Deep Gradient Flow Methods for Option Pricing in Diffusion Models

Jasper Rou[†,*] Antonis Papapantoleon[†] Emmannuil Georgoulis[‡§]

[*]j.g.rou@tudelft.nl, [†]Delft Institute of Applied Mathematics, TU Delft, [‡]Department of Mathematics, Heriot-Watt University, [§]Department of Mathematics, National Technical University of Athens

Introduction

In the pricing of options two things are important: speed and accuracy. Unfortunately, these two do not go hand in hand. Simple models like the Black-Scholes model provide a solution fast, but are not very accurate. More complicated models like the lifted Heston model [1] are accurate but computation can take quite long. Using neural networks is a promising method to compute the option price fast in complicated models.

One way of pricing an option is to write its value as the solution to a partial differential equation (PDE) with the Feynman-Kac formula:

$$\begin{aligned}\frac{\partial u}{\partial t}+\sum_{i,j=0}^{n} a^{ij}\frac{\partial^2 u}{\partial x_i \partial x_j}-\sum_{i=0}^{n} b^i\frac{\partial u}{\partial x_i}-ru=0,\\ u(T)=\Phi(S_T)\end{aligned} \tag{1}$$

In this research we will solve this general PDE using a neural network.

Method

The Deep Galerkin Method [4] is to minimize

$$\left\|\frac{\partial u}{\partial t}+\sum_{i,j=0}^{n} a^{ij}\frac{\partial^2 u}{\partial x_i \partial x_j}-\sum_{i=0}^{n} b^i\frac{\partial u}{\partial x_i}-ru\right\|^2_{[0,T]\times\Omega}+\|u(T)-\Phi(S_T)\|^2_{\Omega}.$$

To apply the Time Deep Nitsche Method [3], we rewrite PDE (1) by splitting the operator into two parts: a symmetric part and an asymmetric part:

$$\begin{aligned}\frac{\partial u}{\partial t}&=-\sum_{i,j=0}^{n}\frac{\partial}{\partial x_j}\left(a^{ij}\frac{\partial u}{\partial x_i}\right)+\sum_{i=0}^{n}\left(b^i+\sum_{j=0}^{n}\frac{\partial a^{ij}}{\partial x_j}\right)\frac{\partial u}{\partial x_i}+ru.\\ &=-\nabla\cdot(A\nabla u)+ru+F(u),\\ F(u)&=\mathbf{b}\cdot\nabla u.\end{aligned}$$

We then divide $[0,T]$ in intervals $(\tau_{k-1},\tau_k]$ with $h=\tau_k-\tau_{k-1}$ and seek approximations $f^k(\theta;\mathbf{x})$ such that

$$\frac{f^k-f^{k-1}}{h}-\nabla\cdot\left(A\nabla f^k\right)+rf^k+F\left(f^{k-1}\right)=0.$$

This is equivalent to finding the minimizer of

$$L=\frac{1}{2}\left\|w-f^{k-1}\right\|^2_{L^2(\Omega)}+h\int_{\Omega}\frac{1}{2}\left((\nabla w)^T A\nabla w+rw^2\right)+F\left(f^{k-1}\right)wdx.$$

Algorithm 1 Time Deep Nitsche Method

Initialize network parameters θ_0^0.

Initialize a neural network approximating the initial condition

$$f^0=\min_{w\in H^1(\Omega)}\|w-\Phi(\mathbf{X})\|_{L_2(\Omega)}.$$

for each time step $k=1,\dots,N_t$ **do**
    Initialize $\theta_0^k=\theta^{k-1}$.
    for each sampling stage **do**
        Generate random points $\mathbf{x}^i$ for training.
        Calculate the cost functional $L(\theta_n^k;\mathbf{x}^i)$ for the sampled points.
        Take a descent step $\theta_{n+1}^k=\theta_n^k-\alpha_n\nabla_\theta L(\theta_n^k;\mathbf{x}^i)$.
    end for
end for

Results

We compare the prices computed by the neural networks with deriving the characteristic function and from that computing the option price using the COS method [2].

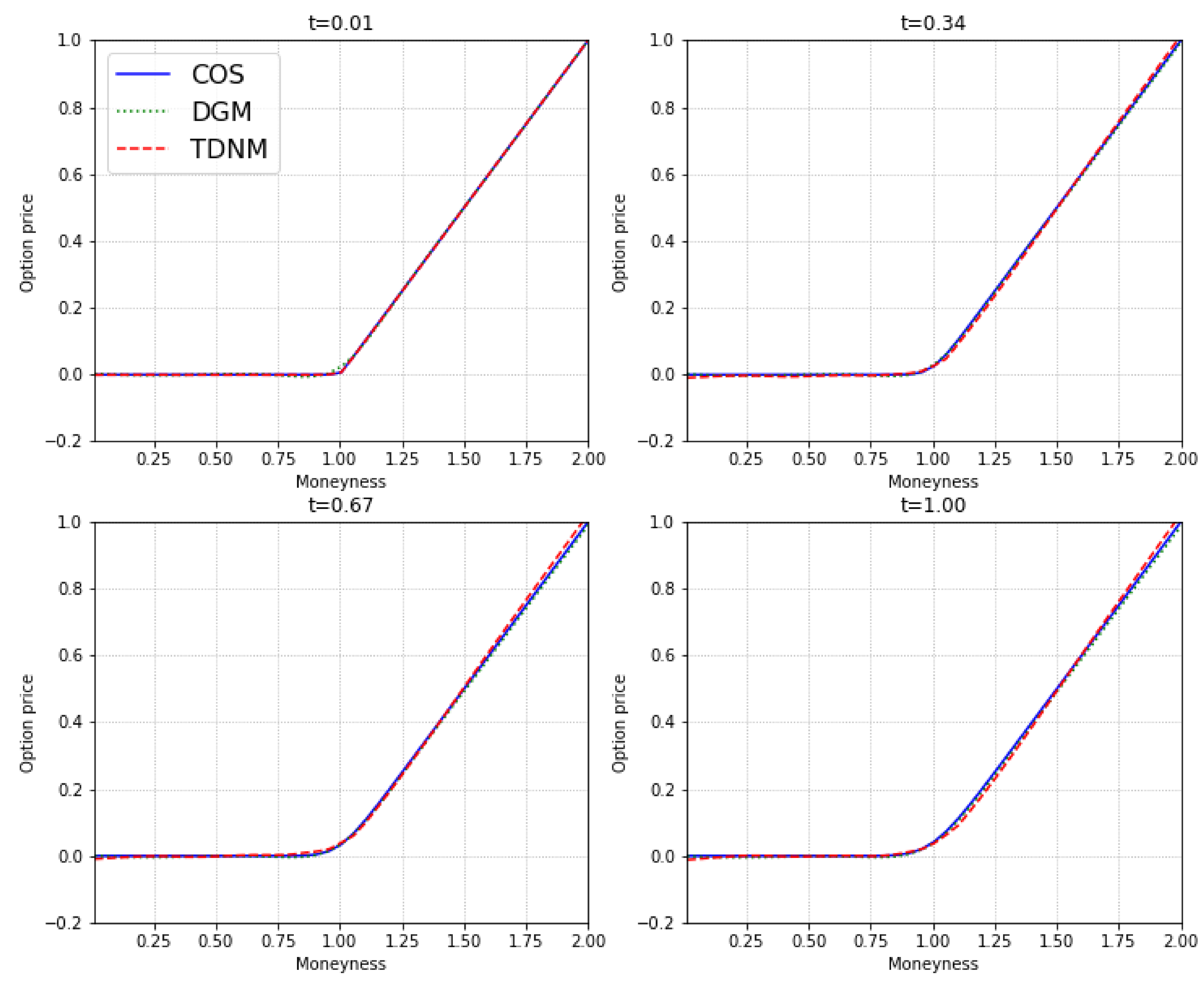

Figure 1: European call option prices in the lifted Heston model with 21 dimensions against the moneyness for four different times to maturity.

Model	BS	Heston	LH, n=1	LH, n=5	LH, n=20
DGM	3.4×10^3	6.3×10^3	2.0×10^4	4.4×10^4	1.6×10^5
TDNM	5.5×10^3	7.0×10^3	7.6×10^3	1.2×10^4	1.9×10^4

Table 1: Training time in seconds of the different methods for a European call option in different models.

Model	BS	Heston	LH, n=1	LH, n=5	LH, n=20
COS	5.0×10^{-4}	1.3×10^{-2}	5.7×10^{-0}	6.2×10^{-0}	6.4×10^{-0}
DGM	1.1×10^{-2}	1.1×10^{-2}	1.1×10^{-2}	1.1×10^{-2}	1.1×10^{-2}
TDNM	2.7×10^{-2}	3.6×10^{-2}	5.3×10^{-2}	5.2×10^{-2}	4.5×10^{-2}

Table 2: Computing time in seconds of the different methods for a European call option in different models.

References

[1] Eduardo Abi Jaber. "Lifting the Heston model". In: *Quantitative Finance* 19.12 (2019), pp. 1995–2013.

[2] Fang Fang and Cornelis W. Oosterlee. "A novel pricing method for European options based on Fourier-cosine series expansions". In: *SIAM Journal on Scientific Computing* 31.2 (2009), pp. 826–848.

[3] Emmanuil H Georgoulis, Michail Loulakis, and Asterios Tsiourvas. "Discrete gradient flow approximations of high dimensional evolution partial differential equations via deep neural networks". In: *Communications in Nonlinear Science and Numerical Simulation* 117 (2023), p. 106893.

[4] Justin Sirignano and Konstantinos Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations". In: *Journal of computational physics* 375 (2018), pp. 1339–1364.