Training with a Feynman-Kac Inspired Loss Function

Danilo Jr Dela Cruz
Mathematics of Random Systems
University of Oxford

30th January 2024

Abstract

In order to solve the filtering equations, Crisan et al. [2022] solve a PDE using a neural network trained on a Feynman-Kac inspired loss function $\mathcal{L}$. To accelerate training, we propose variance reduction techniques in the Monte Carlo estimate of $\mathcal{L}$. Furthermore, we provide an error decomposition which relates $\mathcal{L}$ to the solution's L^2 error and quantifies the other sources of error accrued during the process of estimating $\mathcal{L}$.

1 Introduction

Given a filtered probability space $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$, we have a signal process $X_t \in \mathbb{R}^d$ and observation process $Y_t \in \mathbb{R}^m$; modelling noise via independent standard brownian motions $V_t \in \mathbb{R}^p, W_t \in \mathbb{R}^m$.

$$dX_t = f(X_t)dt + \sigma(X_t)dV_t \tag{1}$$
$$dY_t = h(X_t)dt + dW_t \tag{2}$$

An important step in Crisan et al. [2022]'s algorithm is to solve PDEs of the form 3 over $(0, T]$

$$\begin{aligned}\frac{\partial u}{\partial t}(t,x) &= A^{\star}u(t,x) \\ u(0,x) &= \psi(x)\end{aligned} \tag{3}$$

where $A^{\star}$ is the adjoint of the generator A of X_t.

$$u(T,x) = \mathbb{E}\left[q(T,x)\right] \tag{4}$$
$$q(t,x) = \psi(\widehat{X}_t)\exp\left(\int_0^t r(\widehat{X}_s)ds\right) \tag{5}$$

Feynman-Kac provides a stochastic representation for the PDE solution, see Corollary 2.3 of Crisan et al. [2022], where $\widehat{X}_t$ is a diffusion and $\widehat{X}_0 = x$.

$$d\widehat{X}_t = b(\widehat{X}_t)dt + \sigma(\widehat{X}_t)dV_t \tag{6}$$
$$b = 2\overrightarrow{\mathrm{div}}\,(a) - f \tag{7}$$
$$a = \frac{1}{2}\sigma\sigma^T \tag{8}$$
$$r = \mathrm{div}\left(\overrightarrow{\mathrm{div}}\,(a) - f\right) \tag{9}$$

$\overrightarrow{\mathrm{div}}\,(\cdot)$ is the vector obtained by taking the divergence of each column of a matrix-valued function.

$$\mathrm{div}\,(f) = \sum_{i=1}^{d}\partial_i f, \quad \overrightarrow{\mathrm{div}}\,(a)_j = \mathrm{div}\,(a_{\cdot j}) \tag{10}$$

1.1 Loss Function

The expectation $\mathbb{E}\left[X\right]$ corresponds to the value μ which minimises the mse $\mathbb{E}\left[|X-\mu|^2\right]$. Proposition 2.4 in Crisan et al. [2022] extends this analogy to obtain the Feynmac-Kac inspired loss function $\mathcal{L}$.

Theorem 1.1. *Consider the PDE 3 and stochastic representation $u(t,x)$ given in equation 4. For any $a < b$ and $D \sim \mathrm{Unif}([a,b]^d)$, $U(x) = u(T,x)$ is the unique continuous function which minimises $\mathcal{L}(v)$ over all continuous function $v \in C([a,b]^d, \mathbb{R})$.*

$$\mathcal{L}(v) = \mathbb{E}\left[|q(T,D) - v(D)|^2\right] \tag{11}$$

Given this formulation, we can let $v = \mathcal{N}_\theta$ be a neural network and minimise $\mathcal{L}$. This method has the advantage over other PDE solvers that it does not require boundary conditions. Instead, we aim to fit over a fixed space domain $[a,b]^d$.

1.2 Contributions

Our first contribution is an error decomposition which relates $\mathcal{L}$ to the L^2 error: the quantity typically considered when evaluating PDE solutions.

Furthermore, estimation of $\mathcal{L}$ introduces sources of errors, see Table 1. The error decomposition provides insight on the interaction of these errors and suggests a principled way to choose hyperparameters such as Monte Carlo sample size, step sizes and strong error order of the SDE approximation.

The second contribution, are two variance reduction techniques on the loss function which accelerate the training of the neural network.

2 Error Decomposition

To judge the quality of a PDE solution, we consider the L^2 error; which can be equivalently expressed as an expectation over $D \sim \text{Unif}([a,b]^d)$.

$$\begin{aligned} \mathcal{R}(\theta) &= \|U - \mathcal{N}_\theta\|_2^2 && (12)\\ &= \mathbb{E}\left[|U(D) - \mathcal{N}_\theta(D)|^2\right]. && (13) \end{aligned}$$

As we do not have access to U, we will try to match its stochastic representation $Q(x) := q(T, x)$.

$$U(x) = \mathbb{E}\left[Q(x)\right] := \mathbb{E}_x\left[Q\right]^{1} \quad (14)$$

Furthermore, we will only have an approximation $\widetilde{Q}$ to Q as the SDE solution and integral are approximated. Casing on $D = x$, "adding 0" and taking expectations over the SDE's randomness yields

$$\begin{aligned} |U - \mathcal{N}_\theta|^2 = \; & \mathbb{E}_x\left[(\widetilde{Q} - \mathcal{N}_\theta)^2\right] \\ + \; & \mathbb{E}_x\left[(Q - \widetilde{Q})^2\right] \\ + \; & \mathbb{E}_x\left[(U - Q)^2\right] \\ +2 & \mathbb{E}_x\left[(Q - \widetilde{Q})(\widetilde{Q} - \mathcal{N}_\theta)\right] \\ +2 & \mathbb{E}_x\left[(U - Q)(Q - \mathcal{N}_\theta)\right]. \end{aligned}$$

Taking expectations over D, we arrive at a decomposition in terms of $\mathcal{L}$, the SDE approximation error $\mathcal{S}$ incurred by using $\widetilde{Q} \approx Q$ and a Feynman-Kac approximation error $\mathcal{F}$ incurred from using the stochastic representation $Q \approx U$.

$$\begin{aligned} \mathbb{E}\left[(U - \mathcal{N}_\theta)^2\right] &= \mathcal{R} = \mathcal{L} + \mathcal{S} + \mathcal{F} + C && (15)\\ \mathbb{E}\left[(\widetilde{Q} - \mathcal{N}_\theta)^2\right] &= \mathcal{L} \leq \eta\kappa(\delta_1 + \delta_2)/2 + \widetilde{\mathcal{L}} && (16)\\ \mathbb{E}\left[(Q - \widetilde{Q})^2\right] &= \mathcal{S} \;= \varepsilon_1 + \varepsilon_2 + C && (17)\\ \mathbb{E}\left[(U - Q)^2\right] &= \mathcal{F} && (18) \end{aligned}$$

These can be further decomposed. Assuming we have unbiased estimates $\widehat{\mathcal{L}}$ and use SGD with a constant learning rate η, $\mathcal{L}$ is bounded by a training residual $\widetilde{\mathcal{L}} \geq \mathcal{F}$ and noise floor determined by Monte Carlo (MC) errors δ_1, δ_2. Furthermore, as Q uses an integral, S can be decomposed in terms of a strong error and quadrature error $\varepsilon_1, \varepsilon_2$. In summary,

Quantity	Name
$\mathcal{R}$	Residual
$\mathcal{L}$	Loss (Proxy Residual)
δ_1	Residual MC Error
δ_2	Variance MC Error
$\widetilde{\mathcal{L}}$	Training Residual
$\mathcal{S}$	SDE Approximation Error
ε_1	Strong Error
ε_2	Quadrature Error
$\mathcal{F}$	FK Approximation Error
C	Omitted Cross Terms [2]

Table 1: Terms of the error decomposition.

While it is tempting to focus on the optimisation error of the neural network, failing to address other sources of errors leads to a bottleneck. Indeed, our goal is for $\mathcal{N}_\theta$ to match U. Addressing the errors from different stages in the approximation chain $\widetilde{Q} \to Q \to U$ may have more impact for less effort.

One heuristic is to keep $\mathcal{L}, \mathcal{S}, \mathcal{F}$ the same order. Furthermore, we can consider how the errors and computational cost are controlled by the hyperparameters and minimise error subject to cost.

We begin by considering the errors.

2.1 Loss Function

The loss function is an expectation and we will estimate it with a Monte Carlo average. We gen-

[1] $\mathbb{E}_x\left[\cdot\right]$ conditions on $D = x$ and the expectation is over the randomness of the SDE via the driving brownian motion.

[2] We will not study the cross terms for simplicity and because they decrease with the other sources of errors.

erate N_1 iid initial values $x_i \overset{iid}{\sim} D$ and for each initial value we generate N_2 iid diffusions $(\widehat{X}_T)_{i,j}$ for $\widehat{Q}_j(x_i)$. The estimate $\widehat{\mathcal{L}}$ is formed by averaging the squared residuals $Z_j(x_i)$ of the neural network.

$$\widehat{\mathcal{L}} = \frac{1}{N_1}\sum_{i=1}^{N_1}\frac{1}{N_2}\sum_{j=1}^{N_2} Z_j(x_i) \tag{19}$$

$$Z_j(x_i) = (\widetilde{Q}_j(x_i) - \mathcal{N}_\theta(x_i))^2 \tag{20}$$

2.1.1 Error Analysis

Gradients $G_t = \nabla_\theta \widehat{\mathcal{L}}$ are used to train the neural network. To gain insight on the impact of N_1, N_2 on training with with a constant step size η, we consider Theorem 5.9 in Garrigos and Gower [2023].

We acknowledge the theorem has unrealistic assumptions on the properties of $\mathcal{L}$, but we hope this serves as a starting point for further analysis.

Theorem 2.1. *Let $\mathcal{L}$ be β-smooth and α-strongly convex with the unbiased gradient estimates G_s having bounded L^2-norm. I.e. $\exists M > 0 : \forall s \leq t$,*

$$\mathbb{E}\left[\|G_s - \nabla\mathcal{L}(\theta_s)\|_2^2 \,\middle|\, \theta_s\right] \leq M \tag{21}$$

For a learning rate $\eta \leq 1/\kappa$, SGD converges at a linear rate up to a noise floor ν. Recall that $\kappa = \beta/\alpha$ is known as the condition number of $\mathcal{L}$.

$$\mathbb{E}\left[e_t\right] \leq \nu + (1-\alpha\eta)^t\left[e_0 - \nu\right] \tag{22}$$

$$e_t = \mathcal{L}(\theta_t) - \mathcal{L}(\theta^\star) \tag{23}$$

$$\nu = \frac{\eta M \kappa}{2} \tag{24}$$

Using $G_t = \nabla\widehat{\mathcal{L}}$,

$$\delta = \mathbb{E}\left[\|G - \nabla\mathcal{L}\|_2^2\right] = \sum_k \mathrm{Var}\left(\frac{\partial}{\partial\theta^k}\widehat{\mathcal{L}}\right). \tag{25}$$

By law of total variance and assuming $\widetilde{Q}$ is unbiased, $\delta = \delta_1 + \delta_2$. See Appendix A.

$$\delta_1 = \frac{4}{N_1 N_2^2}\sum_k^{|\theta|} \mathrm{Var}\left(\partial_k \mathcal{N}_\theta(X) e(X)\right) \tag{26}$$

$$\delta_2 = \frac{4}{N_1 N_2}\mathbb{E}\left[\|\nabla_\theta \mathcal{N}_\theta\|_2^2 \,\mathrm{Var}_x\left(\widetilde{Q}(X)\right)\right] \tag{27}$$

This decomposes as a function of the gradients ($\partial_k \mathcal{N}_\theta$ being the partial derivative wrt θ_k), the solution error $e(x) = \mathcal{N}_\theta(x) - U(x)$ and variance of the squared residual $\widetilde{Q}$.

2.2 SDE Approximation Error

Let $\widetilde{X}_t \approx \widehat{X}_t$ be our diffusion approximation and let $I(X)_T$ denote a quadrature scheme which approximates $\int_0^T r(X_t)dt$. $\widetilde{Q} = \psi(\widetilde{X}_T)\exp(I(\widetilde{X})_T)$ and the error $Q - \widetilde{Q}$ can be written as

$$\epsilon_1 \exp\left(\int_0^T r(\widehat{X}_t)dt\right) + \psi(\widetilde{X}_T)(\epsilon_2 + \epsilon_3) \tag{28}$$

where

$$\epsilon_1 = \psi(\widehat{X}_T) - \psi(\widetilde{X}_T)$$

$$\epsilon_2 = \exp\left(\int_0^T r(\widehat{X}_t)dt\right) - \exp\left(I(\widehat{X})_T\right)$$

$$\epsilon_3 = \exp\left(I(\widetilde{X})_T\right) - \exp\left(I(\widehat{X})_T\right)$$

ϵ_1, ϵ_3 correspond to strong error whereas ϵ_2 corresponds to quadrature error. Then we define ϵ_i' to incorporate the factors such that $Q - \widetilde{Q} = \epsilon_1' + \epsilon_2' + \epsilon_3'$

$$\mathcal{S} = \varepsilon_1 + \varepsilon_2 + C \tag{29}$$

$$\varepsilon_1 = \mathbb{E}\left[(\epsilon_1' + \epsilon_3')^2\right] \tag{30}$$

$$\varepsilon_2 = \mathbb{E}\left[(\epsilon_2')^2\right] \tag{31}$$

2.2.1 Error Analysis

In order to simplify the analysis, we assume that ψ, r are bounded by B and in particular their first and second derivatives are bounded by L.

The hyperparameters are step size Δt, the strong error of the SDE scheme m and the order of the quadrature error n. This means that,

$$\mathbb{E}\left[\sup_{[0,T]} |\widehat{X}_t - \widetilde{X}_t|\right] \sim (\Delta t)^m$$

$$\left|\int_0^T r(X_t)dt - I(X)_T\right| \sim TL(\Delta t)^n.^{[3]}$$

We first collect some auxiliary results. First, appealing to the Lipschitz constant of r and order of

[3] Errors for approximation of $\int_a^b f(x)dx$ via Riemann and Simpsons are of the form $M(b-a)(\frac{b-a}{n})^n$ where M is an upper bound on the second derivative of f Libretexts [2021].

strong error of $\widetilde{X}_t$,

$$
\begin{aligned}
|I(\widetilde{X})_T - I(\widehat{X})_T| &\approx \left| \int_0^T r(\widetilde{X}_t) - r(\widehat{X}_t) dt \right| \\
&\leq \int_0^T L|\widetilde{X}_t - \widehat{X}_t| dt \\
&\leq TL(\Delta t)^m.
\end{aligned}
$$

And if $x, y \leq K$ then the restriction of $\exp(x)$ on $(-\infty, K]$ has Lipschitz constant $\exp(K)$. Hence,

$$|\exp(x) - \exp(y)| \leq \exp(K)|x - y|. \tag{32}$$

Putting these together,

$$
\begin{aligned}
\epsilon_1 &\sim L(\Delta t)^m && (33)\\
\epsilon_2 &\sim \exp(BT)TL(\Delta t)^n && (34)\\
\epsilon_3 &\sim \exp(BT)TL(\Delta t)^m. && (35)
\end{aligned}
$$

Considering the multiplicative factors for ϵ_i', defining $B' = B\exp(BT)TL$ and squaring we get

$$
\begin{aligned}
\varepsilon_1 &\sim (B')^2(\Delta t)^{2m} && (36)\\
\varepsilon_2 &\sim (B')^2(\Delta t)^{2n}. && (37)
\end{aligned}
$$

This indicates that we should set $m = n$ to keep the noise the same level. However, a simple Riemann Integration already has order $n = 2$ Libretexts [2021] whereas Euler Maruyama, (Runge Kutta) Milstein and Taylor methods are order 0.5, 1, 1.5 respectively with increasing computational burden Sauer [2013]. Hence, the order of strong error is the bottleneck and it's sufficient to use a simple integration scheme.

2.3 Summary

Aggregating the results,

$$\mathcal{R} = \widetilde{\mathcal{L}} + \eta \left(\frac{A}{N_1 N_2^2} + \frac{B}{N_1 N_2} \right) + (B')^2(\Delta t)^{2m} + \mathcal{F}$$

where $\widetilde{\mathcal{L}} = \mathcal{L}(\theta^\star) + \text{Err}$ is the training residual. (Provided $\mathcal{N}_\theta$ is sufficiently expressive $\mathcal{L}(\theta^\star) \approx \mathcal{F}$). The second term represents the noise floor ν which depends on the learning rate η and N_1, N_2. Lastly, Δt is the step size of the SDE and quadrature schemes; with m being the order of strong error.

As most SDE schemes are variants of Ito-Taylor expansions, the cost of simulating a run is proportional to the number of steps $T/\Delta t$ Sauer [2013]. Letting N_{Training} be the number of training iterations and $\mathcal{N}_\theta^{\text{Eval}}$ be the cost of evaluation and backpropagation of the neural network, the total cost of training is

$$N_{\text{Training}} \left(N_1 N_2 \frac{T}{\Delta T} + \mathcal{N}_\theta^{\text{Eval}} \right) \tag{38}$$

As many of the constants in our analysis are unknown, the hyperparameters will likely be obtained via trial and error. However, we hope that basic insight on their effects on error and cost will lead to more strategic choices.

3 Variance Reduction

A shortcoming of the error decomposition 15 is that $\mathcal{L}, \mathcal{F}$ are bounded below by the variance of $\widetilde{Q}, Q$. Indeed, ignoring the cross terms means we can only make guarantees for $\mathcal{R}$ decreasing to $2\mathcal{F}$. We consider a modification which allows us to decrease $\mathcal{F}$ and imbue utility to the error decomposition.

As an example, we are going to consider a driftless one-dimensional linear filter, see details in Appendix B. Figure 1 shows the stochastic approximation has high variance, but generally the mean will still be close to the true value.

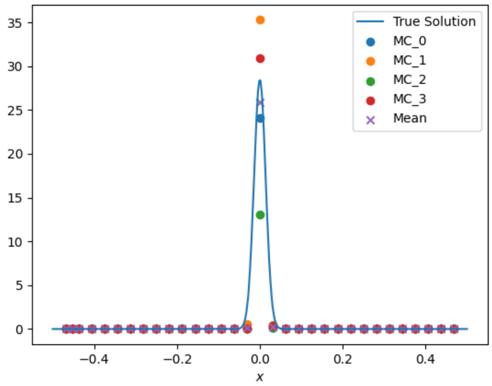


Figure 1: Realisations of stochastic approximations (dots) and their sample mean (cross) plotted against the true mean (blue line) over the domain.

This motivates the use of the sample average of independent stochastic representations Q_{Mod}. It is an unbiased estimate of $U(x)$ with lower variance. See Figure 2 for its effects on the squared residuals.

$$Q_{\text{Mod}}(x) = \frac{1}{N_2} \sum_{j=1}^{N_2} Q_j(x) \tag{39}$$

This is beneficial as it decreases the approximation errors 15 and the associated loss function $\mathcal{L}_{\text{Mod}}$ will be bounded by $\mathcal{F}_{\text{Mod}}$. Effectively, we are decreasing the "approximation floor".

$$\mathcal{S}_{\text{Mod}} = \frac{\mathcal{S}}{N_2}, \quad \mathcal{F}_{\text{Mod}} = \frac{\mathcal{F}}{N_2} \tag{40}$$

The gradient variance of the modified loss function is $\delta_{\text{Mod}} = \delta_{\text{Mod},1} + \delta_{\text{Mod},2}$. See Appendix A.1.

$$\delta_{\text{Mod},1} = \frac{4}{N_1} \sum_{k}^{|\theta|} \text{Var}\left(\partial_k \mathcal{N}_\theta(X) e(X)\right) \tag{41}$$

$$\delta_{\text{Mod},2} = \frac{4}{N_1 N_2^2} \mathbb{E}\left[\|\nabla_\theta \mathcal{N}_\theta\|_2^2 \,\text{Var}_x\left(\widetilde{Q}(X)\right)\right] \tag{42}$$

To access the benefits while keeping cost constant, we would need to increase N_2 and decrease N_1. However, $\delta_{\text{Mod},1}$ will be larger.

Hence, we turn Quasi Monte Carlo (QMC) methods by using a fixed Sobol sequence for $\widehat{X}_0$ Kuo and Nuyens [2016]. This changes the dependence from $N_1 \to N_1^2$, albeit for different constants A, B.

$$\delta_{\text{Mod}} = \frac{A}{N_1^2} + \frac{B}{N_1^2 N_2^2} \tag{43}$$

This leads to biased estimates of the gradients. Fortunately, it can be counteracted by optimising with Adam Kingma and Ba [2017] which performs a normalisation and should nullify the bias.

3.1 Numerical Study

We experimented by varying:

- Training Points - QMC or uniformly random
- Loss Function - Original loss or modified loss

To ensure fairness of comparison, we used the same number of total simulations $N_1 N_2$. I.e. for the original loss we used $N_1 = 2^9, N_2 = 1$ whereas for modified loss we used $N_1 = 2^5, N_2 = 2^4$.

To evaluate these methods, we look at the empirical loss and L^2 residual with the true solution as shown in Figure 3. The empirical loss is noisy and instead we present a rolling moving average. For the L^2 error, the residual decreases steadily and eventually fluctuates. This indicates we have reached a noise floor and require a decreased learning rate.

Concerning L^2 error, we find that using QMC with the original loss marginally increases performance. Whereas QMC is critical for modified loss. Indeed modified loss without QMC has the worst performance and with QMC has the best performance.

For the empirical loss, the original loss does not go to 0 but the L^2 error still decreases to 0. Perhaps, this is because $\mathcal{L}$ and its associated cross terms are the only terms which depends on θ. Hence, provided the cross terms have minimal effect, the gradients should still point in the correct direction.

This highlights a limitation of our decomposition for analysis of the original loss. However, in the case of modified loss, the error decomposition is more relevant due to a lower "approximation floor".

Finally, we note that training with SGD led to extremely poor results. Instead we use Adam with a learning rate of 1e-2. Unfortunately, this means the analysis in section 2.1.1 is not directly applicable as it assumes SGD.

4 Conclusion

In this paper, we have provided an error decomposition for the L^2 error obtained by training on a Feynman-Kac inspired loss function.

This provided insight on the effect of the sources of errors and led to a new loss function. Unfortunately, the analysis is still primitive as it ignores the cross terms and is unable to explain performance under the original loss. Furthermore, we would need to adapt our analysis to account for Adam and how it mitigates against the bias of the QMC estimates. However, we hope this can serve as a starting point for further analysis.

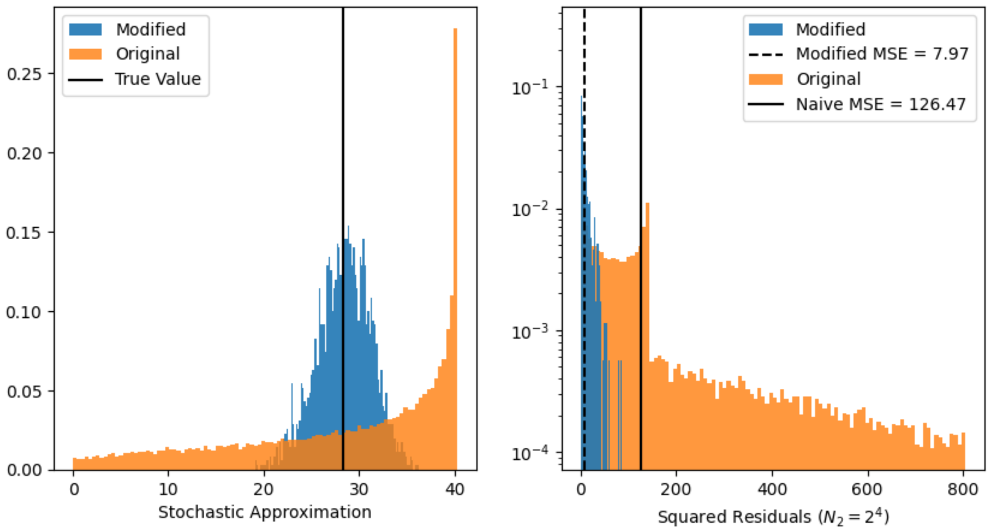


Figure 2: Distribution of residuals of modified and original loss. In the right plot, the black vertical lines correspond to the approximation floors of each method with modified loss' being significantly lower.

A Gradient Variance

Our goal is to prove Equation 26. Firstly, by iid

$$\mathrm{Var}\left(\partial_k \widehat{\mathcal{L}}\right) = \frac{1}{N_1} \mathrm{Var}\left(\frac{1}{N_2}\sum_{j=1}^{N_2} \partial_k Z_j(X)\right). \tag{44}$$

$\mathrm{Var}\,(Y) = \mathrm{Var}\,(\mathbb{E}_x\,[Y]) + \mathbb{E}\,[\mathrm{Var}_x\,(Y)]$ by the law of total variance, hence we consider

$$\mathbb{E}_x\left[\frac{1}{N_2}\sum_{j=1}^{N_2} \partial_k Z_j(x)\right] = \frac{\mathbb{E}_x\,[\partial_k Z(x)]}{N_2} \tag{45}$$

$$\mathrm{Var}_x\left(\frac{1}{N_2}\sum_{j=1}^{N_2} \partial_k Z_j(x)\right) = \frac{\mathrm{Var}_x\,(\partial_k Z(x))}{N_2}. \tag{46}$$

By the chain rule,

$$\partial_x Z(x) = \partial_k (\mathcal{N}_\theta(x) - \widetilde{Q}(x))^2 \tag{47}$$

$$= 2(\partial_k \mathcal{N}_\theta(x))(\mathcal{N}_\theta(x) - \widetilde{Q}(x)). \tag{48}$$

Assuming that $\widetilde{Q}$ is unbiased for U and letting $e(x) = \mathcal{N}_\theta(x) - U(x)$, we deduce

$$\mathbb{E}_x\,[\partial_k Z(x)] = 2(\partial_k \mathcal{N}_\theta(x)) e(x) \tag{49}$$

$$\mathrm{Var}_x\,(\partial_k Z(x)) = 4(\partial_k \mathcal{N}_\theta(x))^2\, \mathrm{Var}_x\left(\widetilde{Q}(x)\right). \tag{50}$$

Computing the variance and summing over leads to the desired result.

A.1 Modified Loss

In equation 26, we set $N_2 = 1$. Then replace $\widetilde{Q}$ with a sample mean of size N_2, which leads to a factor of N_2^{-2} in δ_2.

B Simulation Details

The one-dimensional linear filtering problem is

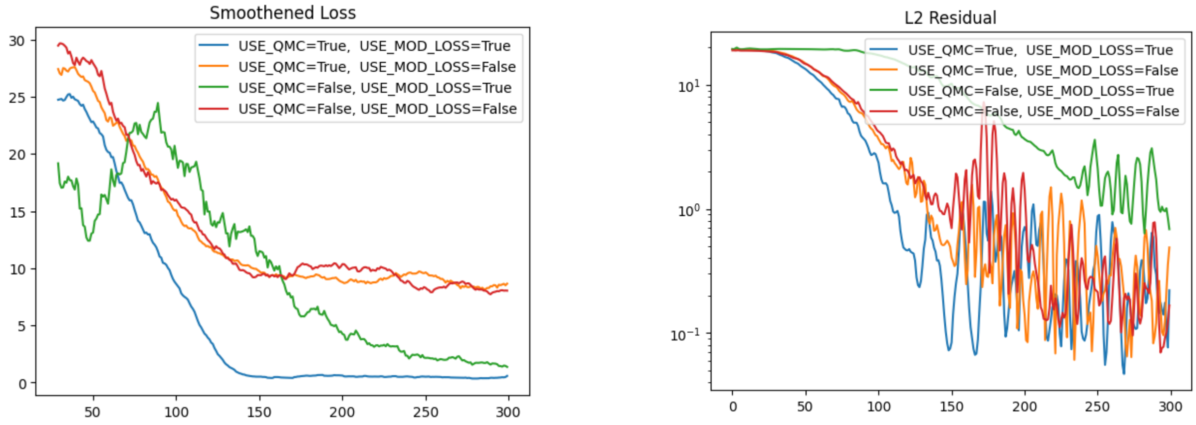
$$f(x) = Mx, \quad \sigma(x) = \Sigma, \quad h(x) = Hx \tag{51}$$

for which $b(x) = -Mx, r(x) = M$. In this case, we have an explicit formula for the auxiliary diffusion

$$\widehat{X}_t = \exp(-Mt)\widehat{X}_0 + \int_0^t \exp(-M(t-s))dW_s$$
$$\sim \mathcal{N}\left(\exp(-Mt)\widehat{X}_0, \frac{1-\exp(-2MT)}{M}\right).$$

The stochastic representation is given by

$$q(t,x) = \psi(\widehat{X}_t)\exp(-Mt). \tag{52}$$



(a) Rolling average of window size 30.

(b) L^2 Error with exact solution.

Figure 3: Smoothened training loss and L^2 residual plotted for a neural network trained on different types of loss functions. We compare using QMC Sobol training points and the modified loss function.

This means $\mathcal{S} = 0$ as we do not have to approximate the diffusion or compute an integral. The initial condition ψ is a gaussian pdf with mean μ_0 and std Σ_0. For the experiments we use values in Table 2.

M	Σ	H	μ_0	Σ_0	T
-1	0.1	90	0	0.01	0.01

Table 2: Experiment parameters

The neural network architecture, Figure 4, will be modification of Crisan et al. [2022]. The key changes are removal of batch normalisation (which slows training) and a softplus layer to enforce positivity instead of regularisation. To train this neural network we used Adam with a learning rate of 1e-2.

```
nn.Sequential(
    # Layer 1
    nn.Linear(dim, 51, bias=True),
    nn.Tanh(),
    # Layer 2
    nn.Linear(51, 51, bias=True),
    nn.Tanh(),
    # Part 3 (get scalar output)
    nn.Linear(51, 1, bias=True),
    # Ensure positive output
    nn.Softplus(),
)
```

Figure 4: Neural Network Architecture in PyTorch

References

Dan Crisan, Alexander Lobbe, and Salvador Ortiz-Latorre. An application of the splitting-up method for the computation of a neural network representation for the solution for the filtering equations, January 2022.

Guillaume Garrigos and Robert M Gower. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv preprint arXiv:2301.11235*, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017.

Frances Y Kuo and Dirk Nuyens. A practical guide to quasi-monte carlo methods, 2016.

Libretexts. 2.5: Numerical integration - midpoint, trapezoid, simpson's rule, Jul 2021. URL https://math.libretexts.org/Courses/Mount_Royal_University/MATH_2200%3A_Calculus_for_Scientists_II/2%3A_Techniques_of_Integration/2.5%3A_Numerical_Integration_-_Midpoint%2C_Trapezoid%2C_Simpson\OT1\textquoterights_rule.

Timothy Sauer. Computational solution of stochastic differential equations. *WIREs Computational Statistics*, 5(5):362–371, September 2013. ISSN 1939-5108, 1939-0068. doi: 10.1002/wics.1272.