UNIVERSITY OF TORONTO | Engineering

Application of Multi Layer Monte Carlo in Solving Partial Differential Equations for Bond Options Pricing

April 2022

Eric Li
Engineering Science (Machine Intelligence)
University of Toronto

Table of Contents

Introduction

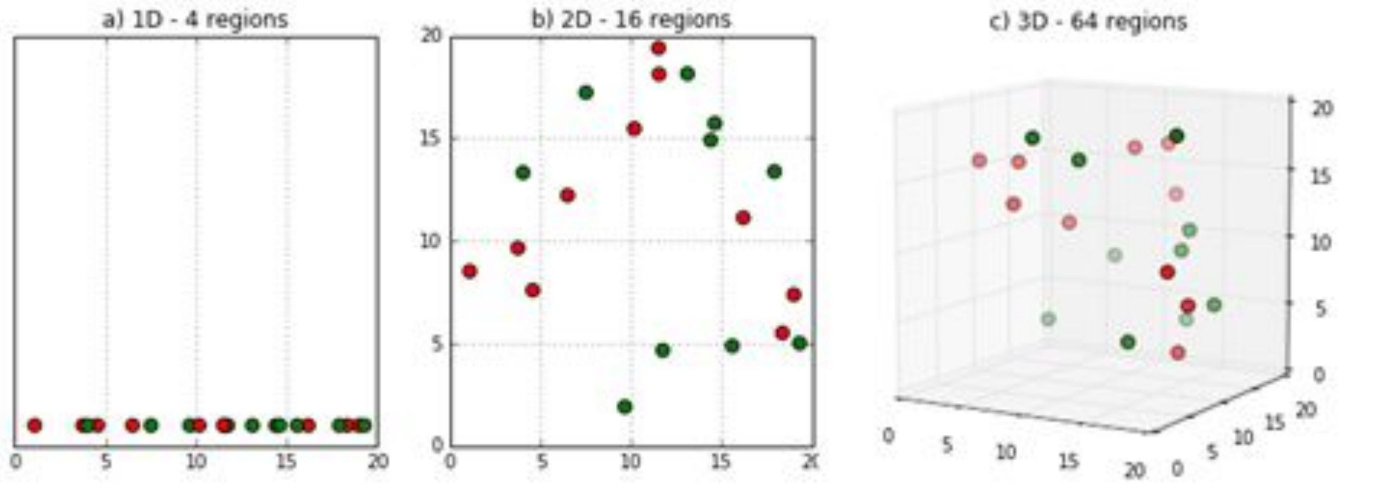
- Motivation
- Curse of Dimensionality

Motivation: Curse of Dimensionality (CoD)

Partial Differential Equations (PDEs): Black Scholes Barenblatt (BSB)

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

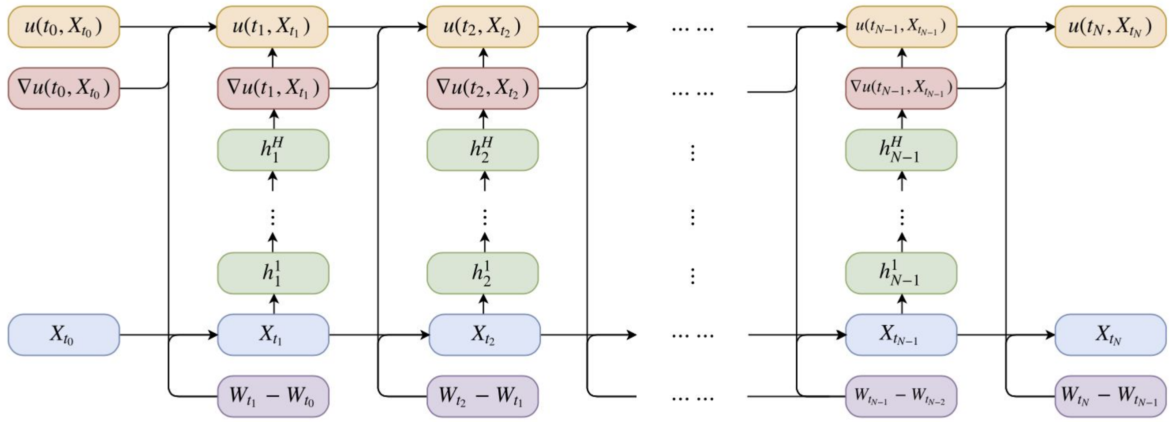
Curse of Dimensionality: More data required due to greater sparsity at larger dimensions



DeepAI, *Data Sparsity for Data Points*. .

Motivation: Time Efficiency and PDE Simplicity

<u>Deep Neural Network</u>: Solution for BSB CoD



<u>Limitation:</u> Time Inefficiency from Monte Carlo Sampling

<u>Limitation:</u> Model Simplicity

Fixed Interest Rate

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

<u>Goal:</u>

1) Increase Path Simulation Time Efficiency
2) Extend PDE Complexity for more realistic real world application

Weinan E, Jiequn Han, and Arnulf Jentzen. "Deep Learning-Based Numerical Methods for High-Dimensional Parabolic Partial Differential Equations and Backward Stochastic Differential Equations". In: Communications in Mathematics and Statistics 5.4 (Nov. 2017), pp. 349–380. issn: 2194-671X.doi:10.1007/s40304- 017- 0117- 6 .url: http://dx.doi.org/10.1007/s40304-017-0117-6

Background

- Forward Backward Stochastic Neural Networks (FBSNN) for High Dimensional PDEs
- Multilayer Monte Carlo (MLMC) Path Simulation

FBSNN: Initial Formulations

Quasi-linear PDEs

$$u_t = f(t, x, u, Du, D^2u)$$

Unknown solution: $u(t, x)$

Terminal Condition: $u(T, x) = g(x)$

Initial Coupled Forward Backward Stochastic Differential Equations (FBSDE) of general form:

$$\begin{aligned} dX_t &= \mu(t, X_t, Y_t, Z_t)dt + \sigma(t, X_t, Y_t)dW_t, \quad t \in [0, T], \\ X_0 &= \xi, \\ dY_t &= \varphi(t, X_t, Y_t, Z_t)dt + Z_t'\sigma(t, X_t, Y_t)dW_t, \quad t \in [0, T), \\ Y_T &= g(X_T), \end{aligned}$$

Solutions consist of stochastic processes X_T, Y_T, Z_T

Ito's Formula

$$Y_t = u(t, X_t), \text{ and } Z_t = Du(t, X_t).$$

Physics Informed Deep Learning

Automatic Differentiation

Approximate with a DNN $u(t, x)$

Maziar Raissi. Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations. 2018. arXiv:1804.07010 [stat.ML].

FBSNN: Modelling with DNN

Goal: Approximate *u(x,t)* with DNN

1. Discretize FBSDE with Euler-Maruyama scheme

$$X^{n+1} \approx X^n + \mu(t^n, X^n, Y^n, Z^n)\Delta t^n + \sigma(t^n, X^n, Y^n)\Delta W^n,$$
$$Y^{n+1} \approx Y^n + \varphi(t^n, X^n, Y^n, Z^n)\Delta t^n + (Z^n)'\sigma(t^n, X^n, Y^n)\Delta W^n,$$

2. Define Loss Function

$$\sum_{m=1}^{M}\sum_{n=0}^{N-1}|Y_m^{n+1} - Y_m^n - \Phi_m^n \Delta t^n - (Z_m^n)'\Sigma_m^n \Delta W_m^n|^2 + \sum_{m=1}^{M}|Y_m^N - g(X_m^N)|^2,$$

N = Number of Timesteps   M = Number of Path Simulations

FBSNN: Financial Applications

FBSD

$$dX_t = \sigma \mathrm{diag}(X_t) dW_t, \quad t \in [0, T],$$
$$X_0 = \xi,$$
$$dY_t = r(Y_t - Z_t' X_t)dt + \sigma Z_t' \mathrm{diag}(X_t) dW_t, \quad t \in [0, T),$$
$$Y_T = g(X_T), \;\; g(x) \;=\; \|x\|^2.$$

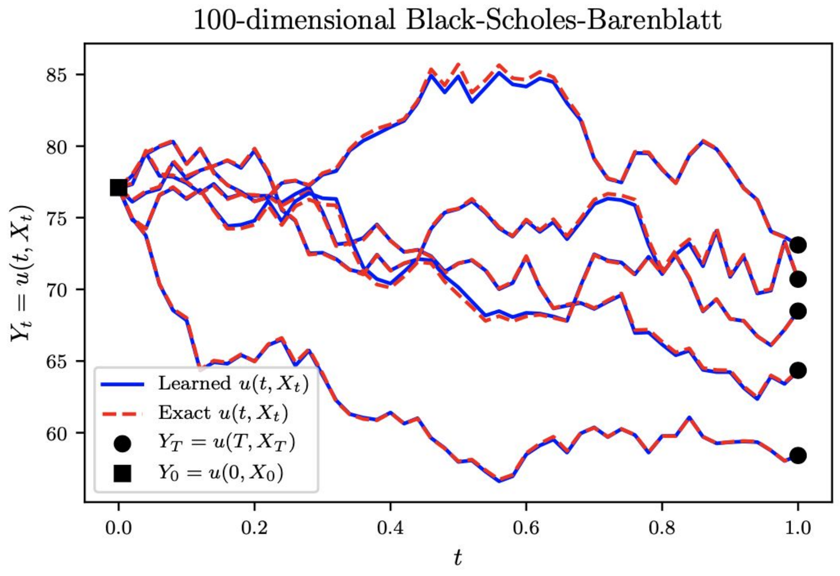
Black Scholes Barenblatt (BSB) PDE

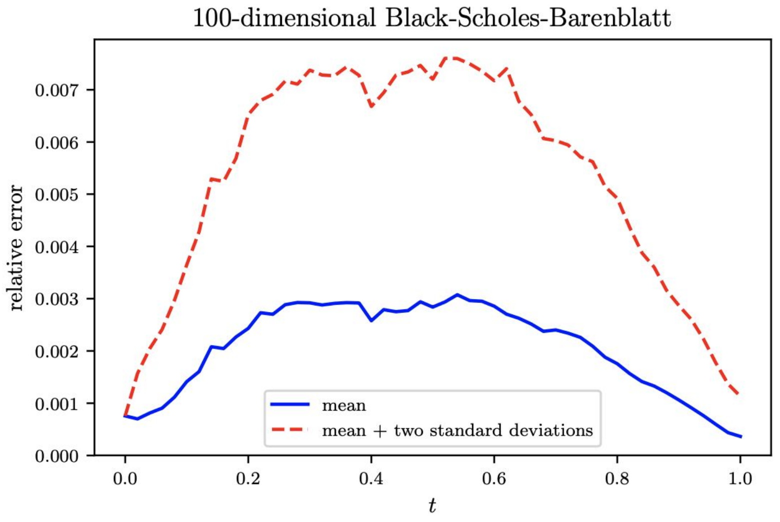
$$u_t = -\frac{1}{2}\mathrm{Tr}[\sigma^2 \mathrm{diag}(X_t^2) D^2 u] + r(u - (Du)'x)$$

Known Explicit Solution

$$u(t,x) = \exp\left((r+\sigma^2)(T-t)\right) g(x)$$

- X_t : Underlying Stock price at time t
- Y_t : Overlying derivative price at time t
- σ : Scalar Volatility
- r : Scalar Interest Rate
- **Monte Carlo Path Sampling** for data generation





FBSNN: Advancements and Limitations

Advancements

- Model parameters do not increase by number of timesteps (N)
- Converges to the exact value $Y_0 = u(0,X_0)$ in first few hundred iterations
- After 500 iterations, relative error of 2.3×10^{-3} is obtained

Limitations

- Monte Carlo sampling is costly
- More time steps required to more accurately estimate $Y_t = u(t,X_t)$ for $t > 0$
- Only basic PDEs modelled; BSB equations for calls and puts are more complex
- Only quasi-linear parabolic PDEs can be modelled and solved

MLMC: An Extension of Previous MC Method

- Previous MC method repeatedly iterates over a constant amount of points in time interval *[0,T]*
- MLMC introduces layers of different time steps covering different iterations

$0 < t < T$ $\quad 0 < l < L$ $\quad h_l = M^{-l}T$

$M = 2,\ T = 1$

$l = 1$, $h_l = 0.5$: $t_1 = 0.5$, N_1

$l = 2$, $h_l = 0.25$: $t_1 = 0.25$, $t_2 = 0.5$, $t_3 = 0.75$, N_2

$l = 3$, $h_l = 0.125$: N_3

...

T: Terminal Time
L: Terminal Layer
h_l: time step size at layer l
M: number of time step increase per layer
ε: relative error

Iterations at each level

$$N_l = \left\lceil 2\,\epsilon^{-2}\sqrt{V_l\,h_l}\,\left(\sum_{l=0}^{L}\sqrt{V_l/h_l}\right)\right\rceil$$

Convergence Condition $\left|\widehat{Y}_L - M^{-1}\widehat{Y}_{L-1}\right| < \frac{1}{\sqrt{2}}\,(M^2-1)\,\epsilon$

MLMC: Initial Paper Results

Theory

- To achieve an accuracy of $O(\varepsilon)$ in sampling , the computation cost is reduced from $O(\varepsilon^{-3})$ of MC to $O(\varepsilon^{-2}(\log\varepsilon)^2)$ of MLMC

Application on Option Pricing

- Time decrease of a factor in the range of 30-65 in comparison with MC or 4-10 in comparison to MC + performance boosting methods (ie: Richardson extrapolation)

Limitations

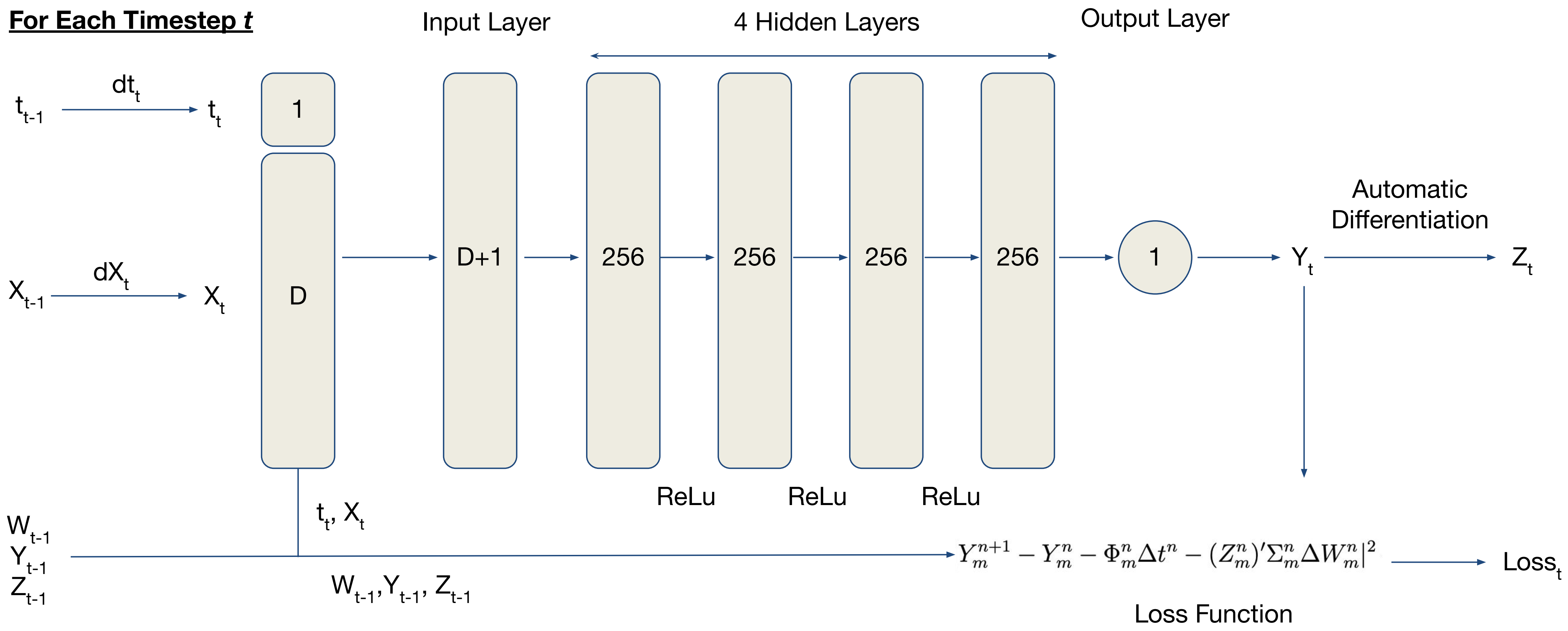
- Lack of complex finance applications (ie: high dimensionality) during testing
- Lack of significant improvement when combined with Richardson Extrapolation; other performance boosting methods (ie: Quasi Monte Carlo, Milstein Discretization) must be explored
- Unknown benefits when used for neural network data generation
- Potential problems arising due to bias from Monte Carlo path simulation is still present

Methods and Results

- Comparing MLMC and MC
- Exploring MLMC Hyperparameters
- Bond Options (Stochastic Interest Rates)

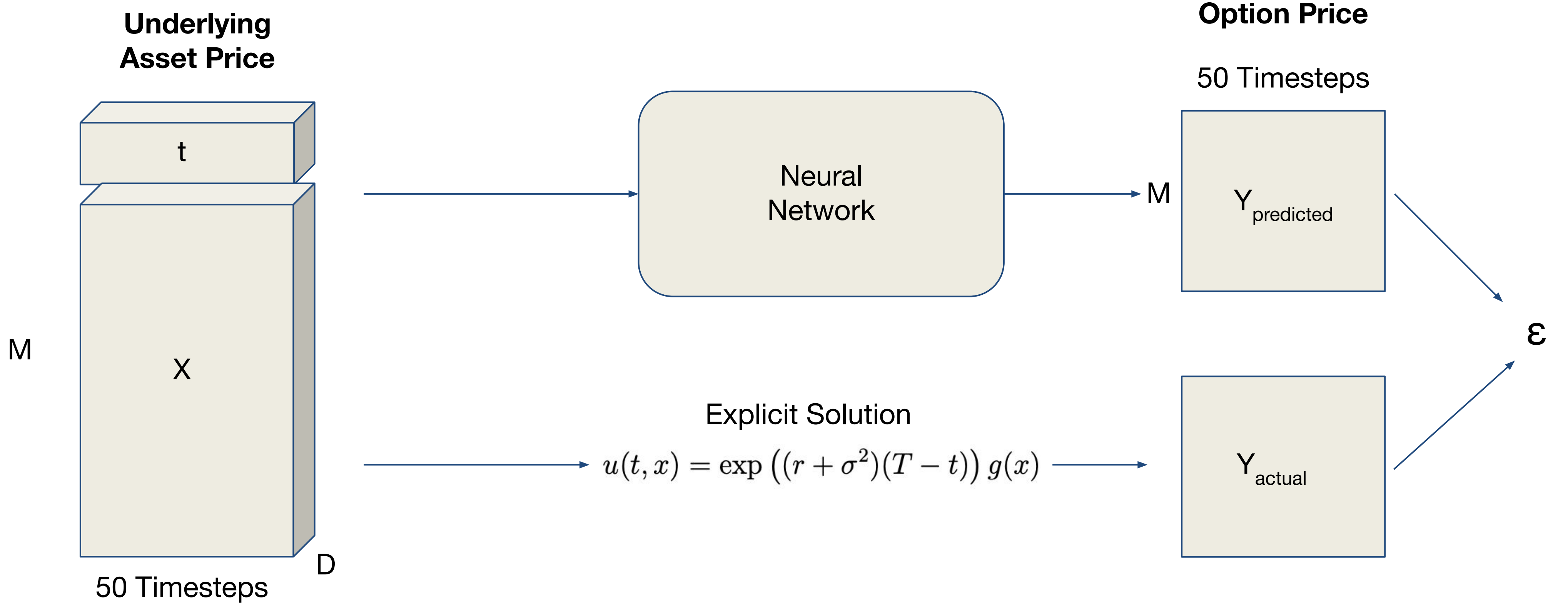
Methods: Comparing MLMC and MC (Training)

Performed across all M batches over N timesteps for each training iteration



Methods: Comparing MLMC and MC (Validation)

<u>For Each Iteration Until ε=User Specified Error</u>



Relative Error = $\varepsilon = |Y_{actual} - Y_{predicted}| / Y_{actual}$

Results: Comparing MLMC and MC

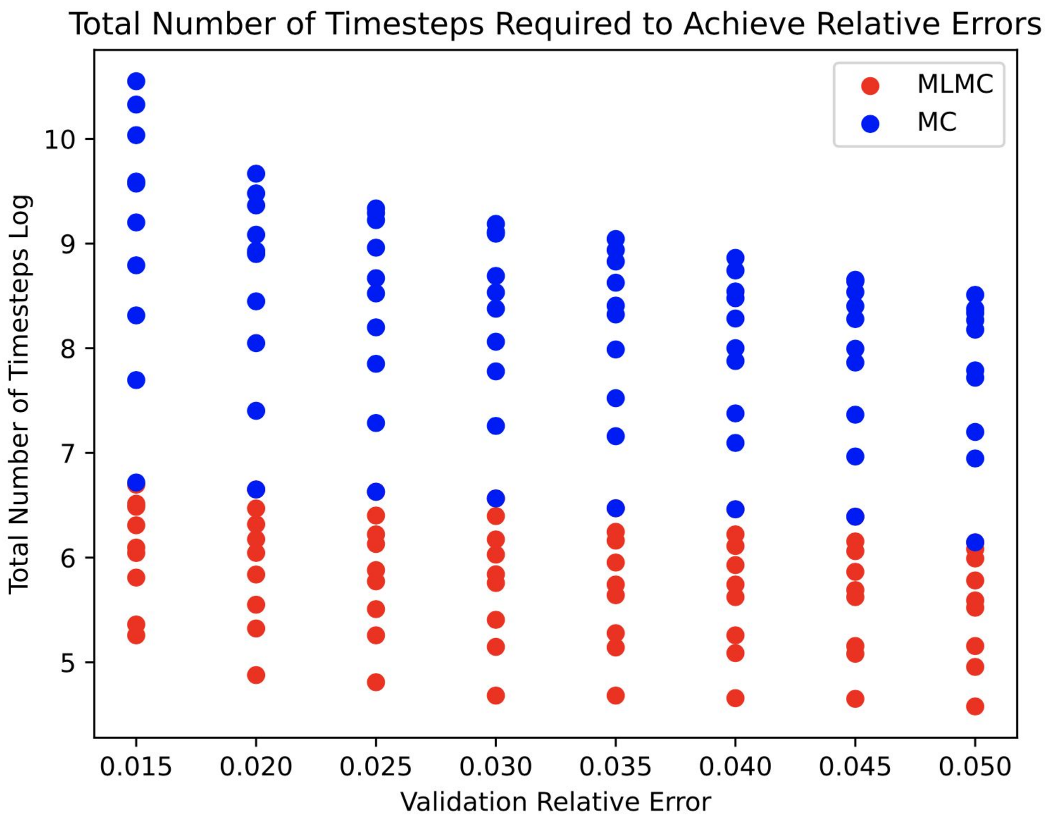


Figure 1
Best MC model requires 5.1-7.2x more timesteps than best MLMC model

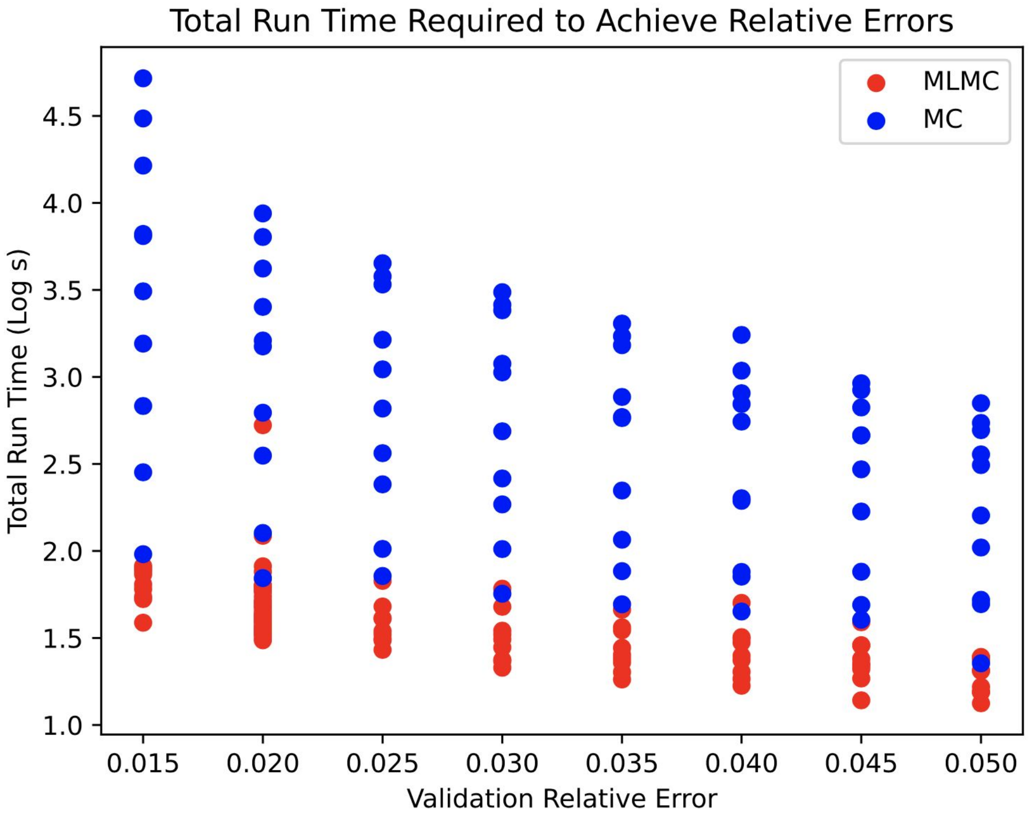
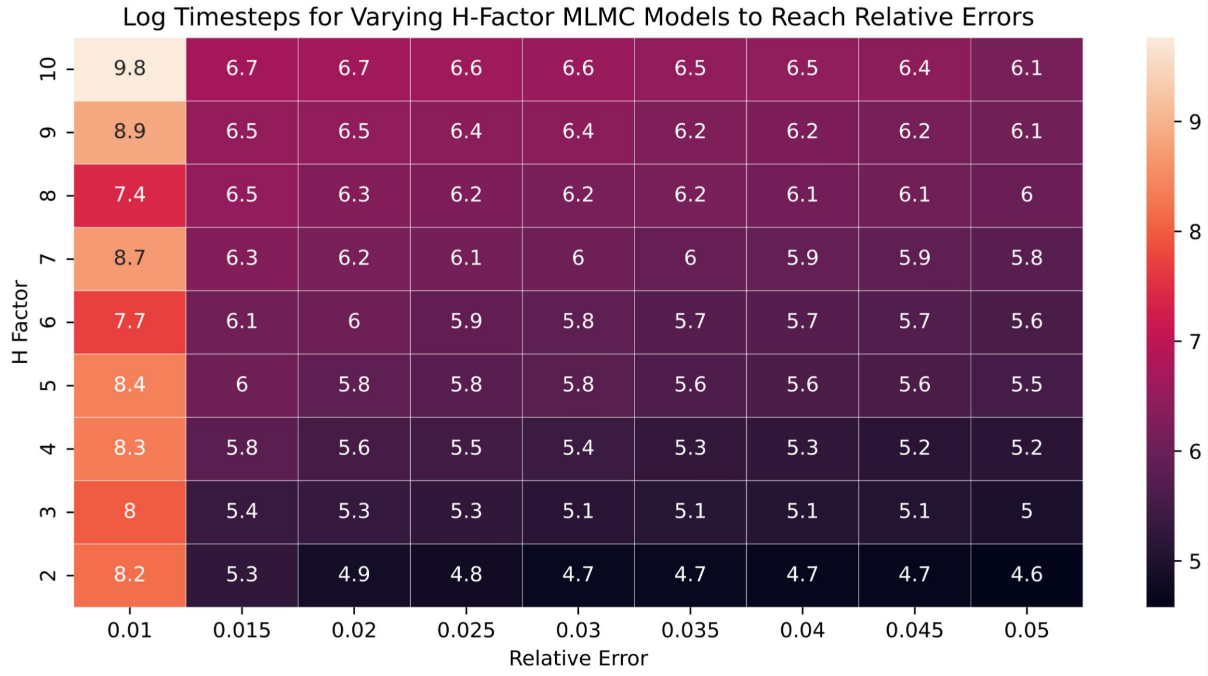
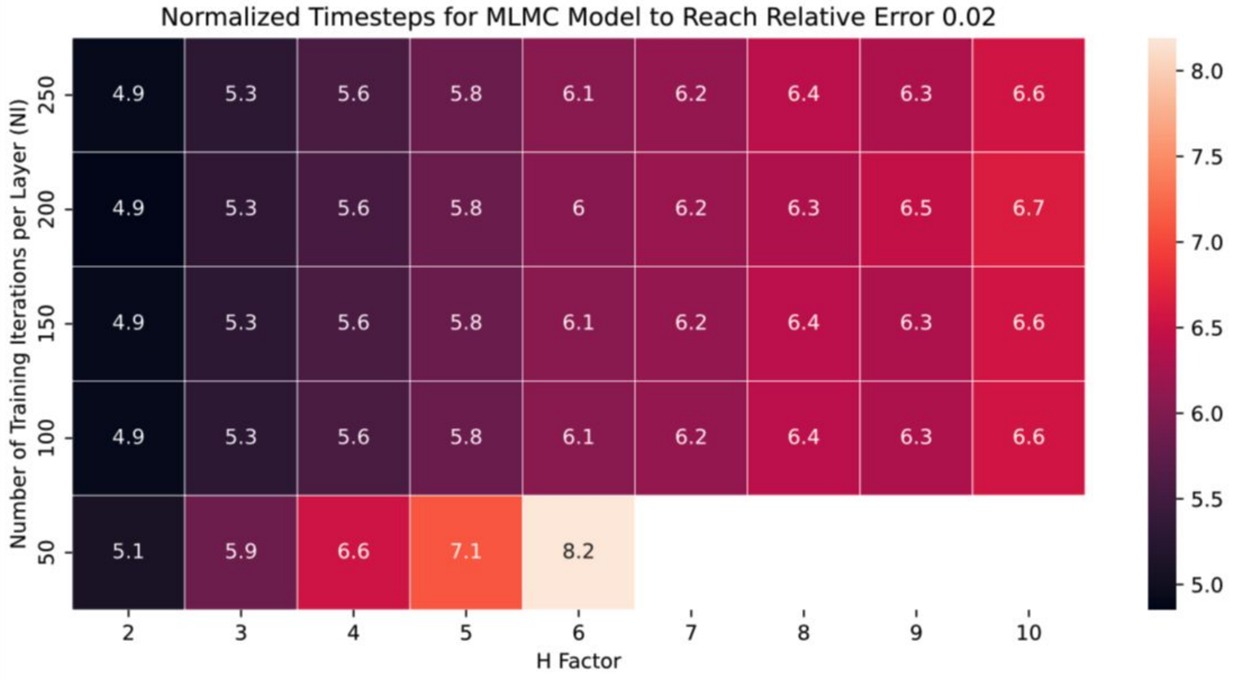


Figure 2
Best MC model requires 1.4-1.6x more time than the best MLMC model

Results: MLMC at Different Hyperparameters





<u>Figure 3</u>
$N_l = 200$
Time steps increase as Relative Error decreases and H Factor increases

<u>Figure 4</u>
Relative Error = 0.02
Time steps increase as H Factor increases, but seem independent of NI

Methods: Bond Pricing (Stochastic Interest Rates)

Equation (1)

$$\begin{cases} \mathrm{d}r_t = \alpha(\theta_t - r_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t^1, \\ \mathrm{d}\theta_t = \beta(\phi - \theta_t)\,\mathrm{d}t + \eta\,\mathrm{d}W_t^2, \end{cases}$$

Equation (2)

$$P_t(T) = \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_t^T r_u\,\mathrm{d}u} P_T(T) \middle| \mathcal{F}_t\right] = \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_t^T r_u\,\mathrm{d}u} \middle| \mathcal{F}_t\right]$$

Equation (3)

$$\mathrm{d}X_t = \mu(t, X_t)\,\mathrm{d}t + \sigma(t, X_t)\,\mathrm{d}B_t$$

$$X_t = (r_t, \theta_t)$$

$$\mu(t, x, y) = \begin{bmatrix} \alpha(y - x) \\ \beta(\phi - y) \end{bmatrix}$$

$$\sigma(t, x, y) = \begin{bmatrix} \sigma & 0 \\ 0 & \eta \end{bmatrix}$$

Feynman-Kac Theorem

Equation (4)

$$P_t(T) = \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_t^T r_u\,\mathrm{d}u} \middle| \mathcal{F}_t\right] = u(t, r_t, \theta_t),$$

Methods: Bond Pricing (Stochastic Interest Rates)

$$P_t(T) = \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_t^T r_u \,\mathrm{d}u} \middle| \mathcal{F}_t\right] = u(t, r_t, \theta_t)$$

PDE

$$\begin{cases} \dfrac{\partial u}{\partial t} + \alpha(y-x)\dfrac{\partial u}{\partial x} + \beta(\phi - y)\dfrac{\partial u}{\partial y} + \dfrac{1}{2}\sigma^2\dfrac{\partial^2 u}{\partial x^2} + \dfrac{1}{2}\eta^2\dfrac{\partial^2 u}{\partial y^2} = xu \\ u(T,x,y) = 1. \end{cases}$$

SDE

$$P_t(T) = e^{A_t(T) - B_t(T) r_t - C_t(T)\theta_t}$$

A_t(T), B_t(T), C_t(T) are known functions

Euler Maruyama Discretization
MLMC Path Sampling

Neural Network Model

After Investigation of Bond Pricing:

- Pricing Bond Options
- Pricing Swaptions

Conclusion and Next Steps

- Comparing MLMC and MC
- Exploring MLMC Hyperparameters
- Bond Options (Stochastic Interest Rates)

Conclusion

Key Results

- MLMC demonstrates significant time complexity improvement over MC with an improvement of 5.1-7.2x less time steps and 1.4-1.6x less overall training time
- Number of required time steps increases as Relative Error decreases and H-Factor increases
- MLMC and DNN's performance with Stochastic Interest Rate can be investigated through applying Feynman-Kac to Zero Coupon Bonds

Next Steps

- Explore MLMC and DNN's with complex Stochastic Interest Rates PDEs at high dimensions for Zero Coupon Bonds, Bond Options, and Swaptions
- Explore the optimization of MLMC hyperparameters for minimizing number of time steps required

Thank you.

ericchang.li@mail.utoronto.ca