Hybrid option pricing through AI and GPU-powered SDEs solvers

Tomasz Bochacik†, Natalia Czyżewska†, Andrzej Kałuża†, Dawid Majchrowski†,
Paweł Morkisz†, ‡, Paweł Przybyłowicz†, Marcelina Studzińska-Wrona†
† Faculty of Applied Mathematics, AGH University of Science and Technology, Krakow, Poland, ‡NVIDIA Poland, Warsaw

Introduction

Financial derivatives pricing is an important field that requires both high precision and substantial computational power. The classical approach relies on Monte Carlo simulations and numerical methods for underlying stochastic differential equations. Another emerging approach, not used in production systems yet, is employing machine learning for this task.
The proposed solution is the hybrid model combining appropriate methodology for performing fast Monte Carlo simulations on GPUs with application of DNNs to approximating prices. The comprehensive discussion of the problem, including SOTA, accuracy and performance analysis, will be presented in the next sections.

Problem and approach

The intuition is as follows: some derivatives, e.g. an Asian option, depend on the future trajectory of the real financial process (e.g. stock price for a chosen company). It is assumed that the financial process belongs to a class of stochastic processes that can be represented by a stochastic differential equation (SDE). Assuming that and having the historical values of the real financial process, one is able to estimate the coefficients of the SDE, which in turn allows to simulate the process' future values by application of some numerical scheme. Since the future values of the derivative are random, a single simulation would result in a different number at each repetition. Hence,the future values are approximated through generating many trajectories by some numerical scheme. Finally, based on the generated approximations, the derivative's (e.g. Asian option's) price can be estimated, as well as its empirical distribution.

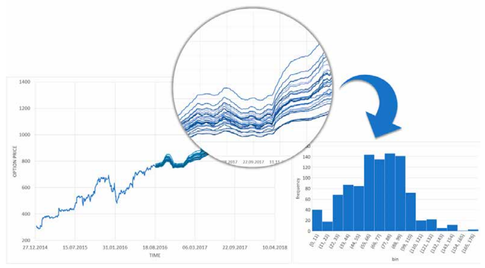

Fig. 1: Illustration of the main phases of option pricing.

Summarizing, the problem of option pricing can be divided into the following phases (cf. Fig. 1):

A) Modelling the underlying stochastic process (e.g. stock price), i.e. deriving a stochastic differential equation whose solution will be a good approximation of the real process.

B) Estimating the parameters of the model.

C) Simulating multiple trajectories of the SDE's solution (this step includes choosing a proper numerical scheme for approximating SDE's solution).

D) Estimating the price of the derivative, together with its probability distribution.

For the phase A the most popular models would be Black Scholes, Heston, Merton or their variations with jumps or including stochastic volatility, cf. [1]. It turns out that the most computationally demanding are phases B, C, and D. Fortunately, each of them can be considerably accelerated by using GPUs. The phase D is commonly realized as the standard Monte Carlo simulation.

New numerical methods and their GPU implementation

The theoretical analysis of Euler and Milstein methods has been performed for various precision levels, both in a classical formulation and in the jump-diffusion case, cf. [2, 3, 4, 5, 6]. The efficient GPU package based on these results is under development. The current progress is documented under the link: http://home.agh.edu.pl/~akaluza/cuSTOCH/. The initial results show that **NVIDIA Titan V allows to obtain about 14x speed-up in relation to Intel Broadwell** for simple implementations. This is due to both **acceleration of random numbers generators** and **ease of paralleling computations for multiple trajectories**. Currently, several well known schemes for solving SDEs (also with jumps) have been implemented, i.e. Euler and Randomized Euler schemes, Milstein scheme and its derivative-free and Adaptive Step Size Control versions, cf. [3].

Numerical experiments

The GPU implementation of Milstein derivative free method was used for the following test problem (called the Merton's model):

$$\begin{cases} \mathrm{d}X(t) = (r - c_0\lambda)X(t)\mathrm{d}t + \sigma X(t)\mathrm{d}W(t) + c_0 X(t-)\mathrm{d}N(t), \\ \qquad t \in [0, T], \\ X(0) = 100, \end{cases}$$

where W is the Wiener process, while N is the Poisson process. Processes with jumps are commonly used to model financial time series, see e.g. [1]. Speed-up was estimated through several tests using different numbers of trajectories. The results displayed in the Fig. 2 confirm the advantage of GPU implementation over the CPU in terms of time efficiency. The speed-up is even more spectacular when the process is modelled in a higher-dimensional space, i.e. when a single drift coefficient is replaced by a vector of drift coefficients and (in general) the diffusion coefficients form a matrix.

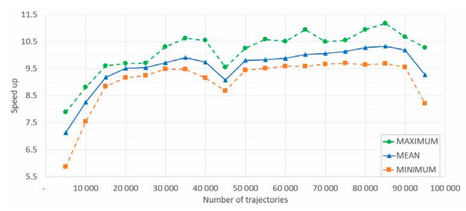

Fig. 2: GPU vs CPU speedup - Titan V vs Intel Broadwell.

Approximation with various precisions

Following the increasing popularity of computations with low precision for significant speedup, a suitable analytic noise model is proposed. It corresponds to standard noisy information about a, b, X and W (in SDEs with jumps also about c and N). This allows to estimate upfront the needed precision and mesh size to obtain predefined approximation error level. A perturbed processes $Y \in \{X, W, N\}$ can be represented as

$$\tilde{Y}(t, \omega) = Y(t, \omega) + \delta \cdot p_Y(t, Y(t, \omega)).$$

This model of noise assumes analytical structure of the perturbed processes. As far as we know the articles [2, 4, 5] were first to investigate the problem of Monte Carlo simulations, based on perturbed data about the driving stochastic processes or coefficients of the underlying equation.

Deep Learning models application

There is a work in progress on application of various time-series-based deep learning models to financial processes, specifically to forecasting their future behavior from the historical data. The general procedure is to train the model on multiple historical points of the trajectory with the sliding window approach. That is, based on past and current values of the process, possibly together with estimated parameters such as the current drift and diffusion coefficients, the model is forecasting just one next value. This approach guarantees sufficiently rigid training data set as in financial applications the most common are time series with long history.
Tests were conducted using Tensorflow Keras. The network A architecture was just two layers of LSTM (first with 128 output units, second with 32 output units) and a single dense layer with an output being the forecast entry. Overall, network A depends on about 88k parameters.

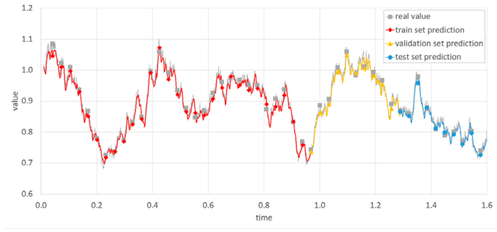

Fig. 3: Forecasting example for a model SDE solution.

The Fig. 3 shows that the model has a high predictive power We observe proper forecasting behavior of the algorithm, even though the model is random to some extent.
Additionally, the following architectures were chosen for further testing:

B) CNN. The smallest analyzed architecture consisting of the convolution layer with 64 output filters followed by max pooling layer with the size of the pool window equal 2 (about 1k parameters).

C) Combination of CNN and RNN. The first layer is the same convolution as in the network B, then at the top there is one stacked layer of the LSTM with 128 hidden units (about 100k parameters).

D) Sequence-to-sequence RNN based on the encoder-decoder architecture. The encoder part of the network (one layer LSTM with 128 hidden units) is used to calculate an encoder vector, which in turn is used as an initial hidden state of the decoder part, which has the same structure as the encoder in this case (about 200k parameters).

E) Dual-Stage Attention-Based RNN. Roughly speaking, the encoder-decoder architecture with the attention mechanism for both parts, which once again consist of one layer LSTM with 128 hidden units (about 200k parameters).

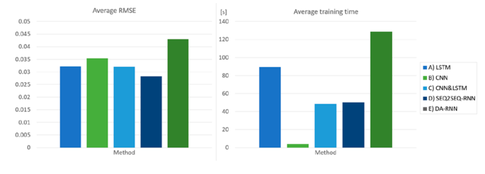

Fig. 4: Models comparison.

DNN models summary

The Root Mean Squared Error (RMSE) has been chosen as an error metric. In Fig. 4 errors for 50 epochs training and for 112 different trajectories are summed up. The conclusion is that all the tested models perform very well in terms of both predictive power and level errors. The tested networks did not allow to precisely determine the impact of architecture and size of a model on the error.

Hybrid models

Adding an extra AI layer on a top of the results obtained from the classic Monte Carlo simulations and from the Deep Learning engine enables to perform fast training of the model when the initially considered financial instrument is replaced by another one. Precisely, the underlying methodology for the SDE's solution approximation as well as the Deep Learning engine might remain unchanged, whereas the new derivative or the new financial process to be analyzed can be input and the calibration of the entire model can be performed efficiently. For this purpose it is planned to intertwine GANs with efficient SDEs solvers. According to the authors' best knowledge, such models have been, thus far, neither a subject of scientific research nor a basis for derivative pricing in production.

Future research

Firstly, it is planned to include the computational model of floating point arithmetic for the considered numerical schemes. Secondly, the further development of the cuSTOCH package is planned. In particular, the aim is to optimize the code and to provide more general API. At the same time possibility of applying new DNNs architectures for time series will be investigated. Finally, the hybrid models will be constructed and tested in practice.

Acknowledgements

This research was supported by the Polish Ministry of Higher Education through grant "Najlepsi z najlepszych! 4.0" and was realized as a part of joint research project between AGH and NVIDIA.

References

[1] Monique Jeanblanc, Marc Yor, and Marc Chesney. "Mathematical Methods for Financial Markets". In: *Springer* (2009).

[2] Andrzej Kałuża, Paweł M. Morkisz, and Paweł Przybyłowicz. "Optimal approximation of stochastic integrals in analytic noise model". In: *Applied Mathematics and Computation* 356 (2019), pp. 74–91.

[3] Andrzej Kałuża and Paweł Przybyłowicz. "Optimal global approximation of jump-diffusion SDEs via path-independent step-size control". In: *Applied Numerical Mathematics* 128 (2018), pp. 24–42.

[4] Paweł M. Morkisz and Paweł Przybyłowicz. "Optimal pointwise approximation of SDE's from inexact information". In: *Journal of Computational and Applied Mathematics* 324 (2017), pp. 85–100.

[5] Paweł M. Morkisz and Paweł Przybyłowicz. "Randomized derivative-free Milstein algorithm for efficient approximation of solutions of SDEs under noisy information". In: (2019). arXiv: https://arxiv.org/abs/1912.06865.

[6] Paweł Przybyłowicz. "Efficient approximate solution of jump-diffusion SDEs via path-dependent adaptive step-size control". In: *Journal of Computational and Applied Mathematics* 350 (2019), pp. 396–411.