FOUNDATION MODELS FOR VARIANCE REDUCTION IN STOCHASTIC PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Estimating expectations of functionals of stochastic processes is a fundamental problem in many disciplines, from finance to life sciences and engineering. However, obtaining accurate estimates with Monte Carlo requires prohibitively many samples. We introduce foundation models for variance reduction that use simulator paths and black-box functional evaluations, without access to the explicit dynamics or task parameters. A pretrained model reads a small set of observed paths and outputs a task-specific variance-reduction controller, an importance sampling proposal or control variates integrand. This architecture leads to estimators which are unbiased by construction and adds little computational overhead. Because the models separate identification from estimation, the allocation of paths between the two provides a natural test-time scaling mechanism. We pretrain on a broad family of synthetic stochastic processes, to ensure strong downstream performance even in out-of-distribution applications, especially for importance sampling. We demonstrate the performance of the models on a variety of applications in finance and biology, showing that the proposed model provides significant variance reduction with matched wall-clock speed.

1 INTRODUCTION

Foundation models have introduced a new paradigm for algorithm design: rather than developing a dedicated method for each task, a single model is pretrained on a broad distribution of tasks and adapts to new ones in-context, without parameter updates. Beyond natural language, this paradigm has proven successful for tabular prediction (Müller et al., 2022; Hollmann et al., 2023), time-series forecasting (Ansari et al., 2024; Das et al., 2024), and vision (Oquab et al., 2024). This is largely possible due to the cheap availability of training tasks that can be generated at low cost, often synthetically. Stochastic simulation, where existing simulators can generate pretraining data cheaply, is therefore a natural scenario to explore, yet it has received comparatively little attention.

A central problem in stochastic simulation is the estimation of expectations $\mathbb{E}[F(X)]$ of functionals of a stochastic process X, which arises throughout finance, the life sciences, and engineering. Standard Monte Carlo estimators are unbiased and easy to parallelize, with known error rates decaying as $N^{-1/2}$ in the number of paths N. When the process variance is large relative to the target accuracy, however, the number of paths required to achieve this accuracy can quickly exceed available computational resources. To counteract this, variance reduction techniques, such as importance sampling and control variates, have been introduced to maintain unbiased estimations while reducing variance. While these are powerful techniques that can provide substantial variance reduction, they are problem-specific and often require derivations based on both the dynamics of the process and the functional. Learned approaches fit a neural control variate or importance-sampling drift to each task and achieve substantial speedups, for instance in option pricing (Hinds & Tretyakov, 2023; Arandjelović et al., 2025). Unfortunately, these require a separate training procedure for new tasks, which incurs a high computational cost that is only worthwhile when the same task is estimated repeatedly. Amortized methods mitigate this cost but require either the parameters of the task as explicit inputs (Golinski et al., 2019; Sabate Vidales et al., 2021) or gradient-based adaptation at test time (Sun et al., 2023). These limitations make variance reduction a potential use case of large, pretrained models (referred hereafter as foundation models). Pretraining data can be generated at scale by simulating families of processes and functionals of our choosing. Moreover, a small number of simulated paths (which the model uses as context) contains much of the information required

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

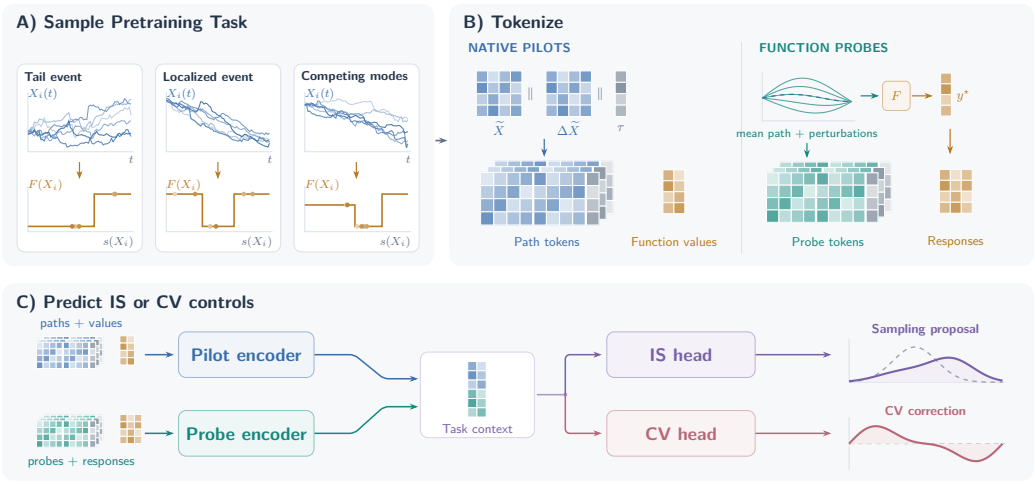


Figure 1: Overarching schematic of our architecture. **(A)** describes the pretraining vocabulary; **(B)** illustrates the tokenization; **(C)** describes the network architectures under the importance sampling (IS) and control variate (CV) variants. Both variants use the same underlying architecture but predict different quantities.

to construct an effective scheme, such as the correlation between the functional and the path, and the important regions of path space that dominate the variance. The pretrained prior compensates for the information these paths do not provide. A pretrained model can thus take as input a small set of pilot paths from a black-box simulator and output a task-specific variance-reduction scheme, without retraining and without access to the parameters of the process. In addition, the allocation of the simulation budget between pilot paths used to identify the scheme, and estimation paths used to apply it, provides a natural test-time scaling mechanism. Such a model must nonetheless retain the properties that make Monte Carlo methods attractive: the estimator must remain unbiased irrespective of the quality of the model, improve with the number of estimations paths, and incur negligible overhead per path. Otherwise, allocating the same computational budget to additional simulations would be preferable.

This leads to the two questions addressed in this work:

1. Which pretraining mix of stochastic processes and functionals, constructed strictly from simulator realizations, enables a single model to transfer to unseen tasks?
2. Which architectures provide variance reduction, while incurring small computational cost?

We emphasize that the latency requirements for such foundation models are considerably more stringent than in tabular prediction or computer vision. The pretrained model is useful only if the variance reduction it provides outweighs the cost of its evaluation, which could otherwise be spent on simulating additional paths. This induces a trade-off between model capacity and inference cost: the architecture must therefore be designed to balance expressivity against speed, achieving substantial variance reduction while adding only marginal cost per path.

Our contributions are as follows:

1. We introduce efficient architectures for foundation models for variance reduction that are unbiased by construction (§ 3) and provide benefits over Monte Carlo at matched computational budget (§ 5).
2. We train these architectures on a diverse synthetic pretraining distribution of stochastic processes (§ 4), and observe scaling with data and model size as well as test-time scaling in the number of pilot and estimation paths (§§ 4 and 5).
3. We evaluate the resulting models on a range of downstream tasks, from estimation of expectations to stochastic optimization (§ 5), including zero-shot generalization to real world for importance sampling and finetuning to further specialize the pretrained model to a target task (§ 5.3).

1.1 Related Work

Learned variance reduction. Given a specific problem, learned variance reduction methods train a control variate or an importance-sampling proposal to reduce the variance of Monte Carlo estimators. Neural networks have naturally emerged as flexible function approximators for these tasks (Grathwohl et al., 2017; Wan et al., 2018; Müller et al., 2019; 2020). In particular, Hua et al. (2023) trained a neural network to approximate the optimal drift for importance sampling for rare-event simulations for molecular dynamics. Closer to our applications, Hinds & Tretyakov (2023) and Arandjelović et al. (2025) trained respectively task-specific neural control variates and neural Girsanov drifts for importance sampling for various option pricing problems, showing significant speedups over naive Monte Carlo. Li et al. (2024) leverage predictions from a cheap machine learning model to learn better control variates for Monte Carlo integration. However, these methods are task-specific and require retraining for each new problem, while our approach consists in training large foundation models that can be reused across a wide range of problems without retraining.

Transferable variance reduction Closer to our approach, some existing lines of work have explored general-purpose variance reduction methods that can be reused across multiple tasks. Golinski et al. (2019) learn importance sampling proposals for a parametric family of integrals, amortizing the cost of learning across tasks from this family. In contrast to us, the model requires explicit knowledge of the task parameters as input. Similarly, Sabate Vidales et al. (2021) focuses on a parametric class of stochastic processes relevant to finance applications. However, their solution requires explicit knowledge of the parameters as input at test time. Sun et al. (2023) trains a control variate model so that after a few gradient steps on a new task, the model performs well on that task. Finally Siahkoohi & Oh (2026) amortize neural control variates across observations, but require posterior-score information.

Pretrained models for tabular data, time series, and stochastic processes. Foundation models emerged as a new paradigm with pretraining on diverse tasks and using context without parameter updates at test time. Müller et al. (2022) developed this idea for Bayesian inference while Hollmann et al. (2023); Qu et al. (2025) specialized it to tabular classification. Foundation models have also been successfully applied to time series forecasting (Ansari et al., 2024; Das et al., 2024) as well as drift and covariance inference for stochastic processes Seifner et al. (2025). Our work builds on these ideas to extend foundation models to variance reduction for Monte Carlo estimation.

2 Setting

2.1 Stochastic processes

In this work we consider general vector-valued stochastic processes $X_{1:T} \in \mathbb{R}^{T\times d}$, where T is the number of time steps and d is the dimension of the process. We are interested in estimating the expectation of a functional F of the stochastic process, denoted as $\mathbb{E}F(X)$. The main classes of stochastic processes we will consider in this work are discretized diffusions and jump-diffusions, which are the most common stochastic processes in finance, physics, and biology. A diffusion process satisfies a stochastic differential equation (SDE) of the form

$$\mathrm{d}X_t = a(t, X_t)\ \mathrm{d}t + \sigma(t, X_t)\ \mathrm{d}W_t, \quad t \in [0, H], \tag{1}$$

where W is a standard Brownian motion, a and σ are the drift and diffusion coefficients, and H is the horizon. The quantity of interest is $\mathbb{E}_\theta F(X)$ for a payoff functional F of the path. Here are two common examples of interest.

Examples. Diffusion processes are the most common in finance. A prime example is the **Black-Scholes model:** taking $a(t, x) = \mu x$ and $\sigma_\theta(t, x) = \sigma x$ recovers the Black–Scholes model for an asset price X_t with drift μ and volatility σ. In this case, one is interested in estimating the payoff of an option $F(X)$, which is a functional of the path X, for instance the European call $F(X) = (X_H - K)_+$. In computational physics, chemical reactions can be modeled as particles following the **Langevin dynamics** governed by a potential functions U, i.e., $a(t, x) = -\nabla U(x)$ (Schlick, 2010). The potential U is often multimodal, each minimum corresponding to a metastable state and transitions between these states are rare events. In this case, one seeks to estimate payoffs that depend on these rare events, such as the probability of transition between two metastable states.

2.2 VARIANCE REDUCTION

Our foundation models are designed around two classical variance-reduction techniques: importance sampling and control variates. These algorithms are widely used in Monte Carlo estimation to reduce the variance of the estimator while preserving unbiasedness.

Denoting by $\mathbb{P}$ the original law of the stochastic process X, importance sampling consists in simulating X under a different law $\mathbb{Q}$ and reweighting the samples by the likelihood ratio to obtain an unbiased estimator of the expectation of interest:

$$\mathbb{E}_\theta F(X) = \mathbb{E}_{\mathbb{Q}}\big[F(X)L\big], \qquad L = \frac{\mathrm{d}\mathbb{P}_\theta}{\mathrm{d}\mathbb{Q}}. \tag{2}$$

In our case, we will consider $\mathbb{Q}$ to be the law under which X solves the same SDE (1) with a changed noise distribution . The likelihood ratio L can then be computed using Girsanov's theorem. Simulating under $\mathbb{Q}$ and averaging $F(X)L$ over independent paths gives an estimator that is unbiased by construction; only the variance depends on the changed distribution, and a well-chosen one can decrease the variance of the estimator significantly.

Control variates, particularly linear control variates, consist in subtracting from $F(X)$ a mean-zero correction built from the driving Brownian motion: for a predictable integrand h_t, the integral $M = \int_0^H h_t^\top \,\mathrm{d}W_t$ satisfies $\mathbb{E}M = 0$, so $F(X) - M$ is an unbiased estimator of $\mathbb{E}F(X)$. The variance of the estimator depends on h though, and a well-chosen h can decrease the variance of the estimator significantly. The martingale representation theorem, see e.g. (Øksendal, 2003, Chap. 4), motivates this class of controls: if $F(X)$ is W-measurable, there exists a predictable process h^* such that

$$F(X) - \mathbb{E}F(X) = \int_0^H (h_t^*)^\top \,\mathrm{d}W_t.$$

Thus in general, linear control variates can be used to reduce the variance of the estimator arbitrarily: this motivates our approach for control variates.

3 METHOD

3.1 PROPOSED APPROACH

Given a set of paths at inference time, we want to produce an unbiased estimator of the expectation of the function $\mathbb{E}F(X)$ with low variance by predicting either an importance-sampling drift or a linear control-variate integrand. For this, we leverage in-context learning. A large model is pretrained on a large number of synthetic stochastic processes, each with its own functional.

At inference time, the model only has black-box access to a simulator and to F. For importance sampling, we assume that we can inject driving noise into the process; for control variates, that we can access the noise increments. The simulated paths are split into two sets. A small set of pilot paths, together with their payoffs, is used to identify the task and produce a controller b or h. A larger set of independent estimation paths is then used to compute the estimator with this controller. This approach has several key benefits:

1. **Unbiased by construction.** The estimator is unbiased for any controller b or h by § 2.2 since pilot and estimation paths are kept separate.
2. **No access to task parameters.** The model only ever sees raw path states, driving noise increments and functional values, not the parameters of the process or the function itself; nothing in the architecture requires the processes seen at pretraining and at inference to share a functional form.
3. **Efficient.** The model achieves variance reduction with little computational overhead, beating Monte-Carlo at matched-time efficiency (§ 4).
4. **Natural test-time scaling.** The number of pilot paths P and estimation paths N can be freely chosen at inference: P trades off identification error against pilot cost, and N trades off Monte Carlo variance against estimation cost, without retraining.

3.2 ARCHITECTURE

The architecture has two encoders, each having a different role in the overall model and one decoder that produces the final controller for either importance sampling or control variates, see Fig. 1. The

first encoder is the *pilot path encoder*, which takes as input the set of observed paths X^{pilot} and the corresponding functions $F(X^{\text{pilot}})$, and produces a context vector that summarizes the observed paths. The primary role for this component is to embed and identify the data generating process from the limited observations. The second encoder, the *function probe encoder*, evaluates the target function on a set of predefined perturbations of the expectation of the pilot paths and can thus identify important regions of the function. Finally, the task context produced by these two encoders is decoded to either predict the importance sampling drift or control variate integrand.

Pilot path encoder. Each pilot path is encoded individually and then pooled to produce a context vector in the fashion of Zaheer et al. (2017). To encode each path, we first normalize the path and its first difference, and then concatenate these with a normalized time index to obtain tokens of $u = [\widehat{X}, \Delta\widehat{X}, t/T]$. We then combine u with the function and its absolute value via a single layer MLP to form an embedding of both the path and the function applied to the path. We then use factorized linear attention (Katharopoulos et al., 2020) to pool the tokens over their number of paths dimension and finally consider the average over time and the terminal time to embed into an MLP. This produces c_{pilot} which will be used in the next sections.

Function probe encoder. Often, important regions of the function may be in sparsely sampled regions of the state space, and only exposing the pilot paths may miss important components of the function. We introduce the function probe encoder as a means to inform the model of which regions have important contributions to the underlying expectation we want to compute. We do this by averaging the input paths and then perturbing the average path by modes derived from the empirical pilot paths. Specifically, when given P pilot trajectories, $\bar{X}_{t,j} = \frac{1}{P}\sum_{i=1}^{P} X_{t,j}^{\text{pilot},(i)}$ with $t \in [1,\dots,T]$ and $j \in [1,\dots,d]$ and define $X_{t,j}^{\text{probe},(k\ell)} = \bar{X}_{t,j} + \underbrace{a_k(t)}_{\text{temporal pattern}} \underbrace{d_{\ell,j}}_{\text{direction}} \underbrace{s_j}_{\text{observed scale}}$.

Intuitively, $a_k(t)$ can be thought of as one of K fixed temporal patterns that include shifts, ramps, and oscillations; d_ℓ can be thought of a normalized cosine direction; s_j adapts the perturbations to match the scale in the pilot paths. These are normalized and embedded using their mean over time and terminal-time representations. The path embedding is concatenated with three transformations of its normalized function response: a bounded value via $\tanh$, a signed-logarithmic range, and an indicator of nonzero entries. These transformations allow the network to easily determine: the signed direction of the function *without* possible extreme values; the relative magnitude; and, how sparse is the function applied to the inputs. This is passed through a MLP and attention is computed on the KL different probes. Finally, we append the means and maxima of the three response channels to the pooled representations which are passed to the output MLP to obtain c_{probe}.

Importance sampling controller. The final component provides the necessary change of measure for the importance sampling conditioned on the context, $c_0 = [c_{\text{pilot}}\, c_{\text{probe}}]$, with c being the normalized context version of c_0. Both representations are used, the c conditions the adapters of the importance sampling controller while c_0 is used in the mixture weights. Let p be the distribution of noise given by the simulator (i.e. a normal distribution when considering Brownian motion driven dynamics). We do this by adapting the weights of an output MLP g_ϕ that parameterizes a Gaussian mixture through a low-rank adaptation of a shared base weight ϕ. The learned importance sampling proposal is constructed $q_{\text{learned}}(z_{1:T} \mid c_0) = \sum_{m=1}^{M} \pi_m(c_0) \prod_{t=1}^{T} \mathcal{N}\Big(z_t; \mu_t^{(m)}(c), I\Big)$ where $\pi(c_0)$ is the mixture weight for M mixtures. The final proposal then uses a convex combination of the original density and the proposal density:

$$q(z_{1:T} \mid c) = \alpha p(z_{1:T}) + (1-\alpha) q_{\text{learned}}(z_{1:T} \mid c),$$

where $\alpha = 0.2$ is the value used in the experimental results. The mixture aims to limit catastrophic failure when a learned shift is inappropriate. For proposal samples $z^{(i)} \sim q$, the final estimator is

$$\widehat{I}_N^{\text{IS}} = \frac{1}{N}\sum_{i=1}^{N} F\Big(X(z^{(i)})\Big) \exp\Big(\log p(z^{(i)}) - \log q(z^{(i)} \mid c)\Big).$$

We minimize the ratio of the log of the variance of $\hat{I}_N^{\text{IS}}$ versus standard Monte Carlo as the loss during training.

Control variate controller. The final component parameterizes the integrand h of the linear control variate of § 2.2. As for the importance-sampling controller, we adapt the weights of an output MLP

g_ϕ conditioned on the context c through a rank-four adaptation of a base weight. In addition to the normalized time t/T, g_ϕ takes as input the current state X_t and running statistics s_t of the path (e.g. running mean, maximum, and minimum): $h_t(X_t) = g_\phi(X_t, t/T, s_t; c)$. With $\Delta W_t^{(i)}$ the noise increments given by the simulator, the final estimator is

$$\widehat{I}_N^{\mathrm{CV}} = \frac{1}{N}\sum_{i=1}^{N}\left[F\big(X^{(i)}\big) - \sum_{t=0}^{T-1} h_t\big(X_t^{(i)}\big)^\top \Delta W_t^{(i)}\right],$$

which is unbiased for any h since h_t only depends on the path up to time t. At pretraining, we minimize the squared residual of the estimator around an auxiliary mean prediction, used only during training.

4 PRETRAINING

The two pretraining distributions differ between importance sampling and control variates since each method is intended to be used in different regimes: importance sampling primarily for tail events and control variates primarily for regular functions.

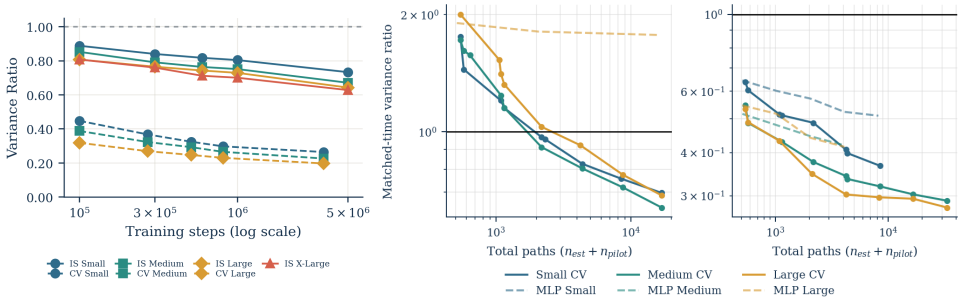
Importance sampling. For importance sampling, we are largely concerned with estimating rare events and learning shifts of the distribution that promote realizations of rare events as defined by F. We choose our pretraining distribution to emphasize this behavior by sampling dynamics and path dependent F that promote diversity in temporal complexity, proposal geometry (i.e. different shapes the model *should* learn to reduce variance), dimensional importance, and event probability. For proposal geometries, we emphasize those that move mass to the tails, localized regions (e.g. a set of interest), and different modes (e.g. moving from one stability region to another). An example of this design strategy is in Fig. 1 (A). We train on different target event probabilities based on values in a set that ranges from very rare events to more common events. We train on dimensions $d \in \{2, 8, 16\}$; time steps $T \in \{16, 64, 128\}$. The number of pilot paths is chosen from $\{128, 256, 512\}$. See the supplementary material for details about the pretraining distribution.

Control variates. The control-variate targets mainly financial applications so we pretrain on ten equiprobable financial process–payoff profiles spanning fixed variance, stochastic variance, and jumps, with active parameters sampled independently and uniformly from fixed ranges. The financial benchmark in §5.1 shares these families. We also sample the number of dimensions $d \in \{1, 2, 8, 16\}$, the number of time steps $T \in \{32, 64, 128\}$.

Pretraining dynamics and scaling. We train four different model sizes for importance sampling SMALL, MEDIUM, LARGE, and X-LARGE, with 141k, 389k, 1.20m, 4.12m parameters respectively for 5 million pretraining steps. For control variates, we train 3 models SMALL, MEDIUM, and LARGE with 80k, 230k, 750k parameters respectively for 3.5 million pretraining steps. We report the validation variance ratios across training checkpoints and model sizes in Fig. 2a. We see that both methods yield variance improvements compared to Monte Carlo, even on this very hard pretraining mix for importance sampling. We observe continuous improvements across both model size and pretraining steps, with the largest model achieving the best held-out variance reduction on the pretraining distribution.

5 EMPIRICAL RESULTS

We now report the numerical efficacy of the proposed foundation models. To do this, we have a few metrics we primarily consider. The first is *variance ratio*, which is the ratio of the variance of the estimator of the foundation model and naive Monte Carlo at the same number of estimation samples. The second one takes into account computation time: we report the *matched-time variance*, which is the ratio of the variance of the estimator of the foundation model and naive Monte Carlo with the same computation time. This matched-time metric is the most relevant for practical applications, as it quantifies the variance reduction achieved by the foundation model while factoring in the computational overhead of running such a model. Runtime includes pilot simulation, functional probes, controller construction, and estimation. Both our method and the Monte Carlo baseline are implemented in JAX (Bradbury et al., 2018), compiled with XLA (Sabne, 2020) and run on NVIDIA



(a) Scaling behavior

(b) Control variate model on the financial benchmark

Figure 2: **(a)** Validation variance ratios across training checkpoints for importance sampling and control variates on their respective pretraining distributions, lower is better. **(b)** Control variate model performance on the financial benchmark under IN-DISTRIBUTION (left) and OUT-OF-DISTRIBUTION (right): matched-time variance relative to naive Monte Carlo as a function of the number of estimation and pilot paths. Values < 1 indicate a benefit at a fixed computational budget. Dashed lines indicate task specific benchmark.

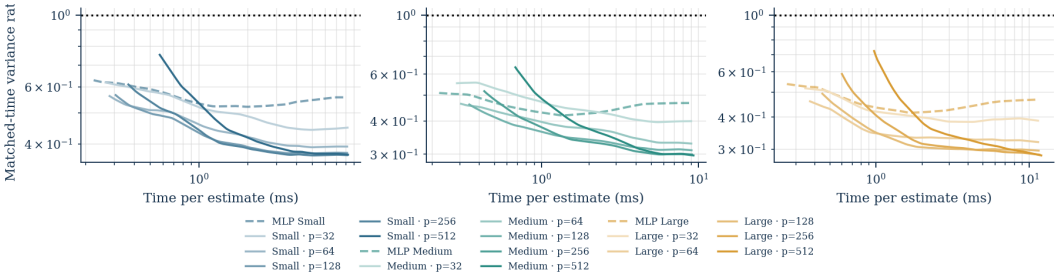


Figure 3: Control variate foundation model on the OUT-OF-DISTRBUTION financial benchmark: from left to right, small to large models. We plot the matched time variance ratio as a function of estimation time. Solid curves use 32, 64, 128, 256, or 512 pilot paths; dashed curves show the MLP baselines. Values < 1 indicate a gain over Monte Carlo. These plots show how increasing the number of pilot paths lowers variance at the cost of estimation time.

V100 and H100 GPUs. Our Monte Carlo baseline is thus highly competitive and is often faster than the implementations used in the literature[1]. Our models must carefully trade-off the variance reduction they achieve with the computational overhead they introduce.

We outline the results in two major parts. First, we present the control variates results on financial benchmarks and then we present the importance sampling results on engineering tasks. These are the primary use-cases for the methods and illustrate general performance on relevant tasks, and those that are far out of distribution in the engineering case. Second, we demonstrate how the gains carry over to real-world data and secondary tasks where variance reduction is necessary, including real volatility surfaces and limit order book data as well as stochastic optimization problems. We choose not to present results for both importance sampling and control variates on all benchmarks to highlight the relative strengths of each method.

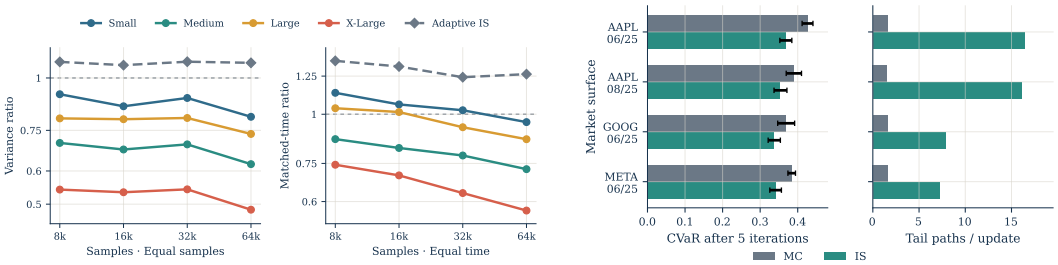
5.1 CONTROL VARIATES ON FINANCIAL BENCHMARKS

We first evaluate the control-variate models on the financial benchmark, which consists of 10 different stochastic processes and payoff functionals, and matches the pretraining distribution. We evaluate two regimes: (i) IN-DISTRIBUTION, low stress (ii) OUT-OF-DISTRIBUTION, higher stress, with some parameters 20% outside of training ranges. We compare both to a Monte Carlo baseline and to a learned task-specific MLP oracle. Given the test task-parameters, the latter consists in training a MLP with the same size as the controller of our models to minimize the variance of the control variate estimator on this precise task, with the training time not included in the reported wall-clock time. This provides a strong task-specific baseline. For Monte-Carlo, we consider a number of estimation

[1]E.g. more than 80 times faster than in Hinds & Tretyakov (2023).

paths that range from 512 to 5.24×10^5; for the foundation model we use a number of pilot paths between 32 and 512 paths and a number of estimation paths between 512 and 3.2×10^4.

We show the results in Fig. 2b: we plot the matched-time variance as a function of total number of paths ($n_{\text{pilot}} + n_{\text{est}}$) showing how the foundation model achieves a better matched-time variance < 1 as soon as the number of paths is sufficient to identify the task. The foundation models also perform better than the task-trained MLP reference, showing that it is able to correctly identifies the task at hand. In Fig. 3, we demonstrate the test-time scaling of the foundation model, taking the OUT-OOF-DISTRIBUTION regime as an example: it shows that, as we increase the number of pilot paths, the estimation time increases but matched-time improves. Moreover, at smaller model sizes, increasing the number of pilot paths is key to beating the MLP baseline. See § A.2 for a zero-shot application to the Newton method in this context and § A.1.2 for the results for importance sampling, where this benchmark is fully out-of-distribution.



(a) Engineering tasks.

(b) Autocallable hedging.

Figure 4: **(a)** Aggregate out-of-distribution performance on six engineering tasks for importance sampling. Left is variance ratio, right is matched-time variance ratio. **(b)** Estimating the optimal hedge for an autocallable options contract under a real-world volatility surface using CVaR at the 99.5% confidence level. Left plot shows the loss after five iterations, and the right plot shows the average number of paths exceeding the 99.5% value-at-risk threshold at each hedge policy update.

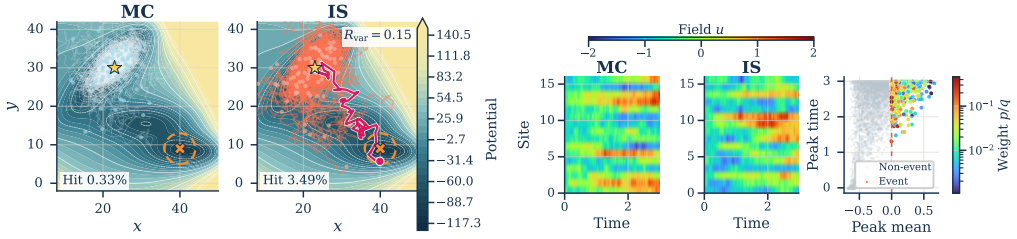


Figure 5: Comparing the learned importance weights for Müller-Brown and Allen-Cahn. In both cases, the network induces larger event probabilities on the region of interest as compared with the pure Monte Carlo baseline.

5.2 IMPORTANCE SAMPLING RARE EVENT ENGINEERING BENCHMARKS

To test out of distribution behavior, we consider a set of benchmark tasks inspired by canonical models used in physical sciences. We modify the models by including stochastic components within the deterministic dynamics. Specifically, we consider models inspired by the Lorenz-96 system (Lorenz, 1996), power-grid dynamics (Dörfler et al., 2013), stochastically forced Duffing oscillator (Ueda, 1991), a diffusion process representing queuing networks (Reiman, 1984), Schlögl model for chemical reactions (Schlögl, 1972), and a spatial subset of the Allen-Cahn equation for phase-separation with stochastic fluctuations. Our main goal for these experiments is to estimate some rare event probability, which has the form of $F(X_{0:T}) = \mathbb{1}\{\max_{1 \le t \le T} \ell(X_t) \ge b\}$. The expectation of this describes the probability that a function of the process exceed some rare threshold given by b over any point in time. We calibrate b such that the event has roughly a 1% probability of occurring. We report the variance ratio for the different model sizes in Fig. 4a. The results are compared to a task specific, two stage importance sampling baseline (Adaptive IS) that uses extra information of the exact noise

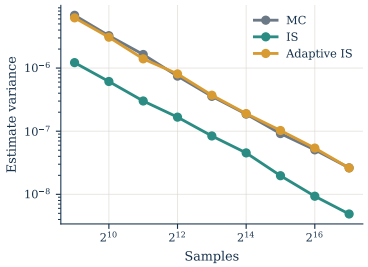


Figure 6: Variance reduction on Müller-Brown

Table 1: Importance sampling variance reduction multiplied by inference time ratio on real local-volatility pricing tasks (smaller is better).

Task	Small	Medium	Large	X-Large
Digital call	0.74	0.28	0.12	0.23
OTM Call	0.59	0.20	0.05	0.19
Digital put	0.71	0.39	1.99	0.56
Knock-in	0.43	0.74	1.96	0.27
Autocall	0.78	0.35	1.57	0.35
Overall	0.64	0.36	0.51	0.30

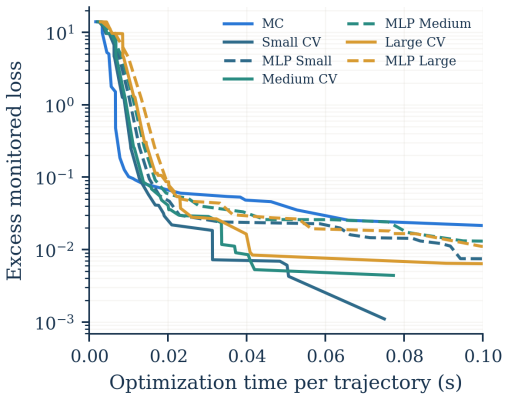


Figure 7: Limit-order book optimization with CEM on July data. Curves report excess monitored loss over the best observed value against complete optimization time per trajectory; lower is better.

increments used to generate the data to find an optimal importance weight, with full details in the appendix. We also study the Müller-Brown potential, which describes the transition of a particle in one low energy state to another (Müller & Brown, 1979) with the dynamics illustrated on the left of Fig. 5. The variance reduction provided by the proposed IS variant is given in Fig. 6.

5.3 Financial Experiments with Real-World Parameters

Pricing. We use historical volatility surfaces (OptionMetrics, LLC, 2026) to evaluate the performance of the importance sampling foundation model on real-world data. Table 1 reports variance reduction and variance reduction multiplied by the time ratio (i.e. $\frac{t_{IS}}{t_{MC}}$) on pricing tasks for different option types, driven by market-calibrated local-volatility surfaces; see § A.3 for control variates.

Deep hedging. We use the importance sampling model to accelerate deep hedging (Bühler et al., 2018) of an autocallable contract, under the same market-calibrated volatility surfaces. The hedging strategy minimizes the conditional value at risk (CVaR) of the hedging error at the 99.5% level, which depends only on rare tail events. As shown in Fig. 4b, importance sampling produces substantially more tail paths per update than Monte Carlo, leading to a lower CVaR after five iterations.

Finetuning for market making. We consider a limit-order book simulator calibrated on two months of market data, June and July 2010, from Huang & Polak (2011). We finetune the control-variate model on the June calibration and deploy it on the July calibration without further training. The downstream task is optimal market making: a policy quotes continuously and is optimized with the cross-entropy method (CEM) to minimize $\mathbb{E}[-W_T + 0.1\,\mathrm{mean}_t Q_t^2 + 0.5 Q_T^2]$, where W_T is the terminal wealth and Q_t the inventory, with each candidate policy evaluated with the control-variate estimator. We compare against Monte Carlo and against an MLP control variate trained directly on July data, whose training time is excluded, with up to 512 estimation paths for the neural models and up to 8,192 for Monte Carlo. As shown in Fig. 7, the finetuned model reaches a better loss than both baselines for most optimization times, suggesting acceleration on downstream optimization.

6 Conclusion

We introduce foundation models for variance reduction in stochastic processes: given a set of pilot paths and their payoffs, these models directly output an importance-sampling drift or a control-variate integrand, without access to the parameters of the process. Pretrained on synthetic processes, they yield unbiased estimators, scale naturally at test time, and are efficient enough to outperform Monte Carlo at matched computational time. Extending these models to more complex processes is a natural next step.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology and experiments, implement methods, support qualitative and thematic data analysis, interpret results. We have not used generative AI tools to generate synthetic datasets, clean and reformat datasets, propose or refine hypotheses, or assist with translation. To help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools to create and modify the presented scientific figures or images, suggest experimental parameters, create and edit software code, draft parts of the manuscript, summarize or analyze existing literature, discover research topics or identify gaps, sourcing/searching for information, edit the manuscript to improve readability, identify relevant literature. We have reviewed all AI-assisted work. Code is tested and verified by the authors. Literature survey is done manually, using generative tools only to find potentially missed relevant references. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Supplementary material contains pretraining and evaluation details. Code will be made available upon publication.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=gerNCVqqtR`. Expert Certification.

Aleksandar Arandjelović, Thorsten Rheinländer, and Pavel V Shevchenko. Importance sampling for option pricing with feedforward neural networks. *Finance and Stochastics*, 29(1):97–141, 2025.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Hans Bühler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *arXiv preprint arXiv: 1802.03042*, 2018.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Florian Dörfler, Michael Chertkov, and Francesco Bullo. Synchronization in complex oscillator networks and smart grids. *Proceedings of the National Academy of Sciences*, 110(6):2005–2010, 2013.

Adam Golinski, Frank Wood, and Tom Rainforth. Amortized monte carlo integration. In *International Conference on Machine Learning*, pp. 2309–2318. PMLR, 2019.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *International Conference on Learning Representations*, 2017.

P. D. Hinds and M. V. Tretyakov. Neural variance reduction for stochastic differential equations. *Journal of Computational Finance*, 2023.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International*

Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

Xinru Hua, Rasool Ahmad, Jose Blanchet, and Wei Cai. Accelerated sampling of rare events using a neural network bias potential. In *AI for Accelerated Materials Design - NeurIPS 2023 Workshop*, 2023. URL https://openreview.net/forum?id=NyXTjtFojv.

Ruihong Huang and Tomas Polak. Lobster: Limit order book reconstruction system. *Available at SSRN 1977207*, 2011.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Fengpei Li, Haoxian Chen, Jiahe Lin, Arkin Gupta, Xiaowei Tan, Honglei Zhao, Gang Xu, Yuriy Nevmyvaka, Agostino Capponi, and Henry Lam. Prediction-enhanced monte carlo: A machine learning view on control variate. *arXiv preprint arXiv:2412.11257*, 2024.

Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, pp. 1–18. Reading, 1996.

Klaus Müller and Leo D. Brown. Location of saddle points and minimum energy paths by a constrained simplex optimization procedure. *Theoretica chimica acta*, 53:75–93, 1979. URL https://api.semanticscholar.org/CorpusID:98239957.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Transactions on Graphics (ToG)*, 38(5):1–19, 2019.

Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. Neural control variates. *ACM Transactions on Graphics (TOG)*, 39(6):1–19, 2020.

Bernt Øksendal. Stochastic differential equations. In *Stochastic differential equations: an introduction with applications*, pp. 65–84. Springer, 2003.

OptionMetrics, LLC. Ivydb database, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/qu25d.html.

Martin I Reiman. Open queueing networks in heavy traffic. *Mathematics of operations research*, 9 (3):441–458, 1984.

Marc Sabate Vidales, David Šiška, and Lukasz Szpruch. Unbiased deep solvers for linear parametric pdes. *Applied Mathematical Finance*, 28(4):299–329, 2021.

Amit Sabne. Xla : Compiling machine learning for peak performance, 2020.

Tamar Schlick. *Molecular modeling and simulation: an interdisciplinary guide*, volume 2. Springer, 2010.

Friedrich Schlögl. Chemical reaction models for non-equilibrium phase transitions. *Zeitschrift für physik*, 253(2):147–161, 1972.

Patrick Seifner, Kostadin Cvejoski, David Berghaus, Cesar Ojeda, and Ramses J Sanchez. In-context learning of stochastic differential equations with foundation inference models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ceCJPoZOKJ.

Ali Siahkoohi and Hyunwoo Oh. Conditional neural control variates for variance reduction in Bayesian inverse problems. In Emilija Perković and Daniel Malinsky (eds.), *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, volume 337 of *Proceedings of Machine Learning Research*, pp. 6321–6341. PMLR, 17–21 Aug 2026.

Martin Sipka, Johannes CB Dietschreit, Lukáš Grajciar, and Rafael Gómez-Bombarelli. Differentiable simulations for enhanced sampling of rare events. In *International Conference on Machine Learning*, pp. 31990–32007. PMLR, 2023.

Zhuo Sun, Chris J Oates, and François-Xavier Briol. Meta-learning control variates: Variance reduction with limited data. In *Uncertainty in Artificial Intelligence*, pp. 2047–2057. PMLR, 2023.

Yoshisuke Ueda. Survey of regular and chaotic phenomena in the forced duffing oscillator. *Chaos, Solitons & Fractals*, 1(3):199–231, 1991.

Ruosi Wan, Mingjun Zhong, Haoyi Xiong, and Zhanxing Zhu. Neural control variates for variance reduction. *arXiv preprint arXiv: 1806.00159*, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep sets. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf.

A ADDITIONAL EXPERIMENTAL RESULTS

A.1 FINANCIAL BENCHMARK

A.1.1 CONTROL VARIATES: ALL REGIMES

To complement the main text, we provide in Figs. 8 and 9 additional regimes: (i) IN-DISTRIBUTION LOW STRESS, (ii) IN-DISTRIBUTION SEVERE STRESS.

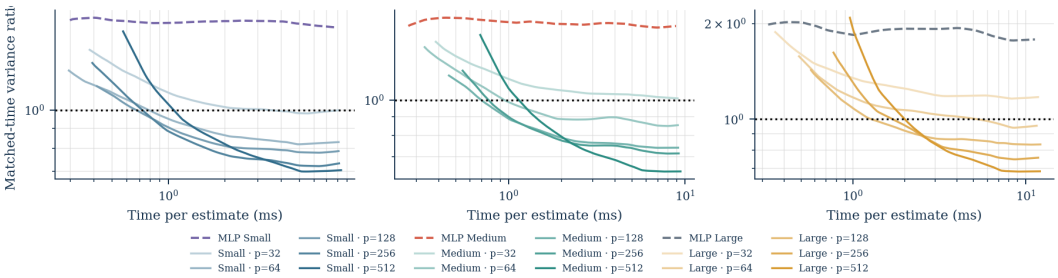


Figure 8: Control variate foundation model on the IN-DISTRIBUTION LOW STRESS financial benchmark: from left to right, small to large models. We plot the matched time variance ratio as a function of estimation time. Solid curves use 32, 64, 128, 256, or 512 pilot paths; dashed curves show the MLP baselines. Ratios below one indicate a gain over Monte Carlo, and the horizontal black solid reference marks parity with Monte Carlo. These plots show how increasing the number of pilot paths increases estimation time but ultimately lower the variance of the estimator.

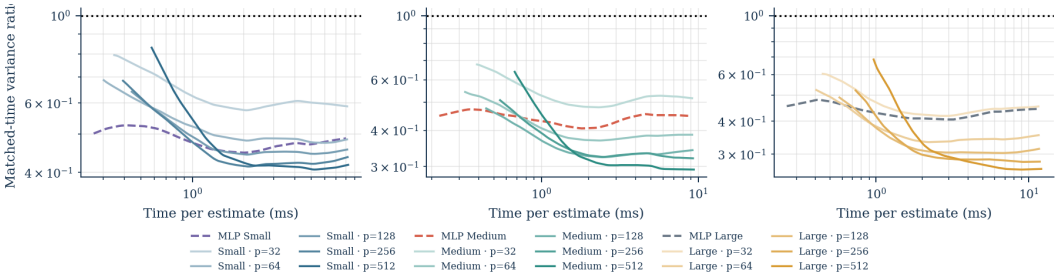


Figure 9: Control variate foundation model on the IN-DISTRIBUTION SEVERE STRESS financial benchmark: from left to right, small to large models. We plot the matched time variance ratio as a function of estimation time. Solid curves use 32, 64, 128, 256, or 512 pilot paths; dashed curves show the MLP baselines. Ratios below one indicate a gain over Monte Carlo, and the horizontal black solid reference marks parity with Monte Carlo. These plots show how increasing the number of pilot paths increases estimation time but ultimately lower the variance of the estimator.

A.1.2 IMPORTANCE SAMPLING

For importance sampling, even though we keep the same names as control variates for clarity, these benchmarks are all out of distribution.

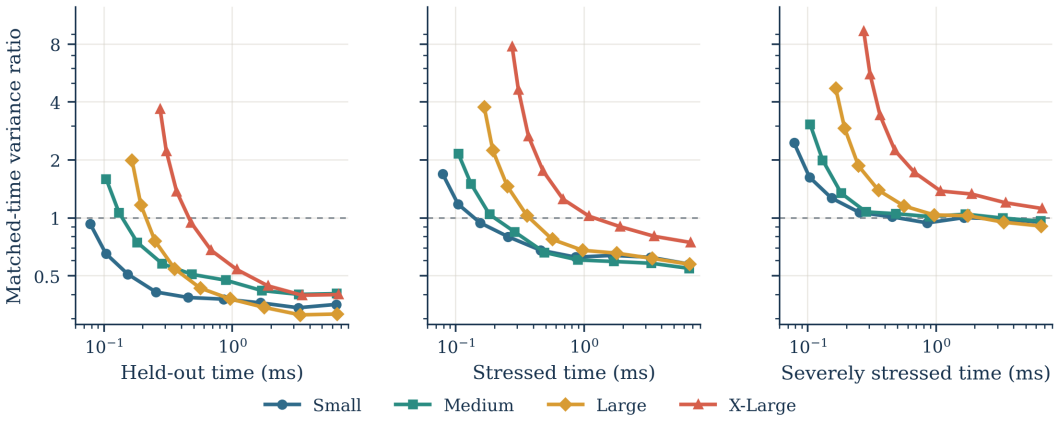


Figure 10: Importance sampling foundation model on the three regimes of the financial benchmark: from left to right, low to high stress. We plot the matched time variance ratio as a function of estimation time. Solid curves use 32, 64, 128, 256, or 512 pilot paths. Ratios below one indicate a gain over Monte Carlo, and the horizontal black solid reference marks parity with Monte Carlo. These plots show how increasing the number of pilot paths increases estimation time but ultimately lower the variance of the estimator.

A.2 Newton method with control variates

In this section, we present a synthetic task to demonstrate how the control variate foundation model can be used in a zero-shot fashion to accelerate Newton iterations.

Given a target option price $C^\star$, strike inversion consists in finding the strike K such that $C(K) = C^\star$. This is an instance of a common structuring problem, in which a contract parameter such as a strike, coupon or barrier is calibrated so that the product matches a target premium. We solve this equation with Newton's method.

As a model for the prices, we consider two of them: a Merton jump-diffusion model in dimension eight and a basket option on 16 assets. Both the price $C(K)$ and its slope $\partial_K C(K)$ are thus expectations of payoffs of stochastic processes and need to be estimated by Monte Carlo at each iteration. This experiment evaluates how variance reduction from the control variate model can help to speed up stochastic optimization.

Starting from $K_0 = 1.3$, Newton iterations consists in iterating

$$K_{t+1} = \Pi_{[0.5,\,1.6]}\left(K_t - \mathrm{clip}_{0.25}\left(\frac{\widehat{C}(K_t) - C^\star}{\max\{|\widehat{\partial_K C}(K_t)|,\,0.01\}}\right)\right),$$

where $\Pi_{[0.5,1.6]}$ denotes projection onto the admissible strike interval and $\mathrm{clip}_{0.25}$ limits the step size to 0.25. Slopes are estimated directly with control variates.

We compare standard Monte Carlo (MC), a task-specific MLP control variate (MLP) and our pretrained control-variate model (CV). The strike changes at every update, the foundation model is applied to new pilot paths every 8 or 16 iterations. For Merton, the runs use 256 Newton iterations, with estimation paths $N \in \{128, 256, 512\}$ and pilot paths $P \in \{64, 256\}$. For the basket task, we use 1,024 iterations up to 2,048 paths for control variate, 8,192 for MLP and 16,384 for Monte Carlo.

For each configuration, we run 128 independent Newton trajectories and report the mean optimization loss against elapsed wall-clock time. Reported times exclude the training of the MLP baseline.

Fig. 11 shows the resulting Pareto fronts. We see how the control variate foundation models, even if they were not trained for to estimate derivative, significantly improve the performance of the Newton method. The resulting procedure beats MC at matched-time and catches up to specifically tuned MLPs.

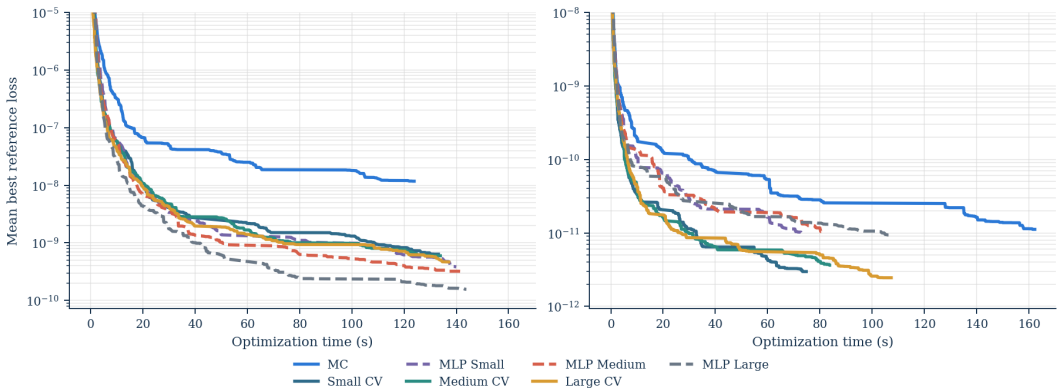


Figure 11: Strike inversion with Newton's method: mean optimization loss against wall-clock time for MC, MLP and CV, on the Merton-terminal (left) and Basket-terminal (right) tasks. Each point is a configuration (number of paths, pilots, refresh interval); lower-left is better.

A.3 Control-variate on market-calibrated volatility data

We report the results of the control variate model for a pricing task with the market-calibrated volatility data on a deep-OTM European put Table 2.

A.4 Control variates on Alaline benchmark

We evaluate transition-probability estimation on surrogate benchmarks inspired by the alanine-dipeptide setting of Sipka et al. (2023). Real torsion models are fitted to molecular-dynamics trajectories using 2 or 5 torsion coordinates. Real projected models use the same trajectories, retaining the first 8 or 16 coordinates of a 60-dimensional internal-coordinate representation ordered with torsions first. Synthetic models use artificial periodic Gaussian Markov dynamics calibrated to an approximately 3% transition probability, with no molecular trajectory source. See Table 3 for the results.

Table 2: Matched time variance by underlying and model size on a deep-OTM European put using market-calibrated volatility. Parentheses report $(n_{\text{pilot}}, n_{\text{est}})$ for the best configuration.

Underlying	Small	Medium	Large
AAPL	0.763 (32, 32k)	0.752 (128, 32k)	0.935 (64, 16k)
GOOG	0.735 (32, 32k)	0.719 (32, 32k)	0.893 (32, 32k)
META	0.654 (32, 4k)	0.649 (32, 4k)	0.725 (32, 4k)

Table 3: Variance ratio against Monte-Carlo for control variates on the Alaline benchmark. Parentheses report $(n_{\text{pilot}}, n_{\text{est}})$ for the best configuration.

Benchmark	Small	Medium	Large
Real torsion 2D	0.885 (512, 4k)	0.893 (256, 4k)	0.885 (512, 4k)
Real torsion 5D	0.820 (32, 8k)	0.820 (128, 8k)	0.820 (32, 8k)
Real projected 8D	0.752 (512, 16k)	0.758 (128, 16k)	0.758 (32, 16k)
Real projected 16D	0.885 (128, 2k)	0.877 (64, 2k)	0.877 (32, 2k)
Synthetic 2D	0.855 (128, 4k)	0.855 (512, 4k)	0.855 (128, 4k)
Synthetic 5D	0.800 (256, 1k)	0.806 (64, 1k)	0.800 (256, 1k)
Synthetic 8D	0.877 (64, 32k)	0.877 (32, 32k)	0.877 (64, 32k)

B Architectural Details

We provide more detail on the overarching methodology introduced in the main text. The goal in this section is to illustrate the components of the architecture directly through equations instead of textually as done in the main text.

B.1 Pilot path encoder

As mentioned, the purpose of this component is to identify correlations between realized sample paths in the context.

The full definition of the pilot path transformations are given here:

$$
\begin{aligned}
\mu_0 &= \frac{1}{P}\sum_{i=1}^{P} X_0^{(i)}, \\
s_x &= \left[\frac{1}{PT}\sum_{i,t}\left(X_t^{(i)} - \mu_0\right)^{\odot 2} + \varepsilon\right]^{1/2}, \\
\widetilde{X}_t^{(i)} &= \frac{X_t^{(i)} - \mu_0}{s_x}, \\
\widetilde{y}_i &= \operatorname{clip}\left(\frac{y_i - \bar{y}}{s_y}, -20, 20\right), \qquad y_i = F(X^{(i)}), \\
h_{it}^{\text{pilot}} &= \tanh\left[\operatorname{MLP}_x\left(\widetilde{X}_t^{(i)},\, \Delta\widetilde{X}_t^{(i)},\, \frac{t}{T}\right) + \operatorname{MLP}_y(\widetilde{y}_i,\, |\widetilde{y}_i|)\right], \\
u_t^{\text{pilot}} &= \operatorname{Attn}_{i=1:P}\left(h_{1t}^{\text{pilot}}, \ldots, h_{Pt}^{\text{pilot}}\right), \\
c_{\text{pilot}} &= \operatorname{MLP}_{\text{pilot}}\left(\left[\frac{1}{T}\sum_{t=1}^{T} u_t^{\text{pilot}},\; u_T^{\text{pilot}}\right]\right).
\end{aligned}
\tag{3}
$$

B.2 FUNCTION PROBE ENCODER

The purpose of this component is to sample the functional space with higher fidelity than available through the base distribution itself. The full set of transformations is given by:

$$
\begin{aligned}
\bar{X}_t &= \frac{1}{P}\sum_{i=1}^{P} X_t^{(i)}, \\
X^{\text{probe}}_{k\ell,t} &= \bar{X}_t + a_k(t)\, d_\ell \odot s^{\text{probe}}, \\
y^{\text{probe}}_{k\ell} &= F\Big(X^{\text{probe}}_{k\ell}\Big), \\
\widetilde{y}^{\text{probe}}_{k\ell} &= \text{clip}\left(\frac{y^{\text{probe}}_{k\ell}}{\text{RMS}(y_{1:P})}, -20, 20\right), \\
\rho_{k\ell} &= \begin{bmatrix} \tanh\Big(\widetilde{y}^{\text{probe}}_{k\ell}\Big) \\ \text{sgn}\Big(\widetilde{y}^{\text{probe}}_{k\ell}\Big)\log\Big(1+\Big|\widetilde{y}^{\text{probe}}_{k\ell}\Big|\Big) \\ \mathbf{1}_{\{\widetilde{y}^{\text{probe}}_{k\ell}\neq 0\}} \end{bmatrix}, \\
e^{\text{path}}_{k\ell} &= \text{MLP}_{\text{path}}\left(\left[\frac{1}{T}\sum_t \text{MLP}_x\Big(\widetilde{X}^{\text{probe}}_{k\ell,t}, \Delta\widetilde{X}^{\text{probe}}_{k\ell,t}, \frac{t}{T}\Big),\ \text{MLP}_x\Big(\widetilde{X}^{\text{probe}}_{k\ell,T}, \Delta\widetilde{X}^{\text{probe}}_{k\ell,T}, 1\Big)\right]\right), \\
z^{\text{probe}}_{k\ell} &= \text{MLP}_{\text{token}}\Big([e^{\text{path}}_{k\ell}, \rho_{k\ell}]\Big), \\
c_{\text{probe}} &= \text{MLP}_{\text{probe}}\left(\left[\text{Attn}_{k,\ell}\Big(z^{\text{probe}}_{k\ell}\Big),\ \underset{k,\ell}{\text{mean}}\rho_{k\ell},\ \underset{k,\ell}{\max}\rho_{k\ell}\right]\right).
\end{aligned}
\tag{4}
$$

We used the following choices for $a_k(t)$ and d_ℓ:

Here $u_t = t/T$, G is the total number of probes, and $L = G/8$.

B.3 IMPORTANCE SAMPLING CONTROLLER

These contexts get passed into the importance sampling weight controller which then performs the following transformations:

$$
\begin{aligned}
c &= \text{RMSNorm}([c_{\text{pilot}}, c_{\text{probe}}]), \\
(A_\ell(c), B_\ell(c)) &= \text{reshape}(G_\ell c + g_\ell), \qquad W_\ell(c) = W_\ell^{(0)} + \frac{\gamma}{r}A_\ell(c)B_\ell(c), \qquad r = 4, \\
\mu_{m,t} &= \mu_{\max}\tanh\big[\text{MLP}_{\{W_\ell(c)\}}(\phi_t)\big], \qquad \phi_t = \left[0,\ldots,0,\frac{t}{T},0,\ldots,0\right], \\
\pi(c) &= \text{softmax}([W_\pi c + b_\pi,\ 0]), \\
p(\epsilon) &= \mathcal{N}(\epsilon; 0, I), \qquad q_{\text{learned}}(\epsilon \mid c) = \sum_{m=1}^{K}\pi_m(c)\mathcal{N}(\epsilon;\mu_m, I), \\
q(\epsilon \mid c) &= \alpha p(\epsilon) + (1-\alpha)q_{\text{learned}}(\epsilon \mid c), \\
w(\epsilon) &= \frac{p(\epsilon)}{q(\epsilon \mid c)} = \left[\alpha + (1-\alpha)\sum_{m=1}^{K}\pi_m(c)\exp\left\{\sum_{t,j}M_j\left(\epsilon_{t,j}\mu_{m,t,j} - \frac{1}{2}\mu^2_{m,t,j}\right)\right\}\right]^{-1}.
\end{aligned}
\tag{5}
$$

Table 4: Temporal patterns and spatial directions used to construct the function probes $X_{t,j}^{\mathrm{probe},(k\ell)} = \bar{X}_{t,j} + a_k(u_t)d_{\ell,j}s_j$, where $u_t = t/T$. Every temporal pattern is paired with every spatial direction, producing $G = 8L$ probes.

k	Qualitative behavior	$a_k(u)$
1	Positive constant shift	0.75
2	Negative constant shift	-0.75
3	Positive ramp	$1.5u$
4	Negative ramp	$-1.5u$
5	Positive localized hump	$3\sin(\pi u)$
6	Negative localized hump	$-3\sin(\pi u)$
7	Midpoint switch	$2\,\mathbf{1}_{\{u\geq 1/2\}} - 2\,\mathbf{1}_{\{u<1/2\}}$
8	Oscillation	$2\sin(2\pi u)$

Quantity	Definition	Interpretation
Direction slot	$\ell \in \{0, \ldots, L-1\}$	Spatial probe index
Cosine mode	$m_\ell = \ell \bmod d$	Mode repeated across amplitude tiers
Raw direction	$c_{\ell,j} = \cos\left(\frac{\pi m_\ell (j+\frac{1}{2})}{d}\right)$	Discrete cosine direction
RMS normalization	$\widehat{c}_{\ell,j} = \frac{c_{\ell,j}}{\sqrt{d^{-1}\sum_{q=0}^{d-1} c_{\ell,q}^2}}$	Unit RMS across coordinates
Tier index	$r_\ell = \left\lfloor \frac{\ell}{d} \right\rfloor$	Repeats modes at larger amplitudes
Number of tiers	$R = \left\lceil \frac{L}{d} \right\rceil$	Depends on probe count and dimension
Tier amplitude	$\lambda_\ell = 1 + \frac{1.5 r_\ell}{\max(R-1, 1)}$	Constrains magnitude to be in $[1, 2.5]$
Final direction	$d_{\ell,j} = \lambda_\ell \widehat{c}_{\ell,j}$	RMS-normalized, amplitude-scaled direction

B.4 Control variate controller

For the control variate output head, we have a similar architecture with the main differences being that the outputs are different from the importance sampling weights.

$$
\begin{aligned}
c_i^{\mathrm{pilot}} &= \mathrm{PilotEnc}\Big(\Big\{(X^{(j)}, Y_j) : j < B(i)\Big\}\Big), \qquad Y_j = F(X^{(j)}),\\
c^{\mathrm{probe}} &= \mathrm{ProbeEnc}\left(\left\{X_k^{\mathrm{probe}}, F(X_k^{\mathrm{probe}})\right\}_{k=1}^{K_{\mathrm{probe}}}\right),\\
c_i &= c_i^{\mathrm{pilot}} + \mathrm{MLP}_{\mathrm{probe}}\big(c^{\mathrm{probe}}\big), \qquad \widehat{c}_i = \mathrm{RMSNorm}(c_i),\\
(A_{\ell,i}, B_{\ell,i}) &= \mathrm{reshape}(G_\ell \widehat{c}_i + g_\ell), \qquad W_{\ell,i} = W_\ell^{(0)} + \frac{\alpha_{\mathrm{ad}}}{r} A_{\ell,i} B_{\ell,i}, \qquad r = 4,\\
\phi_{i,t} &= \left[X_{t-1}^{(i)},\, \overline{L}_{<t}^{(i)},\, L_{<t}^{(i),\max},\, \frac{t}{T},\, W_{t-1}^{(i)},\, \chi_i\right],\\
h_{i,t} &= 8\,\tanh\big[\mathrm{MLP}_{\{W_{\ell,i}\}}(\phi_{i,t}; c_i)\big] \odot M,\\
C_i &= \chi_i \sum_{t=1}^{T} A_t\, h_{i,t}^{\mathsf{T}} \Delta W_t^{(i)}, \qquad \Delta W_t^{(i)} = \sqrt{\Delta t}\, Z_t^{(i)},\\
r_i &= \left[\frac{1}{i}\sum_{j<i}\left(Y_j - \overline{Y}_{<i}\right)^2 + \varepsilon\right]^{1/2},\\
\widehat{\mu}_{\mathrm{CV}} &= \frac{1}{N}\sum_{i=1}^{N}\left[Y_i - r_i C_i\right].
\end{aligned}
\tag{6}
$$

C Adaptive Importance Sampling Baseline

The adaptive importance sampling baseline uses P_{fit} independent pilot paths generated under the true measure p to estimate $\widehat{\mu}_t = \frac{\sum_{i=1}^{P_{\text{fit}}} |F(X^{(i)})_t| Z_t^{(i)}}{\sum_{i=1}^{P_{\text{fit}}} |F(X_t^{(i)})|}$ with $Z^{(i)}$ being the latent noise sequence that generated $X_t^{(i)}$. We then consider using no shift ($\mu_t = 0$), constant shift ($\mu_t = \frac{1}{T}\sum_{t=1}^{T} \widehat{\mu}_t$), 4 and 8 block shift (average over blocks of 4 and 8 respectively), and fully time independent versions of the proposal shift. We also consider 3 shift magnitudes of $\xi \in \{0.25, 0.5, 1\}$ such that we apply $\xi\mu_t$ to give 13 different proposal distributions. We choose the best variant of these 13 combinations by using a held out set of P_{tune} paths which chooses the lowest variance estimator of the 13. This provides a comprehensive baseline that shows what an importance sampler can provide in terms of variance reduction given additional information of the realized latent noise paths.