*. DOI: 10.5281/zenodo.19101209.
- Øksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications*. 6th ed. Springer.

- Reed, M. and Simon, B. (1978). *Methods of Modern Mathematical Physics*. Academic Press.
- Trefethen, L. N. (2013). Approximation Theory and Approximation Practice. *SIAM*.
- von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A. and Vladymyrov, M. (2023). Transformers learn in-context by gradient descent. *International Conference on Machine Learning (ICML)*, 35151–35174.

Appendix A: Numerical Examples

This appendix gives the numerical details behind the algorithmic form of §4.6. The first example is a plain European call. The second is the Longstaff-Schwartz American option benchmark, rewritten in projected-generator form.

A.1 European Call

We begin with the simplest possible case: a standard European option, which isolates the baseline projection and propagation steps without early exercise complexity. Consider a Black-Scholes stock with

$$S_0 = 100, \qquad K = 100, \qquad r = 5\%, \qquad \sigma = 20\%, \qquad T = 1.$$

The payoff is $g(S) = \max(S - K, 0)$. In the projected-generator calculation we choose a finite basis $\{\phi_0, \dots, \phi_{N-1}\}$ on a bounded computational domain and compute

$$M_{jk} = \langle \phi_j, \mathcal{L}\phi_k \rangle, \qquad A = M - rI, \qquad \mathbf{c}_0 = \exp(AT)\mathbf{g},$$

where $\mathbf{g}$ is the projected payoff vector. For Black-Scholes,

$$\mathcal{L}f(S) = rSf'(S) + \frac{1}{2}\sigma^2 S^2 f''(S).$$

The Black-Scholes closed-form reference for this contract is

Quantity	Value
Black-Scholes reference	10.4505835722
Payoff	$\max(S - 100, 0)$
Exercise decision	none
Propagation steps	one

This is the cleanest projected-generator contract. There is no early exercise, no regression, and no path memory. The computation is payoff projection, one matrix exponential, and evaluation at S_0.

The step-by-step validation calculation implements this contract using a local support basis (piecewise-linear hat functions, $N = 51$) on the domain $[0, 300]$. This exposes the full algebraic structure: the Gram matrix G, the generator matrix M, the ODE matrix A, the propagator P, the payoff projection, and the backward step.

Spreadsheet field	Value
Basis	Piecewise Linear (Hat)
N	51
Domain	$[0, 300]$
Projected-generator price	10.498552
Black-Scholes price	10.450584

(Note: A naive attempt using global Chebyshev polynomials in raw price space fails dramatically for this payoff, yielding a toxic negative price of -49637*. This instability is a known property of global polynomials on non-smooth payoffs (the Runge phenomenon). This is why basis selection is critical. See Appendix B for a guide to basis selection.)*

Stable projected-generator European pricing uses either a log-space spectral/COS basis or a local basis. The paper's benchmark surface (§7.3) records the stable spectral case: at $N = 64$, the European option errors are $O(10^{-15})$ against Black-Scholes references.

Thus the European call example has two lessons. Algebraically, it is the pure semigroup case $\mathbf{c}_0 = \exp(AT)\mathbf{g}$. Numerically, it is the smallest example that exposes the basis-selection requirement.

A.2 American Put: Longstaff-Schwartz Benchmark

The second example uses the economic contract from the Longstaff-Schwartz 8-path example:

- non-dividend-paying stock;
- $S_0 = 1.00$;
- strike $K = 1.10$;
- risk-free rate $r = 0.06$, so one-year discounting is $e^{-0.06} = 0.94176$;
- exercise dates $t = 1, 2, 3$, with final maturity at $t = 3$.

The original example specifies simulated stock paths, but it does not specify a volatility or a full continuous stochastic model. A projected-generator calculation needs that model because the generator is the input. For this numerical comparison, we use the Black-Scholes generator and calibrate σ so that the model-based Bermudan reference price matches the 8-path Longstaff-Schwartz value 0.1144. This gives $\sigma \approx 18.736\%$. The calibration is only a numerical alignment device; the eight paths do not uniquely identify a volatility.

For a transparent hand calculation, start with the minimal polynomial basis

$$\phi_0(S) = 1, \qquad \phi_1(S) = S, \qquad \phi_2(S) = S^2.$$

The Black-Scholes generator maps these basis functions as follows:

k	ϕ_k	ϕ_k'	ϕ_k''	$\mathcal{L}\phi_k$
0	1	0	0	0
1	S	1	0	$0.06S$
2	S^2	$2S$	2	$0.1551S^2$

Thus

$$M = \mathrm{diag}(0,\ 0.06,\ 0.1551), \qquad A = M - rI = \begin{bmatrix} -0.06 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.0951 \end{bmatrix}.$$

For one year, the propagation matrix is

$$P = \exp(A) = \begin{bmatrix} 0.9418 & 0 & 0 \\ 0 & 1.0000 & 0 \\ 0 & 0 & 1.0998 \end{bmatrix}.$$

The maturity payoff is $g(S) = \max(1.10 - S, 0)$. Projecting it into the $N = 3$ basis gives

$$\mathbf{c}_3 = \begin{bmatrix} 1.1345 \\ -1.3211 \\ 0.3643 \end{bmatrix}.$$

The backward induction alternates between propagation and exercise:

$$\tilde{\mathbf{c}}_2 = P\mathbf{c}_3 = \begin{bmatrix} 1.0684 \\ -1.3211 \\ 0.4006 \end{bmatrix}.$$

The continuation function is

$$\widetilde{V}_2(S) = 1.0684 - 1.3211S + 0.4006S^2.$$

At $S = 1.00$, continuation is 0.1479 while exercise is 0.10, so the holder continues. At lower stock values exercise can dominate; after applying the pointwise max and re-projecting,

$$\mathbf{c}_2 = \begin{bmatrix} 1.1647 \\ -1.4125 \\ 0.4222 \end{bmatrix}.$$

Repeating the same operation gives

$$\tilde{\mathbf{c}}_1 = P\mathbf{c}_2 = \begin{bmatrix} 1.0969 \\ -1.4125 \\ 0.4643 \end{bmatrix}, \qquad \mathbf{c}_1 = \begin{bmatrix} 1.1824 \\ -1.4923 \\ 0.4821 \end{bmatrix}.$$

The final propagation to $t = 0$ gives

$$\mathbf{c}_0 = P\mathbf{c}_1 = \begin{bmatrix} 1.1135 \\ -1.4923 \\ 0.5302 \end{bmatrix}.$$

Evaluating at $S_0 = 1.00$,

$$V_0^{N=3} = 1.1135 - 1.4923 + 0.5302 = 0.1514.$$

This $N = 3$ example is deliberately small so that the arithmetic fits on the page. It overprices the option because three smooth polynomials cannot resolve the payoff kink at $K = 1.10$. Production runs use a localized basis. The same algorithm then converges to the model reference:

Basis size N	Price at $S_0 = 1.00$
3 (monomials)	0.1514
101 (hat functions)	0.114375
201 (hat functions)	0.114418
401 (hat functions)	0.114435
801 (hat functions)	0.114437

A 3000-step Bermudan binomial reference under the same calibrated Black-Scholes model is 0.114440.

For comparison, the eight simulated paths used by Longstaff and Schwartz are:

Path	$t = 0$	$t = 1$	$t = 2$	$t = 3$
1	1.00	1.09	1.08	1.34
2	1.00	1.16	1.26	1.54
3	1.00	1.22	1.07	1.03
4	1.00	0.93	0.97	0.92
5	1.00	1.11	1.56	1.52
6	1.00	0.76	0.77	0.90
7	1.00	0.92	0.84	1.01
8	1.00	0.88	1.22	1.34

The Longstaff-Schwartz regression procedure gives the cash flows

Path	$t = 1$	$t = 2$	$t = 3$
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.00	0.00	0.07
4	0.17	0.00	0.00
5	0.00	0.00	0.00
6	0.34	0.00	0.00
7	0.18	0.00	0.00
8	0.22	0.00	0.00

Discounting and averaging gives

$$V_0^{LSM} = \frac{1}{8}\left(0.07e^{-0.18} + 0.17e^{-0.06} + 0.34e^{-0.06} + 0.18e^{-0.06} + 0.22e^{-0.06}\right) = 0.1144.$$

The comparison shows exactly what changes between the two methods. Longstaff-Schwartz estimates continuation values by regression on simulated paths. The projected generator propagates the whole continuation function by a matrix exponential and then applies the exercise max directly.

A.3 Dividends and Event Shifts

The Longstaff-Schwartz example is non-dividend-paying, so the dividend amount is $D = 0$. If a deterministic cash dividend D is paid at time t_d, the projected-generator treatment is still simple. The stock jumps from S to $S - D$, so the option value just before the dividend is

$$V(S, t_d^-) = V(S - D, t_d^+).$$

In coefficient form, this is a spatial shift. For example,

$$(S - D)^2 = S^2 - 2DS + D^2.$$

Thus the dividend event is represented by a constant shift matrix S_D:

$$\mathbf{c}(t_d^-) = S_D \mathbf{c}(t_d^+).$$

The pricing loop remains the same: propagate by $\exp((M - rI)\Delta t)$, apply the dividend shift when a dividend date is crossed, compare continuation with exercise at exercise dates, and re-project. No path storage or regression logic is introduced by the dividend event.

A.4 Merton Jump-Diffusion Benchmark

The finite-activity jump validation (Theorem 5, §5.8) uses the Merton jump-diffusion as a controlled test case. Under the risk-neutral model

$$\frac{dS_t}{S_{t^-}} = (r - \lambda\kappa_J)\,dt + \sigma\,dW_t + (e^Y - 1)\,dN_t, \qquad \kappa_J = \mathbb{E}[e^Y - 1],$$

where N_t is a Poisson process with intensity λ and $Y \sim \mathcal{N}(m_J, \delta_J^2)$, the log-price characteristic exponent is

$$\psi(u) = iu(r - \lambda\kappa_J - \tfrac{1}{2}\sigma^2) - \tfrac{1}{2}\sigma^2 u^2 + \lambda\left(e^{ium_J - \frac{1}{2}\delta_J^2 u^2} - 1\right).$$

The projected-generator computation in jump_diffusion_pgm_validation.py uses a Fourier/COS basis. For frequency $u_k = k\pi/(b - a)$, the finite generator is diagonal with entries $\psi(u_k)$, so one propagation step multiplies mode k by $\exp(T\psi(u_k))$. This is the matrix exponential $\exp(TM_N)$ written in the basis that diagonalizes the translation-invariant jump operator.

The independent reference is not a COS calculation. It conditions on the jump count and uses the exact Poisson mixture of Black-Scholes prices:

$$V_{\text{Merton}} = \sum_{n=0}^{\infty} e^{-\lambda T} \frac{(\lambda T)^n}{n!} V_{\text{BS}}\left(S_0 e^{-\lambda \kappa_J T + n m_J + \frac{1}{2} n \delta_J^2}, K, r, \sqrt{\sigma^2 + n\delta_J^2/T}, T\right).$$

The benchmark includes three checks:

Case	Purpose	Reference	PGM N=128 abs. error
ATM call, downward jumps	typical Merton call	13.9638790873	1.99×10^{-12}
OTM put, stress jumps	jump-heavy downside stress	14.2501806896	6.92×10^{-12}
$\lambda = 0$ sanity check	collapse to Black-Scholes	10.4505835722	1.07×10^{-14}

The benchmark therefore tests both the positive jump-extension claim and the degenerate no-jump limit. It validates only finite-activity Poisson jumps with a known Lévy symbol. It does not validate state-dependent intensities, infinite-activity jump measures, or jump mechanisms whose characteristic exponent is not analytic on the projection strip.

A.5 Market Calibration and Synthetic Stress Boundaries

The benchmark suite includes market-representative pricing tests using parameters drawn from observable market data (May 2026 snapshots):

Contract	Parameters	PG Price	Reference	Error
SPX ATM Put (1yr)	S=5350, =18.5%, r=4.25%, q=1.4%	$318.18	BS European	exact (deterministic)
SPX Deep OTM Put (3mo)	S=5350, K=4280, =22% (skew)	$3.47	BS $3.45	0.5% (early ex premium)
SX5E ITM Put (2yr)	S=4850, K=5335, =22%, r=q=2.5%	$979.98	BS $901.10	8.8% early ex premium
DE Power Call (OU, 6mo)	S=€48, K=€55, =35, =€45, =€18	€0.0024	OU analytic	exact

The American rows (deep OTM and ITM puts) are compared against a Black–Scholes *European* baseline, so the reported gap is the early-exercise premium, not a validated pricing error — a positive gap is expected for an in-the-money American put. An independent American reference (binomial or finite-difference) for these specific market-calibrated cases is left to the implementation companion; the controlled American-put accuracy claim rests on the binomial benchmark of §6.2.

The calibration scripts (available in the companion repository) achieve production-grade fit quality:

- SPX forward curve RMSE: 0.14% (Grade A)
- Vol surface fit (exponential decay): RMSE 0.28 vol pts, max error 0.43 vol pts
- Power market OU forward fit: RMSE 1.34% (Grade B)

Tail accuracy. A dedicated stress test covers extreme parameters — deep OTM (30-40%), long maturity (5-10yr), high volatility (50%), and combined stress cases. Results at N=128 basis functions: mean error 0.23%, max error 0.82% across all standard cases. Convergence is exponential (18+ bits gained per doubling at high N). The only slow-convergence regime is very low vol with short maturity (=10%, T=0.1yr), where the near-discontinuous exercise boundary requires N>200 for sub-1% accuracy — a known Gibbs limitation addressable by exercise boundary smoothing.

Synthetic Validation Boundary. The Phoenix autocallable benchmarks remain strictly synthetic. We use no live market quotes, dealer marks, or issuer term sheets for the structured product validation. The benchmark suite evaluates six synthetic Phoenix term sheets that vary correlation, barrier schedules, coupons, volatility levels, and basket sizes. Live structured product validation requires a full calibration input bundle and independent market marks, which fall outside the scope of this study. The market-calibrated American and European benchmarks (§6.4) demonstrate that the engine produces correct prices when fed market-consistent parameters.

A.6 Rough Volatility via the Markovian Lift

Part 28 lifts the rough-volatility limitation by approximating the fractional kernel $K(t) = t^{H-1/2}/\Gamma(H+1/2)$ with an n-factor sum of exponentials, which is the kernel of an affine (hence projected-generator-exact) Markovian process. For $H = 0.1$ — deep in the rough regime where no standard generator exists on the raw process — a least-squares exponential fit over a 2-year maturity window gives a kernel approximation error that decreases monotonically in the factor count, from 5.6×10^{-1} at $n = 1$ to 4.6×10^{-4} at $n = 32$ (a factor of 1200; rough_vol_lift_results.json). The lift error is the first of the two deterministic certificate components (the second is the usual spectral truncation); their sum is a deterministic pricing certificate for rough volatility.

We also test the lift at the option-price level. In rough_heston_lift_benchmark_results.json, an ATM European call is priced under a conservative rough-Heston Markovian-lift simulation with $H = 0.1$, $v_0 = 4\%$, $\kappa = 1.5$, $\theta = 4\%$, $\xi = 4\%$, $\rho = -0.7$, $T = 1$, and 12,000 common Monte Carlo paths. A 64-factor lift is used as the internal numerical reference. All factor counts converge rapidly:

Lift factors	Call price	MC SE	Abs. error vs 64-factor reference	Kernel rel. L2 error
4	8.2552	0.1186	0.00000	4.71×10^{-2}
8	8.2552	0.1186	0.00002	4.60×10^{-2}
16	8.2552	0.1186	0.00001	4.58×10^{-2}
32	8.2552	0.1186	0.00000	4.58×10^{-2}
64	8.2552	0.1186	0.00000	4.58×10^{-2}

Crucially, this Markovian-lift price is validated against two completely independent references (rough_heston_cf_reference_results.json): (i) the Black-Scholes analytical price with $\sigma = \sqrt{v_0} =$

20%, which gives 8.4333 (the exact limit for $\xi \to 0$); and (ii) a direct Volterra MC simulation that uses the full fractional convolution integral — no exponential lift — and obtains 8.4168 ± 0.0601. All three methods agree within statistical tolerance (direct Volterra vs lift: 1.2 combined SE), confirming that the Markovian lift correctly translates into option prices.

This is not a market calibration and not a published closed-form reference. It is an option-price-level lift-convergence test with independent cross-validation, showing that the Markovian lift pathway is numerically sound for the rough-volatility companion paper.

A.7 Empirical Bounds: Validating the Error Certificates

The mechanized proof suite (Parts 18–23) makes formal claims about closure residuals, convergence rates, complexity separation, and temporal exactness. The following table validates these claims against floating-point computation on concrete models. Full numerical details are in certificate_validation_results .md.

Part	Theorem	Formal claim	Empirical result
18	T131	OU closure residual = 0 in Hermite basis	Exact 0 at all N tested (5–50)
18	T134–T136	Truncation error decays geometrically per mode	Ratio ≈ 0.31 per odd mode, constant to 4 digits
19	T141	PG $O(\log 1/\varepsilon)$ vs MC $\Omega(1/\varepsilon^2)$	MC/PG cost ratio: $2\times$ at $\varepsilon = 0.1$, $57\times$ at 10^{-2}, $268{,}000\times$ at 10^{-4}
20	T144–T145	CIR closure residual = 0 in polynomial basis	Exact 0 at all N tested (5–50)
22	T155–T161	Zero temporal error on invariant subspace	Matrix-exp error $< 10^{-15}$; Euler (1000 steps) error $\sim 10^{-4}$
23	T168	Regularity hierarchy: disc $<$ Lip $<$ C^2 $<$ analytic	Confirmed; analytic reaches 10^{-8} at $N = 30$

The validation uses orthonormal probabilists' Hermite functions for the OU/analytic tests and explicit monomial-basis construction for the CIR degree-preservation test. The payoff $f(x) = e^{-x^2/4}$ serves as the analytic benchmark; indicator, hat, and bump functions test the lower regularity classes. ## Appendix B: Guide to Basis Selection and Regularization

A naive application of the projected-generator method using global polynomials directly in raw price space can be unstable for non-smooth payoffs. A European call payoff has a discontinuous first derivative at the strike. Projecting such functions onto high-degree global polynomials produces Runge or Gibbs-type oscillations near the kink and the domain boundaries. Because the method propagates coefficients backward via the matrix exponential, these projection errors are amplified, leading to unphysical results (such as negative option prices).

To prevent this toxic oscillatory behavior, the system must incorporate specific mathematical and structural safeguards (regularization techniques):

B.1 Structural Safeguard: Local Support Bases

Instead of covering the entire domain with a single global polynomial, the domain is partitioned into small segments using piecewise-linear (hat) functions or cubic B-splines.

- **Mechanism:** A local basis function is non-zero only on its specific sub-interval. If the payoff kinks at K, it only affects the basis functions strictly overlapping K. The basis functions at distant nodes remain completely undisturbed.
- **Result:** This eliminates global oscillations and naturally accommodates kinks, producing highly sparse, stable generator matrices in raw price space.

B.2 Geometric Safeguard: Log-Space Transformation

Instead of working in the raw price space (S), the problem is transformed into log-price space ($x = \ln S$).

- **Mechanism:** The Black-Scholes generator in raw price space contains an S^2 diffusion term ($\frac{1}{2}\sigma^2 S^2 \frac{\partial^2}{\partial S^2}$), which grows unbounded as $S \to \infty$. This unbounded growth interacts catastrophically with global polynomials. By substituting $x = \ln S$, the generator becomes a constant-coefficient operator ($\frac{1}{2}\sigma^2 \frac{\partial^2}{\partial x^2} + (r - \frac{1}{2}\sigma^2)\frac{\partial}{\partial x}$).
- **Result:** Chebyshev and Fourier bases are exceptionally stable and exponentially convergent when applied to constant-coefficient operators in log-space.

B.3 Physical Safeguard: Payoff Smoothing (Mollification)

The root cause of the instability is the infinite sharpness of the payoff kink (the derivative jumps).

- **Mechanism:** Before projecting the payoff onto the basis, the payoff is "mollified" (smoothed) over a microscopically small time step $\epsilon \ll \Delta t$. For example, one can apply the exact analytical Black-Scholes formula for $\epsilon = 10^{-4}$ years to the payoff.
- **Result:** The projection operator now sees a smooth, infinitely differentiable (C^∞) function rather than a sharp kink. Global polynomials can approximate this smoothed function without triggering Runge oscillations.

B.4 Mathematical Safeguard: Spectral Filtering (Damping)

If global polynomials are strictly required for performance, the higher-order (high-frequency) terms can be artificially damped.

- **Mechanism:** Oscillations are driven by the highest-index coefficients ($c_{N-10}, \ldots, c_{N-1}$). A spectral filter (such as an exponential filter or Tikhonov regularization) penalizes high derivatives during the projection step.
- **Result:** The resulting polynomial remains smoother and is prevented from "bending too hard" to fit the kink, sacrificing a tiny amount of sharpness at the strike to guarantee global stability.

B.5 Logical Safeguard: Positivity-Preserving Projection

Standard Galerkin projection minimizes the squared error (L^2 norm) without regard for the financial reality that option prices cannot be negative.

- **Mechanism:** The projection is formulated not as a simple matrix multiplication, but as a constrained optimization problem: *Find the coefficients* $\mathbf{c}$ *that minimize the projection error, subject to the constraint that* $V(S) \geq 0$ *for all* S.
- **Result:** This acts as a hard floor, guaranteeing that the numerical method will never return a toxic negative price, even if the underlying basis struggles with the payoff shape.

Appendix C: Proofs

This appendix collects the longer proofs of the theorems stated in §5. The statements, intuition, model tables, and consequences remain in the main text; the detailed arguments are gathered here so the accuracy section reads as a narrative rather than a proof sheet. Short proof sketches stay inline in §5.

C.1 Polynomial-Degree Invariance Criterion

Criterion 1 (Polynomial-Degree Invariance Criterion). *Let $\mathcal{L}$ be the generator of a one-dimensional diffusion with polynomial drift $\mu(x)$ (degree p) and polynomial diffusion coefficient $\sigma^2(x)$ (degree q), defined by:

$$\mu(x) = \sum_{j=0}^{p} a_j x^j, \quad \sigma^2(x) = \sum_{j=0}^{q} b_j x^j$$

and let $\mathcal{H}_N = \text{span}\{1, x, x^2, \dots, x^{N-1}\}$ be the polynomial basis of degree $\leq N-1$. Then $\mathcal{L}$ raises the degree of a monomial by a fixed amount,*

$$\deg \mathcal{L}(x^k) = k + d_{\mathcal{L}}, \qquad d_{\mathcal{L}} := \max(p-1,\, q-2), \quad (k \geq 2),$$

so invariance does not depend on N: either every $\mathcal{H}_N$ ($N \geq 1$) is invariant, or none is. Precisely,

$$\mathcal{L}\mathcal{H}_N \subseteq \mathcal{H}_N \text{ for all } N \iff d_{\mathcal{L}} \leq 0 \iff p \leq 1 \text{ and } q \leq 2.$$

If $d_{\mathcal{L}} > 0$ — the generator strictly raises polynomial degree — then no finite polynomial subspace is invariant, regardless of how large N is. There is no "take N large enough" threshold: degree non-elevation is the decisive, N-independent condition.

The number $\max(p, q)+1$ — the smallest basis degree that can represent the coefficient polynomials μ, σ^2 themselves — is a representational quantity, not the invariance threshold. For instance CEV with $\beta > 1$ has $d_{\mathcal{L}} = 2\beta - 2 > 0$ and admits no invariant polynomial subspace at any N, whereas GBM in price space has $d_{\mathcal{L}} = 0$: its monomials are eigenfunctions, $\mathcal{L}(x^k) = [rk + \frac{1}{2}\sigma^2 k(k-1)]\, x^k$, so every $\mathcal{H}_N$ is invariant — the price-space basis is avoided not for lack of invariance but because the lognormal moment problem is ill-conditioned.

Proof. The generator $\mathcal{L}f = \mu f' + \frac{1}{2}\sigma^2 f''$ applied to x^k gives:

$$\mathcal{L}(x^k) = k \sum_{j=0}^{p} a_j x^{j+k-1} + \frac{k(k-1)}{2} \sum_{j=0}^{q} b_j x^{j+k-2}.$$

The highest degree term in the first sum is x^{p+k-1}; in the second sum, x^{q+k-2}. For $k \geq 2$ both terms are present, so $\deg(\mathcal{L}(x^k)) = k + d_{\mathcal{L}}$ with $d_{\mathcal{L}} = \max(p-1, q-2)$; the low-order cases are $\deg \mathcal{L}(x) = p$ and $\mathcal{L}(1) = 0$. Invariance of $\mathcal{H}_N$ requires $\deg \mathcal{L}(x^k) \leq N-1$ for every $k \leq N-1$. Since $\deg \mathcal{L}(x^k)$ is increasing in k, the binding constraint is the top monomial $k = N-1$:

$$(N-1) + d_{\mathcal{L}} \leq N-1 \iff d_{\mathcal{L}} \leq 0.$$

The N cancels — the criterion is $d_{\mathcal{L}} \leq 0$, i.e. $p \leq 1$ and $q \leq 2$, independent of N (the low-order constraints $p \leq N-1$ are then automatic for $N \geq 2$, and $\mathcal{H}_1 = \{1\}$ is invariant since $\mathcal{L}(1) = 0$). Hence when $d_{\mathcal{L}} \leq 0$ *every* $\mathcal{H}_N$ is invariant.

Conversely, if $d_{\mathcal{L}} > 0$ then $\deg \mathcal{L}(x^{N-1}) = (N-1) + d_{\mathcal{L}} > N-1$ for every $N \geq 2$, so the top basis monomial always leaves $\mathcal{H}_N$: *no* finite polynomial subspace is invariant. The degree elevation is nonetheless bounded by $d_{\mathcal{L}}$, so the closure residual is $\delta(N) = O(N^{-s})$ with s set by the regularity of the target function, and the method converges algebraically rather than achieving exact closure at finite N.

The multi-factor extension follows by tensorization: for a d-factor model where each factor has polynomial coefficients, the joint generator on the tensor product basis satisfies $\mathcal{L}_{\text{joint}}(\mathcal{H}_{N_1} \otimes \cdots \otimes \mathcal{H}_{N_d}) \subseteq \mathcal{H}_{N_1} \otimes \cdots \otimes \mathcal{H}_{N_d}$ if and only if each factor satisfies the invariance condition independently and the cross terms (from correlation) preserve the tensor product structure. □

C.2 Exponential Convergence of Projected Generator Pricing

This theorem establishes the formal convergence rate of the exact generator method under exponential damping.

Theorem 1 (Exponential Convergence of Projected Generator Pricing). *Assume:*

(A1) The payoff g admits an analytic extension to a strip $|\operatorname{Im}(z)| < \tau$ in the complex plane, with Bernstein ellipse parameter $\rho = e^{\tau} > 1$.

(A2) The basis $\{\phi_k\}_{k=0}^{N-1}$ is a spectral basis (Fourier, Chebyshev, or Hermite) on $[a, b]$ with L^2 orthonormality: $\langle \phi_j, \phi_k \rangle = \delta_{jk}$.

(A3) The generator $\mathcal{L}$ maps $\mathcal{H}_N$ approximately into itself: $\|(I - \Pi_N)\mathcal{L}\Pi_N\|_{\text{op}} \leq \delta(N)$ where $\delta(N) \to 0$ as $N \to \infty$.

(A4) The discount factor satisfies $\beta = e^{-r\Delta t} < 1$.

Under (A1)–(A4), for M-step Bermudan backward induction with N basis functions, the total pricing error satisfies

$$\|V_0 - V_0^N\|_{L^2} \leq \frac{2M_V}{\rho - 1} \cdot \frac{\beta}{1 - \beta} \cdot \rho^{-N} + \frac{\delta(N)}{1 - \beta},$$

where $M_V = \sup_{z \in \mathcal{E}_\rho} |V(z)|$ is the analytic bound on the Bernstein ellipse $\mathcal{E}_\rho$.

Proof. The argument proceeds in three steps.

Step 1 (Single-step projection error via Jackson-Bernstein). Let Π_N denote the L^2-orthogonal projection onto $\mathcal{H}_N$. By Galerkin optimality, the projection error equals the best approximation error:

$$\|f - \Pi_N f\|_{L^2} = \inf_{p \in \mathcal{H}_N} \|f - p\|_{L^2}.$$

For a function f satisfying (A1), the Jackson-Bernstein theorem for spectral bases (Trefethen, 2013, Theorem 8.2) gives

$$\|f - \Pi_N f\|_{L^2} \leq \frac{2M_f}{\rho - 1} \cdot \rho^{-N},$$

where $M_f = \sup_{z \in \mathcal{E}_\rho} |f(z)|$. Each additional basis function gains $\log \rho$ bits of accuracy.

Step 2 (Bellman contraction propagation). At each exercise date t_i, the backward-induction operator has the form

$$\mathcal{T}V = \max(g,\ \beta \cdot P_{\Delta t} V),$$

where $P_{\Delta t} = \exp(\mathcal{L}\Delta t)$ is the transition semigroup. The max operator is a non-expansion in L^2, and scalar multiplication by $\beta < 1$ is a strict contraction. Therefore $\mathcal{T}$ is a β-contraction:

$$\|\mathcal{T}V - \mathcal{T}W\|_{L^2} \leq \beta \|V - W\|_{L^2}.$$

This contraction property is formally verified in the proof suite (Bellman contraction, Part 8). If e_i denotes the single-step projection error at date t_i, the total error after M backward steps satisfies the geometric accumulation bound:

$$\|V_0 - V_0^N\|_{L^2} \leq \sum_{k=0}^{M-1} \beta^k \, e_{M-k} \leq \frac{\beta}{1 - \beta} \cdot \max_i e_i.$$

Substituting the spectral bound from Step 1 yields the first term.

Step 3 (Closure residual accumulation). The projected semigroup $\Pi_N \exp(\mathcal{L}\Delta t)\Pi_N$ differs from the exact semigroup by the closure residual $\delta(N) = \|(I - \Pi_N)\mathcal{L}\Pi_N\|_{\text{op}}$ from assumption (A3). At each backward step, this residual contributes an additive error bounded by $\delta(N)$. Through the same geometric series with contraction factor β, the accumulated closure error is

$$\sum_{k=0}^{M-1} \beta^k \, \delta(N) \leq \frac{\delta(N)}{1 - \beta}.$$

Combining Steps 1–3 gives the stated bound. □

C.3 Complexity Separation

Here we demonstrate that the projected generator fundamentally shifts the complexity bound from a simulation-based regime to an operator-algebra regime.

Theorem 2 (Complexity Separation). *Let $\mathcal{F}_\rho$ denote the class of European pricing problems where the payoff g admits an analytic extension to the Bernstein ellipse $\mathcal{E}_\rho$ with $\rho > 1$, and the generator $\mathcal{L}$ satisfies (A1)–(A3). Then:*

(i) (Upper bound — projected generator.) The projected-generator engine with N basis functions achieves worst-case error

$$\sup_{g\in\mathcal{F}_\rho} |V_{\text{true}} - V_{\text{calc}}^N| \le C_1\rho^{-N},$$

requiring $N = O(\log(1/\varepsilon)/\log\rho)$ evaluations for ε-accuracy.

(ii) (Lower bound — any Monte Carlo method.) Let $\hat{V}_P$ be any estimator of V_{true} constructed from P independent samples of the terminal value $g(X_T)$. Then

$$\inf_{\hat{V}_P}\sup_{g\in\mathcal{F}_\rho} \mathbb{E}[(\hat{V}_P - V_{\text{true}})^2] \ge \frac{\text{Var}(g(X_T))}{P}.$$

Achieving ε-accuracy in mean square requires $P = \Omega(\varepsilon^{-2})$ samples.

(iii) (Separation.) The operational cost ratio is

$$\frac{N_{\text{PG}}(\varepsilon)}{P_{\text{MC}}(\varepsilon)} = \frac{O(\log(1/\varepsilon))}{\Omega(\varepsilon^{-2})} \to 0 \quad \text{as } \varepsilon \to 0.$$

The projected generator requires exponentially fewer operations than any sampling-based method to achieve the same worst-case error on $\mathcal{F}_\rho$.

Proof. Part (i) restates Theorem 1 for the European case ($M = 1$, $\varepsilon_{\text{bridge}} = 0$).

Part (ii) is a minimax lower bound for mean estimation, and the inf is over *all* estimators, not only unbiased ones. The unbiased Cramér-Rao bound (Lehmann and Casella, 1998, Theorem 2.5.1) gives $\text{Var}(g(X_T))/P$ as the floor for the sample mean, but a pointwise-biased estimator (the constant 0 being the trivial case) can beat it at a single g, so the unbiased bound alone does not establish (ii). The minimax claim follows instead from a two-point (Le Cam) argument over the class. Pick two problems $g_0, g_1 \in \mathcal{F}_\rho$ whose true prices differ by $2s$ and whose terminal-value laws have squared Hellinger / KL separation $O(s^2/\sigma^2)$ with $\sigma^2 = \text{Var}(g(X_T))$; such a pair exists in $\mathcal{F}_\rho$ because the class is closed under small analytic perturbations of the payoff. After P i.i.d. draws the two product measures remain statistically indistinguishable with constant probability whenever $P \lesssim \sigma^2/s^2$, so any estimator incurs squared error $\gtrsim s^2$ on at least one of them. Optimizing s yields the worst-case bound $\inf_{\hat{V}_P}\sup_g \mathbb{E}[(\hat{V}_P - V_{\text{true}})^2] \gtrsim \sigma^2/P$, matching the sample-mean rate up to a constant. The key structural point is unchanged: the i.i.d. terminal-value access of Monte Carlo is not a design choice but a consequence of the strong Markov property, so any path-sampling method inherits this $\Omega(1/P)$ floor.

Part (iii) follows by substitution: the PG cost for ε-accuracy is $N = \lceil \log(C_1/\varepsilon)/\log\rho \rceil = O(\log(1/\varepsilon))$, while the MC cost is $P = \lceil \text{Var}(g(X_T))/\varepsilon^2 \rceil = \Omega(\varepsilon^{-2})$. The ratio $\log(1/\varepsilon)/\varepsilon^{-2} = \varepsilon^2\log(1/\varepsilon) \to 0$. □

C.4 Computable Residual Interval

This theorem provides the mechanism for quantifying the exact pointwise approximation debt when basis closure is not perfect.

Theorem 3 (Computable Residual Interval). *Let V_{calc}^N denote the engine output from M-step Bermudan backward induction with basis size N. Under assumptions (A1)–(A4) of Theorem 1, the residual accounting gives*

$$V_{\text{true}} \in \left[V_{\text{calc}}^N - \varepsilon_{\text{total}}(N),\ V_{\text{calc}}^N + \varepsilon_{\text{total}}(N)\right],$$

where the total residual decomposes into four computable components:

$$\varepsilon_{\text{total}}(N) = \underbrace{\varepsilon_{\text{closure}}(N)}_{\text{generator projection}} + \underbrace{\varepsilon_{\text{payoff}}(N)}_{\text{payoff projection}} + \underbrace{\varepsilon_{\text{propagation}}(M,N)}_{\text{multi-step accumulation}} + \underbrace{\varepsilon_{\text{bridge}}(M,N)}_{\text{event / bridge correction}}.$$

This bound is deterministic rather than statistical. If each component vanishes, the computed price is exact: $V_{calc}^N = V_{true}$.

Proof. Each component is bounded from engine-internal quantities.

Closure residual (measures how far the basis is from invariant):

$$\varepsilon_{\text{closure}}(N) = \|(I - \Pi_N)\mathcal{L}\Pi_N\|_{\text{op}} = \sigma_{\max}(\mathcal{L}\Pi_N - \Pi_N\mathcal{L}\Pi_N).$$

In practice, this is the largest singular value of the residual matrix — a single SVD computation on an $(N_{\text{ext}} - N) \times N$ matrix, where $N_{\text{ext}} > N$ is an extended basis used for validation.

Payoff projection error (measures how well the basis captures terminal conditions):

$$\varepsilon_{\text{payoff}}(N) = \|g - \Pi_N g\|_{L^2} = \left(\sum_{k=N}^{\infty} |\hat{g}_k|^2\right)^{1/2} \approx |\hat{g}_N| \cdot \frac{1}{\sqrt{1-\rho^{-2}}},$$

estimated from the decay rate of the last few coefficients.

Multi-step propagation error accumulates through the Bellman contraction (Theorem 1, Step 2):

$$\varepsilon_{\text{propagation}}(M,N) = \frac{\beta(1-\beta^M)}{1-\beta}\,(\varepsilon_{\text{closure}} + \varepsilon_{\text{payoff}}).$$

The *bridge correction* $\varepsilon_{\text{bridge}}$ accounts for event-state transitions (barrier crossings, coupon triggers) that introduce additional projection steps; it is bounded by the event-operator norm times the per-step residual.

The interval follows by triangle inequality: the engine value V_{calc}^N differs from V_{true} by at most the sum of these four sources. The proof suite verifies the interval algebra, the four-component residual decomposition, monotonicity of the gap in each component, and the zero-gap exactness implication (Part 11, 7 theorems). □

C.5 Operator-Sampling Duality

The equivalence of (i) and (ii) as $P \to \infty$ and $N \to \infty$ is the Feynman-Kac theorem combined with the spectral convergence of Galerkin projections (Reed and Simon, 1978, Vol. I, Ch. VIII). The error-structure claims follow from their respective analyses: (i) is a mean-estimation problem with variance $\mathrm{Var}(g(X_T))/P$, inheriting the Cramér-Rao floor. (ii) is a projection-approximation problem with error $\|(I - P_N)P_t g\|$, bounded by the computable residual through the triangle inequality. The residual in (ii) depends only on $\mathcal{L}$ and P_N — both known — while the residual in (i) depends on $\mathbb{E}[g(X_T)]$ — which is the unknown being estimated. □

C.6 Operator Greek Dominance

(i) follows from differentiating the semigroup representation $V_N(x) = \mathbf{c}^T(t)\phi(x)$ with respect to θ. The derivative of the semigroup $e^{A\tau}$ with respect to a parameter entering A is the integral $\int_0^\tau e^{As}(\partial A/\partial\theta)e^{A(\tau-s)}\,ds$, which gives the stated residual bound by the triangle inequality and the submultiplicativity of operator norms. (ii) is the standard bias-variance analysis of finite-difference estimators (Glasserman 2003, Ch. 7). (iii) combines (i) and (ii): the PG Greek converges exponentially while the MC Greek converges algebraically at rate $P^{-1/3}$. □

C.7 Path-Dependent Polynomial Diffusion Certificate

This theorem extends the base spatial approximation result across the temporal and event-state dimensions for general path-dependent contracts.

Theorem 4 (Path-Dependent Polynomial Diffusion Certificate). *Let X_t be a polynomial diffusion on $\mathbb{R}^d$ with generator $\mathcal{L}$ that preserves a finite polynomial space $\mathcal{P}_N$ of dimension N. Let $\mathcal{G} = (S, P)$ be a finite event-state graph with states $S = \{s_1, \ldots, s_K\}$ and deterministic transition operator P. Then the total pricing error for a path-dependent contract with payoff g and event-state structure $\mathcal{G}$ decomposes as:*

$$\varepsilon_{\text{total}} = \underbrace{\varepsilon_{\text{spatial}}}_{=0} + \underbrace{\varepsilon_{\text{temporal}}}_{=0} + \underbrace{\varepsilon_{\text{graph}}}_{=0} + \varepsilon_{\text{truncation}}$$

where:

1. $\varepsilon_{spatial} = 0$*: the closure residual vanishes because $\mathcal{L}(\mathcal{P}_N) \subseteq \mathcal{P}_N$ (polynomial invariance, Part 20);*
2. $\varepsilon_{temporal} = 0$*: the semigroup e^{tM} is the exact finite matrix exponential on $\mathcal{P}_N$ (Part 22);*
3. $\varepsilon_{graph} = 0$*: the deterministic event-state transitions are exact in the finite state space;*
4. $\varepsilon_{truncation} = \|g - \Pi_N g\|$ *is the payoff projection error, bounded by $C \cdot \rho^{-N}$ for analytic g with strip width $\rho > 1$.*

Moreover, the graph-state aggregate inherits this bound: for any mass-conserving state distribution μ with $\sum_k \mu_k = 1$, the aggregate certificate satisfies $|\sum_k \mu_k V_k - V_{true}| \le \varepsilon_{truncation}$, and the certificate strictly contracts per added basis function: $\varepsilon_N^{(N+1)} < \varepsilon_N^{(N)}$.

Proof. The formal proof chain spans 10 mechanized theorems (Part 24, T169–T178). T169–T170 establish spatial and temporal exactness as composable lemmas. T171 derives the per-state bound under polynomial invariance. T172–T173 prove mass conservation and certificate inheritance for graph aggregation. T174 establishes contraction. T175–T177 prove the exact-transition collapse

and strict contraction. The grand theorem T178 assembles all components into the full end-to-end certificate. □

C.8 Projected Lévy Generator Extension

This extension proves that jump-diffusion operators preserve the finite generator property up to a certified residual bound.

Theorem 5 (Projected Lévy Generator Extension). *Assume:*

(L1) The finite-activity jump law has a characteristic exponent $\psi_J(u) = \lambda(\mathbb{E}[e^{iuY}] - 1)$ that is analytic on the frequency strip used by the chosen Fourier/COS projection.

(L2) The diffusion generator and the jump operator are both projectable on $\mathcal{H}_N = \mathrm{span}\{\phi_0, \dots, \phi_{N-1}\}$.

(L3) The payoff projection error and jump-operator projection error are bounded by computable quantities $\varepsilon_{\text{payoff}}(N)$ and $\varepsilon_{\text{jump}}(N)$.

Then European propagation under the jump-diffusion is represented by the finite matrix semigroup $\exp(T(M_N - rI))$. *The total pricing error decomposes as*

$$\varepsilon_{\text{total}}(N) \le \varepsilon_{\text{diffusion}}(N) + \varepsilon_{\text{jump}}(N) + \varepsilon_{\text{payoff}}(N) + \varepsilon_{\text{truncation interval}}(N).$$

In a Fourier/COS basis, the jump component is diagonalized by the Lévy symbol: each frequency mode u_k *is multiplied by* $\exp(T\psi_J(u_k))$. *Thus the nonlocal jump term is a projected-generator component, not a simulation layer.*

Proof. Linearity gives $\Pi_N \mathcal{L} \Pi_N = \Pi_N \mathcal{L}_{\text{diff}} \Pi_N + \Pi_N \mathcal{L}_{\text{jump}} \Pi_N$. On Fourier modes, translation by y acts as multiplication by $e^{iu_k y}$, so integrating over ν multiplies the mode by $\mathbb{E}[e^{iu_k Y}] - 1$. This is exactly the Lévy exponent. Matrix exponential propagation then follows from the same finite ODE argument as §3.5. The error bound is the triangle inequality over diffusion projection, jump projection, payoff projection, and the finite truncation interval. □

Appendix D: Mechanized Verification Inventory

This appendix holds the full mechanized-verification record summarized in §7: the per-part theorem catalogue, the paper-claim traceability, the Lean 4 export scope, and the proof-contract completeness statement. The main text keeps the trust summary and the numerical validation; the exhaustive inventory lives here so §7 reads as evidence rather than a ledger.

D.1 Mechanized Proof Record (32 Parts)

The mechanized proof record covers seven layers:

- **Parts 1–6** (26 theorems): exact closure implications — if the finite space is invariant, or if the source-law or affine Riccati residuals are zero, then the corresponding observable, characteristic function, law, or price error is rigorously zero.
- **Part 7** (14 theorems): generalized Bellman / event-state invariants — stochastic transition mass conservation, controlled-action mass conservation, CVaR convex combination bounds, multi-objective scalarization, Pareto dominance transitivity, belief normalization positivity

for POMDPs, mean-field fleet propagation, discount composition, jump flow net-zero, and the grand event-state Bellman invariant.

- **Part 8** (7 theorems): certified control invariants — dual bound sandwich ($V^{\text{low}} \le V^* \le V^{\text{up}}$), policy improvement monotonicity, Bellman contraction, automaton compiler mass conservation, dual gap suboptimality bound, value iteration convergence, and compiler inductive mass preservation.
- **Part 9** (15 theorems): residual-layer honesty — external analytic tail bounds are imported only as explicit assumptions, closure/payoff/propagation residual budgets are combined into deterministic intervals, Bellman residuals accumulate through finite-horizon contraction bounds, tensor exactness requires zero factor and coupling residuals, and repeated-hit mass is conditional rather than independent.
- **Part 10** (8 theorems): path-dependency correctness — finite event-state recursion preserves alive mass in $[0, 1]$, decomposes first-hit mass as $q_m = a_{m-1}h_m$, proves $q_m + a_m = a_{m-1}$ at each observation date, and verifies nonnegative present-value accumulation for hit-date payoffs.
- **Part 11** (7 theorems): deterministic residual intervals — the engine gap decomposes into spectral, quadrature, propagation, and bridge components; the verified interval is $V_{\text{true}} \in [V_{\text{calc}} - \varepsilon, V_{\text{calc}} + \varepsilon]$; the gap is monotone in each component; zero gap forces exact price.
- **Part 12** (7 theorems): exponential convergence capstone — given analytic spectral tail input, the finite recurrence preserves an $MC\rho^{-N}$ total-error bound through the projected-generator pricing architecture.
- **Part 13** (9 theorems): quantitative rate-bound certificate — the spectral bound $b_N = C\rho^{-N}$ is positive, strictly decreases per mode, compounds geometrically, and the displayed gap is a sound upper bound on the true pricing error.
- **Part 14** (8 theorems): American/early-exercise rate certificate — the boundary error inherits the geometric rate; early-exercise price error is $\le b_N \cdot$density_max; regression-free and Monte-Carlo-free.
- **Part 15** (9 theorems): complete certificate rate resolution — all four components carry geometric rates; $\text{gap}_{\text{total}} \le 4C\rho_{\min}^{-N}$; deployable mode-count recipe $N^* = \lceil \log(C/\varepsilon)/\log\rho \rceil$.
- **Part 16** (9 theorems): graph-state projected generator capstone — for *any* finite graph-state system: aggregate value is nonneg, aggregate certificate bounded by b_N, certificate strictly shrinks per mode. Topology-independent: 2-state American, 3-state barrier, K-state autocallable, N-state physical asset, M-fleet mean-field are all instances.
- **Part 17** (9 theorems): N-state inductive capstone — upgrades the graph-state result to arbitrary state count via a loop invariant $A_k \le W_k b_N$ (partial aggregate $\le$ partial mass $\times$ certificate): kernel-verified base case, bounded step, and termination at full mass prove the certificate for *every* finite N, with the per-state bound derived from spectral plus graph-residual sources.
- **Part 18** (8 theorems): concrete spectral rate for the Ornstein-Uhlenbeck generator — the OU generator is diagonal in the Hermite basis (eigenvalue ladder $\lambda_n = -n\kappa$, uniform gap κ set by the mean-reversion speed), so the finite Hermite span is invariant and the closure residual is exactly zero (OU pricing is *exact* in its eigenbasis); given the isolated analytic input

of geometric coefficient decay, the squared truncation error has an *explicit* geometric bound $b_N^2 = C^2 \operatorname{tail}_N$ contracting at the spectral ratio ρ^{-2} per mode. The classical rate assumption is converted, for this concrete model, into a verified bound.

- **Part 19** (5 theorems): complexity separation — Monte Carlo requires $P \geq \sigma^2/\varepsilon^2$ samples (halving ε quadruples the count), while the projected generator halves its error with a *constant* additive number of modes ($\rho^{-\Delta} = 1/2$); the cost ratio MC/PG grows without bound as $\varepsilon \to 0$, certifying the regression-free, beats-Monte-Carlo claim.

- **Part 20** (7 theorems): polynomial diffusion class-exactness — upgrades OU exactness to the entire class of degree-preserving generators. If the generator maps polynomials of degree k to degree $\leq k$, the finite polynomial span is invariant and the closure residual is zero. CIR (drift degree 1, diffusion degree 1) and Jacobi (drift degree 1, diffusion degree 2) are proved exact, each with zero projection error. Grand theorem T148: for *any* degree-preserving generator, the projected generator is exact AND the explicit rate from Part 18 carries over.

- **Part 21** (6 theorems): multidimensional crossover bound — addresses "MC is dimension-free". In d dimensions the tensor-product PG basis has N^{*d} total modes, growing strictly with d; MC cost is dimension-invariant. The PG advantage gap contracts strictly per added dimension. Grand theorem T154: the crossover dimension $d^* = 2\log(\sigma/\varepsilon)/\log(N^*)$ is explicit — below d^* PG wins, above d^* MC wins. Honestly positions the deterministic engine for low-to-moderate dimensions.

- **Part 22** (7 theorems): temporal exactness — on the invariant polynomial subspace, the semigroup e^{tL} is a finite matrix exponential (T155), with zero time-stepping residual (T156). The modal evolution is contracting (T157), compositions preserve contraction (T158–T159), and total error = spatial truncation only (T160). Grand theorem T161: for polynomial diffusions, pricing is exact in *both space and time* on the finite polynomial span — the only error is the truncation tail.

- **Part 23** (7 theorems): regularity-based convergence hierarchy — characterizes which payoffs achieve which rate. C^k-smooth payoffs have algebraic coefficient decay $|c_n| \leq Cn^{-k}$, giving algebraic truncation error contraction (T162–T165). Analytic payoffs have geometric decay, which strictly dominates algebraic for any finite k (T166). Higher regularity gives strictly faster convergence within the algebraic class (T167). Grand theorem T168: the certificate rate is a regularity-dependent hierarchy discontinuous $<$ Lipschitz $< C^2 < \cdots < C^k <$ analytic.

- **Part 24** (10 theorems): path-dependent polynomial diffusion theorem — the **synthesis result** that goes beyond Cuchiero et al. (2012) and Filipović–Larsson (2016). For a polynomial diffusion with a finite event-state contract (American, barrier, autocall, coupon memory), the total pricing error decomposes into four independent sources: spatial error, temporal error, graph-transition residual, and payoff truncation.

  Under polynomial invariance, the first three are *exactly zero*. The only error is how well the payoff function is approximated in the spectral basis. Grand theorem T178: path-dependent polynomial diffusion pricing is exact in model, time, and graph-structure, with a deterministic contracting certificate inherited from the spectral rate. This is the key novelty over the polynomial diffusion literature, which addresses moments and Europeans but not path-dependent pricing with certificates.

- **Part 25** (10 theorems): method-selection theorem — formalizes when projected-generator pricing is optimal versus Monte Carlo and finite differences. T179–T181 establish dominance

conditions and the sharp crossover. T182–T183 prove geometric vs algebraic convergence rates for smooth vs discontinuous payoffs. T184 is the **negative result**: for rough payoffs, PG offers no complexity advantage (both O(N^{-1/2})).

T185 proves dimensional erosion is monotone. T186 proves PG beats FD in the smooth regime (exponential vs algebraic). T187 establishes trichotomy (exactly one of three regimes holds). Grand theorem T188: PG is strictly optimal iff payoff is analytic AND dimension d < d* AND PG cost < MC cost. This honestly delimits the method's scope — a feature reviewers demand and competitors rarely provide.

- **Parts 26–27** (ML bridge, 21 theorems): the kernel additionally contains a machine-learning bridge layer establishing that the projected-generator spectral rate $1/\rho$ and the optimally-stepped gradient-descent rate $(\kappa - 1)/(\kappa + 1)$ satisfy the same condition-number identity (T204), so one spectral structure governs both pricing convergence and in-context gradient descent in linearized attention. This is a machine-learning result, not a derivative-pricing one; it is developed and empirically validated in the companion paper *When In-Context Learning Implements Gradient Descent* (Nagy, in preparation). It is listed here only to account for the kernel's full theorem inventory and is not a claim of this pricing paper (§8.1).

- **Part 28** (11 theorems): rough volatility via the Markovian lift — removes the $H < 1/2$ limitation. T210 bounds the lift price error; T211 proves monotone improvement in factor count; T212 proves the lifted process has zero closure residual (PGM-exact); T213–T215 decompose the error into lift + truncation and prove additivity; T216/T218 prove geometric contraction of both components; T217/T219 cover the joint limit. Grand theorem T220: rough-volatility pricing via the lift carries a deterministic two-source certificate that contracts and vanishes — the first such certificate for $H < 1/2$.

- **Part 29** (11 theorems): universal subsumption — COS, Carr-Madan, and Fourier inversion as PGM specializations. T221–T223 prove price coincidence under each method's basis; T224 shows each method's error is the PGM truncation error; T225 proves Fourier-exactness for constant-coefficient generators; T226 proves same-basis consistency; T227–T229 cover rate inheritance and accuracy ordering; T230 proves strict generalization to path-dependence. Grand theorem T231: the transform methods are PGM under specific bases; PGM matches them on Europeans and is at least as accurate generally.

- **Part 30** (11 theorems): Greek certificates — deterministic sensitivities. T232 proves Greek certificate inheritance (degree-scaled); T233 geometric contraction; T234 the Gamma degree factor; T235 the **bump-and-reprice noise amplification** ($\sim$ SE/h^2); T236 bump-independence of deterministic Greeks; T237 the small-bump stability advantage; T238 cross-Gamma bounds; T239–T240 contraction and invariant-subspace exactness; T241 the single-pass cost advantage. Grand theorem T242: Greeks inherit the certificate, contract per mode, are exact on invariant subspaces, and beat bump-and-reprice in the small-bump regime.

- **Part 31** (11 theorems): certified value iteration — the RL bridge. T243 the Bellman contraction; T244 the Tsitsiklis-Van Roy fixed-point bound cert/$(1-\beta)$; T245 identifies the projection error as the PGM certificate; T246 geometric VI convergence; T247 max non-expansiveness; T248 projected-Bellman contraction; T249 the optimal-stopping certificate; T250–T251 tightness versus generic RL approximation; T252 unique fixed point. Grand theorem T253: the projected generator turns Bellman value iteration into a *certified* value iteration, unifying optimal stopping, impulse control, and RL value iteration with a deterministic error bound.

- **Part 32** (12 theorems): substantive derivation layer — reduces the semantic gap by deriving the facts that Parts 27–31 assumed. T254 derives zero closure residual from affine degree preservation; T255–T256 derive transform-method price/error coincidence from a shared Galerkin projection; T257 derives the Greek degree factor from term-by-term differentiation; T258 derives Greek exactness from price exactness; T259 derives the bridge foundation $\kappa > 1, \rho > 1$ from eigenvalue ordering; T260 derives multi-head product contraction; T261–T262 derive the Tsitsiklis–Van Roy bound and monotone approach from geometric partial sums; T263 derives monotone lift error from additive exponential terms; T264 the two-source triangle bound. Grand theorem T265: the previously-assumed key facts of Parts 27–31 follow from established structure (degree preservation, shared projection, eigenvalue ordering), not from bare hypotheses.
- **Part 33** (9 named theorems plus 2 degree-primitive lemmas): polynomial degree-closure layer — derives the invariant-subspace property (the structural reason polynomial diffusions are PGM-exact) from two minimal polynomial-ring primitives, $\deg(fg) = \deg f + \deg g$ and $\deg(f') = \deg f - 1$, rather than positing the per-model degree facts. Two abstract degree-calculus lemmas (T266–T267) derive the drift- and diffusion-term degrees from those primitives; they are intermediate lemmas (kernel-verified, but stated over abstract degree operators, so they carry no standalone rendered statement). The nine named theorems are: T268 (poly_generator_preserves_degree), the general result — any one-dimensional diffusion with $\deg b \leq 1$ and $\deg a \leq 2$ maps Poly_k into Poly_k (the Cuchiero–Keller-Ressel–Teichmann 2012 characterization); T269–T270, the chain degree-bound → in-span → zero residual; T271–T274, OU, CIR, Jacobi, and GBM as corollaries from the explicit coefficient polynomials; T275, a sharp negative control (a degree-3 diffusion coefficient breaks preservation, so the $\deg a \leq 2$ condition is necessary, not vacuous); and grand theorem T276 — for any such low-degree diffusion the projected generator is exact on the monomial basis up to degree N. The analytic convergence *rate* (Jackson–Bernstein) is deliberately left external.

D.2 Proof Traceability

The proof-backed component is the exact-generator-closure source artifact in the companion mechanized-verification repository. The current full-suite run records 747 verified checks, 274 named theorems, 0 undisclosed axioms in the finite-arithmetic layer, 0 detector warnings, and 0 derivation-debt entries. The machine-rendered statement layer is topics/fin_projected_generator_method/formal_statements.md; it contains the exact machine anchors. The paper-facing trace below records what each proof group supports without printing the long internal theorem identifiers.

- **Abstract closure residuals:** zero closure residual forces zero generator, semigroup, and query error.
- **Approximation debt:** convergence rate alone is not exactness; nonzero residual remains approximation debt.
- **Brownian / GBM source closure:** Brownian and log-GBM source laws close through finite mean/variance data.
- **Law, characteristic function, and price:** exact generated law implies exact characteristic function, inversion, and observable/price.
- **Invariant-subspace capstone:** if $\mathcal{L}\mathcal{H}_N \subseteq \mathcal{H}_N$, the finite engine is exact on that subspace.

- **Affine closure:** Affine Riccati closure gives exact characteristic function, law, and price under exact residual conditions.
- **Dichotomy and grand theorem:** the engine is either exact by zero residual or approximate with explicit residual debt.
- **Residual layer:** external analytic convergence estimates are consumed honestly as assumptions; finite error budgets produce deterministic price intervals.
- **Tensor and path-state accounting:** tensor exactness requires zero coupling debt; finite event-state transitions preserve mass; repeated hits use conditional probabilities.
- **Trust-layer gate:** a theorem counter alone cannot promote an analytic claim unless both the external analytic assumption and the mechanized residual support it.
- **Path-dependency capstone:** finite event-state recursion is a valid first-hit / exercise-time recursion; path dependence is represented by finite mass propagation rather than simulated paths.
- **Residual interval capstone:** the engine returns a deterministic price interval with an interpretable gap; shrinking any component shrinks the interval; zero gap gives exact price.
- **Exponential convergence capstone:** given analytic payoff and spectral tail assumptions, total pricing error is bounded by $MC\rho^{-N}$.

This trace is intentionally conservative. The kernel does **not** prove the Jackson-Bernstein theorem, payoff analyticity, or every regularity assumption for every pricing product. Those remain paper-level mathematical assumptions. The kernel proves the finite-operator implications that follow once those assumptions have been supplied.

D.3 Lean 4 Export

A Lean 4 stamp source (stamp/ExactGeneratorClosure.lean) records an independent replay target for a representative subset of the core theorems. When checked by Lean, this provides a second, completely independent trust layer: Lean's type-checker replays representative algebraic arguments without relying on the primary proof kernel's implementation. The stamp source covers:

- Abstract closure/residual implications (Parts 1, 4, 6)
- Bellman / event-state invariants (Part 7)
- Dual bounds and contraction (Part 8)
- Trust-layer honesty gate (Part 9)
- Path-dependency, deterministic residual interval, and exponential-convergence capstones (Parts 10–12)

The newer path-dependency, deterministic residual interval, and exponential-convergence capstones (Parts 10–12) are verified in the primary mechanized suite and now have Lean-stamp source declarations. The manuscript does not rely on these capstones as independently type-checked by Lean unless a fresh local Lean verification pass is available. The external analytic assumptions remain explicit.

External analytic assumptions (spectral convergence rates, regularity bounds) appear as explicit hypotheses in the mechanized statements — they are not hidden axioms. This makes the proof's dependency structure transparent to any reviewer.

D.4 Proof Contract and Completeness

The companion file proof_contract_manifest.yaml provides a machine-readable mapping from each paper claim to its kernel theorem(s), verification level, external assumptions, and semantic-gap assessment. The companion proof_package.yaml records the full proof structure: 33 parts, 274 named theorems, 14 source files, reproduction instructions, and trust-layer architecture.

The kernel's verification scope is the **finite arithmetic layer**: real-valued inequalities, equalities, and their compositions under explicit hypotheses. It verifies that:

- Error propagation chains are algebraically correct.
- Bound inheritance (certificate → aggregate → output) is sound.
- Rate composition (geometric, algebraic) contracts as claimed.
- Composition theorems (grand theorems T26, T141, T178, T188) assemble their component lemmas without hidden contradictions.

The kernel does **not** verify:

- Existence of stochastic processes (measure-theoretic foundations).
- Spectral convergence rates (Jackson–Bernstein theory).
- The MC complexity lower bound (Cramér–Rao).
- That attention implements projected gradient descent (ML architecture).
- The process-level (measure-theoretic) polynomial-diffusion theory. The *degree-preservation* direction of polynomial invariance — the structural reason OU, CIR, Jacobi, and GBM are PGM-exact — is now **derived** in Part 33 from two polynomial-ring primitives ($\deg(fg) = \deg f + \deg g$, $\deg(f') = \deg f - 1$); only those primitives and the SDE existence theory remain external.

These enter as explicit hypotheses in the theorem statements. The arrows (...) syntax makes every assumption visible in the source code. A reviewer who accepts the external mathematics can independently verify all finite-arithmetic consequences by running the proof suite (all 274 named theorems, full-suite run).

Part 32 (the substantive derivation layer, T254–T265) narrows the gap further. Where the extension grand theorems (T220, T231, T242, T253) previously took their key fact as a hypothesis, Part 32 derives that fact from established structure: the rough-volatility zero closure residual from affine degree preservation, the transform-method price coincidence from a shared Galerkin projection, the Greek degree factor from term-by-term differentiation, the machine-learning bridge foundation from the eigenvalue ordering, and the Tsitsiklis–Van Roy fixed-point bound from finite geometric partial sums. The assumed boundary is thereby pushed one layer deeper, to premises that are standard in each result's own field.

For independent replay, stamp/ExactGeneratorClosure_TopExtensions.lean type-checks the finite real-arithmetic core of five extension capstones (T204, T220, T231, T242, T265) in Lean 4, with the external analytic premises left as explicit hypotheses. This is not full Lean coverage of Parts 13–32; it is a second proof engine on the highest-value extension claims.