Deep Gradient Flow Methods for Option Pricing in Diffusion Models

Finance Research Day

Jasper Rou

December 15, 2023

Joint work with Emmanuil Georgoulis & Antonis Papapantoleon

Pricing

Price of a derivative with pay-off $\Phi(S_T)$

$$u(t) = \mathbb{E}\left[e^{-r(T-t)}\Phi(S_T)|S_t\right]$$

Pricing

Price of a derivative with pay-off $\Phi(S_T)$

$$u(t) = \mathbb{E}\left[e^{-r(T-t)}\Phi(S_T)|S_t\right]$$

Feynman-Kac formula:

$$\frac{\partial u}{\partial t} + \sum_{i,j=0}^{n} a^{ij}\frac{\partial^2 u}{\partial x_i \partial x_j} - \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} - ru = 0,$$

$$u(T) = \Phi(S_T)$$

Pricing

Price of a derivative with pay-off $\Phi(S_T)$

$$u(t) = \mathbb{E}\left[e^{-r(T-t)}\Phi(S_T)|S_t\right]$$

Feynman-Kac formula:

$$\frac{\partial u}{\partial t} + \sum_{i,j=0}^{n} a^{ij}\frac{\partial^2 u}{\partial x_i \partial x_j} - \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} - ru = 0,$$

$$u(T) = \Phi(S_T)$$

Can we solve this PDE using a neural network?

Deep Galerkin Method [1]

$$\frac{\partial u}{\partial t} + \sum_{i,j=0}^{n} a^{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} - \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} - ru = 0,$$

$$u(T) = \Phi(S_T)$$

Deep Galerkin Method [1]

$$\frac{\partial u}{\partial t} + \sum_{i,j=0}^{n} a^{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} - \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} - ru = 0,$$

$$u(T) = \Phi(S_T)$$

Minimize

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

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

Deep Galerkin Method [1]

$$\frac{\partial u}{\partial t} + \sum_{i,j=0}^{n} a^{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} - \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} - ru = 0,$$

$$u(T) = \Phi(S_T)$$

Minimize

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

Issue: Taking second derivative makes training in high dimensions slow

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

Idea

Rewrite PDE as energy minimization problem

Idea

Rewrite PDE as energy minimization problem

- Only first order derivative
- No norm

Idea

Rewrite PDE as energy minimization problem

- Only first order derivative
- No norm

Split in symmetric and non-symmetric part

Splitting method

$$\frac{\partial u}{\partial t} = -\sum_{i,j=0}^{n} a^{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} + ru$$

Splitting method

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= -\sum_{i,j=0}^{n} a^{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} + ru \\
&= -\sum_{i,j=0}^{n} \frac{\partial}{\partial x_j} \left(a^{ij} \frac{\partial u}{\partial x_i} \right) + \sum_{i,j=0}^{n} \frac{\partial a^{ij}}{\partial x_j} \frac{\partial u}{\partial x_i} + \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} + ru
\end{aligned}
$$

Splitting method

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= -\sum_{i,j=0}^{n} a^{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} + ru \\
&= -\sum_{i,j=0}^{n} \frac{\partial}{\partial x_j} \left(a^{ij} \frac{\partial u}{\partial x_i} \right) + \sum_{i,j=0}^{n} \frac{\partial a^{ij}}{\partial x_j} \frac{\partial u}{\partial x_i} + \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} + ru \\
&= -\sum_{i,j=0}^{n} \frac{\partial}{\partial x_j} \left(a^{ij} \frac{\partial u}{\partial x_i} \right) + \sum_{i=0}^{n} \left(b^i + \sum_{j=0}^{n} \frac{\partial a^{ij}}{\partial x_j} \right) \frac{\partial u}{\partial x_i} + ru.
\end{aligned}
$$

Splitting method

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= -\sum_{i,j=0}^{n} a^{ij}\frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} + ru \\
&= -\sum_{i,j=0}^{n} \frac{\partial}{\partial x_j}\left(a^{ij}\frac{\partial u}{\partial x_i}\right) + \sum_{i,j=0}^{n} \frac{\partial a^{ij}}{\partial x_j}\frac{\partial u}{\partial x_i} + \sum_{i=0}^{n} b^i \frac{\partial u}{\partial x_i} + ru \\
&= -\sum_{i,j=0}^{n} \frac{\partial}{\partial x_j}\left(a^{ij}\frac{\partial u}{\partial x_i}\right) + \sum_{i=0}^{n}\left(b^i + \sum_{j=0}^{n}\frac{\partial a^{ij}}{\partial x_j}\right)\frac{\partial u}{\partial x_i} + ru. \\
&= -\nabla\cdot(A\nabla u) + ru + F(u), \\
F(u) &= \mathbf{b}\cdot\nabla u.
\end{aligned}
$$

Example: Heston model

$$
\begin{aligned}
dS_t &= rS_t dt + \sqrt{V_t} S_t dW_t, \qquad && S_0 > 0, \\
dV_t &= \kappa(\theta - V_t) dt + \eta \sqrt{V_t} dB_t, \qquad && V_0 > 0.
\end{aligned}
$$

Example: Heston model

$$
\begin{aligned}
dS_t &= rS_t dt + \sqrt{V_t} S_t dW_t, & S_0 > 0, \\
dV_t &= \kappa(\theta - V_t)dt + \eta\sqrt{V_t} dB_t, & V_0 > 0.
\end{aligned}
$$

$$
\frac{\partial u}{\partial t} = -rS\frac{\partial u}{\partial S} - \kappa(\theta - V)\frac{\partial u}{\partial V} - \frac{1}{2}S^2 V\frac{\partial^2 u}{\partial S^2} - \frac{1}{2}\eta^2 V\frac{\partial^2 u}{\partial V^2} - \rho\eta SV\frac{\partial^2 u}{\partial S\partial V} + ru
$$

Example: Heston model

$$\frac{\partial u}{\partial t} = -rS\frac{\partial u}{\partial S} - \kappa(\theta - V)\frac{\partial u}{\partial V} - \frac{1}{2}S^2 V\frac{\partial^2 u}{\partial S^2} - \frac{1}{2}\eta^2 V\frac{\partial^2 u}{\partial V^2} - \rho\eta SV\frac{\partial^2 u}{\partial S\partial V} + ru$$

Example: Heston model

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= -rS\frac{\partial u}{\partial S} - \kappa(\theta - V)\frac{\partial u}{\partial V} - \frac{1}{2}S^2V\frac{\partial^2 u}{\partial S^2} - \frac{1}{2}\eta^2V\frac{\partial^2 u}{\partial V^2} - \rho\eta SV\frac{\partial^2 u}{\partial S\partial V} + ru \\
&= -rS\frac{\partial u}{\partial S} - \kappa(\theta - V)\frac{\partial u}{\partial V} - \frac{\partial}{\partial S}\left(\frac{1}{2}S^2V\frac{\partial u}{\partial S}\right) + SV\frac{\partial u}{\partial S} \\
&\quad - \frac{\partial}{\partial V}\left(\frac{1}{2}\eta^2V\frac{\partial u}{\partial V}\right) + \frac{1}{2}\eta^2\frac{\partial u}{\partial V} - \frac{\partial}{\partial S}\left(\frac{1}{2}\rho\eta SV\frac{\partial u}{\partial V}\right) + \frac{1}{2}\rho\eta V\frac{\partial u}{\partial V} \\
&\quad - \frac{\partial}{\partial V}\left(\frac{1}{2}\rho\eta SV\frac{\partial u}{\partial S}\right) + \frac{1}{2}\rho\eta S\frac{\partial u}{\partial S} + ru
\end{aligned}
$$

Example: Heston model

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= -rS\frac{\partial u}{\partial S} - \kappa(\theta - V)\frac{\partial u}{\partial V} - \frac{1}{2}S^2 V\frac{\partial^2 u}{\partial S^2} - \frac{1}{2}\eta^2 V\frac{\partial^2 u}{\partial V^2} - \rho\eta SV\frac{\partial^2 u}{\partial S \partial V} + ru \\
&= -rS\frac{\partial u}{\partial S} - \kappa(\theta - V)\frac{\partial u}{\partial V} - \frac{\partial}{\partial S}\left(\frac{1}{2}S^2 V\frac{\partial u}{\partial S}\right) + SV\frac{\partial u}{\partial S} \\
&\quad - \frac{\partial}{\partial V}\left(\frac{1}{2}\eta^2 V\frac{\partial u}{\partial V}\right) + \frac{1}{2}\eta^2\frac{\partial u}{\partial V} - \frac{\partial}{\partial S}\left(\frac{1}{2}\rho\eta SV\frac{\partial u}{\partial V}\right) + \frac{1}{2}\rho\eta V\frac{\partial u}{\partial V} \\
&\quad - \frac{\partial}{\partial V}\left(\frac{1}{2}\rho\eta SV\frac{\partial u}{\partial S}\right) + \frac{1}{2}\rho\eta S\frac{\partial u}{\partial S} + ru \\
&= -\nabla\cdot\left(\frac{V}{2}\begin{bmatrix} S^2 & \eta\rho S \\ \eta\rho S & \eta^2 \end{bmatrix}\nabla u\right) + \begin{bmatrix} (V - r + \frac{1}{2}\rho\eta)S \\ \kappa(V-\theta) + \frac{1}{2}\eta\rho V + \frac{\eta^2}{2} \end{bmatrix}\cdot\nabla u + ru
\end{aligned}
$$

Time Deep Nitsche Method [2]

$$\begin{cases} u_\tau - \nabla \cdot (A \nabla u) + ru + F(u) = 0, & (\tau, \mathbf{x}) \in [0, T] \times \Omega, \\ u(0, \mathbf{x}) = \Phi(\mathbf{x}), & \mathbf{x} \in \Omega. \end{cases}$$

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

Time Deep Nitsche Method [2]

$$\begin{cases} u_\tau - \nabla \cdot (A\nabla u) + ru + F(u) = 0, & (\tau, \mathbf{x}) \in [0, T] \times \Omega, \\ u(0, \mathbf{x}) = \Phi(\mathbf{x}), & \mathbf{x} \in \Omega. \end{cases}$$

- Divide $[0, T]$ in intervals $(\tau_{k-1}, \tau_k]$ with $h = \tau_k - \tau_{k-1}$

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

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

Time Deep **Nitsche** Method

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

Time Deep **Nitsche** Method

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

$$0 = \int_\Omega \left(\left(U^k - U^{k-1} \right) + h \left(-\nabla \cdot \left(A \nabla U^k \right) + rU^k + F\left(U^{k-1} \right) \right) \right) v d\mathbf{x}$$

Time Deep **Nitsche** Method

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

$$\begin{aligned} 0 &= \int_\Omega \left(\left(U^k - U^{k-1}\right) + h\left(-\nabla \cdot \left(A \nabla U^k\right) + rU^k + F\left(U^{k-1}\right)\right)\right) v d\mathbf{x} \\ &= i'(0) \end{aligned}$$

$$i(\tau) = I^k(U^k + \tau v)$$

Time Deep **Nitsche** Method

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

$$\begin{aligned} 0 &= \int_\Omega \left(\left(U^k - U^{k-1}\right) + h\left(-\nabla \cdot \left(A \nabla U^k\right) + rU^k + F\left(U^{k-1}\right)\right)\right) v d\mathbf{x} \\ &= i'(0) \end{aligned}$$

$$i(\tau) = I^k(U^k + \tau v)$$

$$\begin{aligned} I^k(u) &= \frac{1}{2}\left\|u - U^{k-1}\right\|^2 + h \int_\Omega \frac{1}{2}\left((\nabla u)^T A \nabla u + ru^2\right) + F\left(U^{k-1}\right) u dx \\ U^k &= \operatorname*{arg\,min}_{u \in H^1(\Omega)} I^k(u) \end{aligned}$$

Time **Deep** Nitsche Method

$$I^k(u) = \frac{1}{2}\left\|u - U^{k-1}\right\|^2 + h\int_\Omega \frac{1}{2}\left((\nabla u)^T A\nabla u + ru^2\right) + F\left(U^{k-1}\right) udx$$

$$U^k = \underset{u\in H^1(\Omega)}{\arg\min}\, I^k(u)$$

Time **Deep** Nitsche Method

$$
\begin{aligned}
I^k(u) &= \frac{1}{2}\left\|u - U^{k-1}\right\|^2 + h\int_\Omega \frac{1}{2}\left((\nabla u)^T A\nabla u + ru^2\right) + F\left(U^{k-1}\right) udx \\
U^k &= \underset{u\in H^1(\Omega)}{\arg\min}\, I^k(u) \\
f^k(\theta) &= \underset{u\in \mathcal{C}(\theta)}{\arg\min}\, I^k(u) \\
\mathcal{C}(\theta) &= \text{space of neural networks with parameters } \theta
\end{aligned}
$$

Algorithm

1: Initialize θ_0.

Algorithm

1: Initialize θ_0.
2: **for** each time step $k = 1, ..., N_t$ **do**
3: Initialize $\theta_0^k = \theta^{k-1}$.

Algorithm

Initialize θ_0.
for each time step $k = 1, ..., N_t$ **do**
  Initialize $\theta_0^k = \theta^{k-1}$.
  for each sampling stage n **do**
    Generate random points $\mathbf{x}^i$ for training.

Algorithm

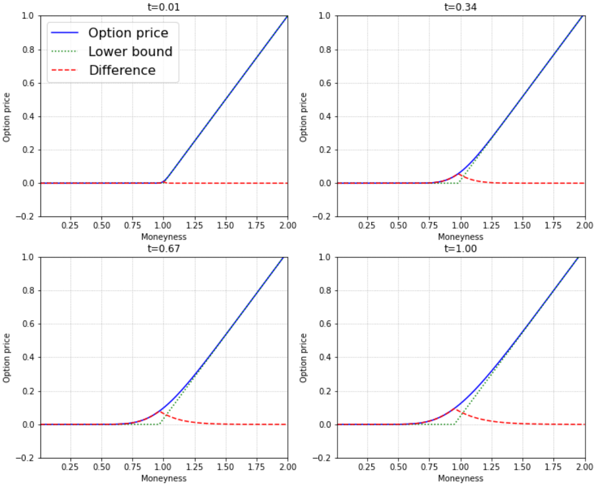
Initialize θ_0.
for each time step $k = 1, ..., N_t$ **do**
    Initialize $\theta_0^k = \theta^{k-1}$.
    for each sampling stage n **do**
        Generate random points $\mathbf{x}^i$ for training.
        Calculate the cost functional $I^k(f(\theta_n^k; \mathbf{x}^i))$.

Algorithm

Initialize θ_0.
for each time step $k = 1, ..., N_t$ **do**
    Initialize $\theta_0^k = \theta^{k-1}$.
    for each sampling stage n **do**
        Generate random points $\mathbf{x}^i$ for training.
        Calculate the cost functional $I^k(f(\theta_n^k; \mathbf{x}^i))$.
        Take a descent step $\theta_{n+1}^k = \theta_n^k - \alpha_n \nabla_\theta I^k(f(\theta_n^k; \mathbf{x}^i))$.
    end for
end for

Base

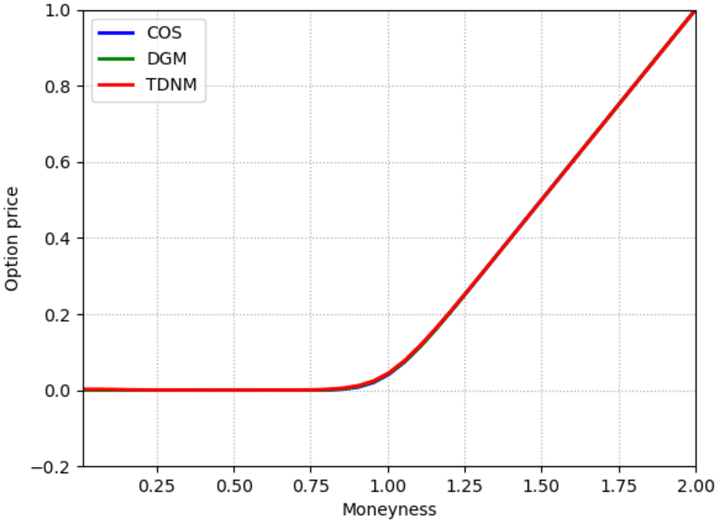
No-arbitrage bound: $u(t,S) \geq S_t - Ke^{-rt}$



Architecture

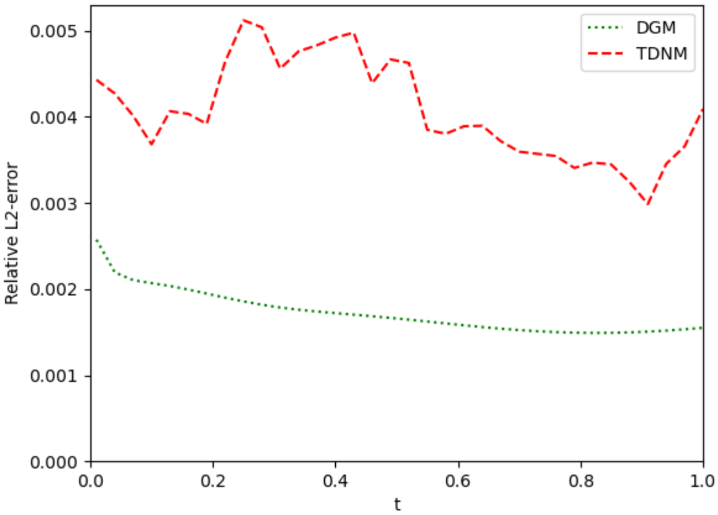
$$
\begin{aligned}
S^1 &= \sigma_1\left(W^1\mathbf{x} + b^1\right), && \\
Z^l &= \sigma_1\left(U^{z,l}\mathbf{x} + W^{z,l}S^l + b^{z,l}\right), && l = 1, \dots, L, \\
G^l &= \sigma_1\left(U^{g,l}\mathbf{x} + W^{g,l}S^1 + b^{g,l}\right), && l = 1, \dots, L, \\
R^l &= \sigma_1\left(U^{r,l}\mathbf{x} + W^{r,l}S^l + b^{r,l}\right), && l = 1, \dots, L, \\
H^l &= \sigma_1\left(U^{h,l}\mathbf{x} + W^{h,l}\left(S^l \odot R^l\right) + b^{h,l}\right), && l = 1, \dots, L, \\
S^{l+1} &= \left(1 - G^l\right) \odot H^l + Z^l \odot S^l, && l = 1, \dots, L, \\
f(\theta) &= \text{base} + \sigma_2\left(WS^{L+1} + b\right), && \sigma_2 > 0.
\end{aligned}
$$

Heston



COS
DGM
TDNM
Option price
Moneyness
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Heston



DGM
TDNM
Relative L2-error
0.000
0.001
0.002
0.003
0.004
0.005
0.0
0.2
0.4
0.6
0.8
1.0
t

Lifted Heston [3]

- Rough Heston: more accurate, but V not Markovian

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

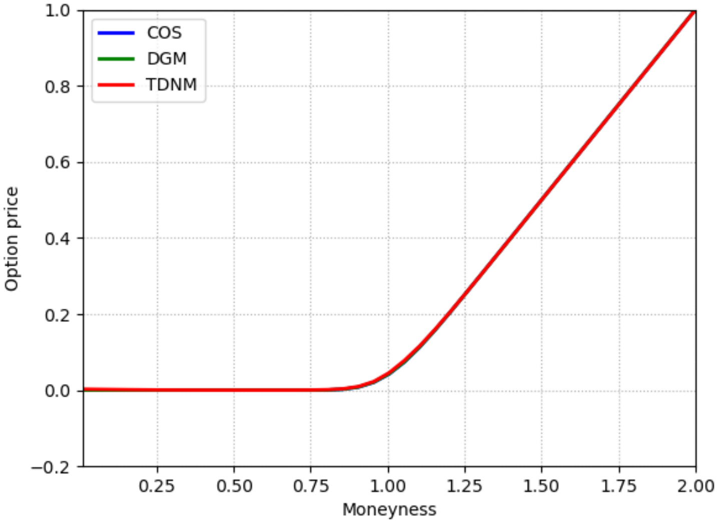
Lifted Heston [3]

- Rough Heston: more accurate, but V not Markovian
- Lifted Heston: Markovian, but multiple dimensions

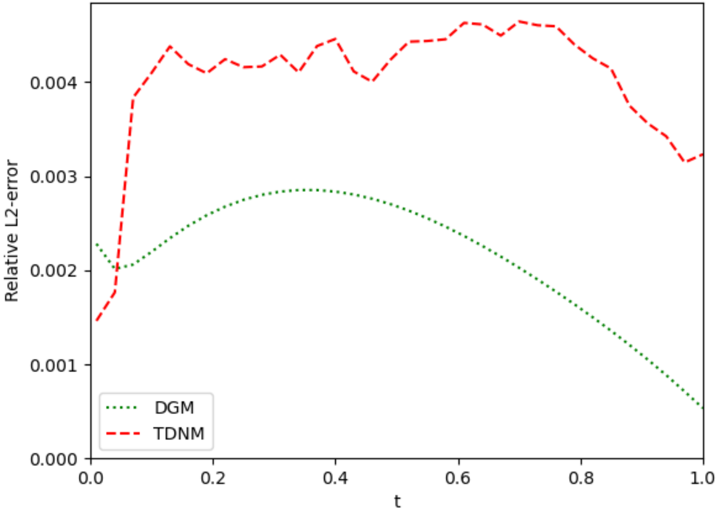
$$
\begin{aligned}
dS_t &= rS_t dt + \sqrt{V_t^n} S_t dW_t, & S_0 > 0, \\
V_t^n &= g^n(t) + \sum_{i=1}^{n} c_i^n V_t^{n,i}, \\
dV_t^{n,i} &= -\left(\gamma_i^n V_t^{n,i} + \lambda V_t^n\right) dt + \eta \sqrt{V_t^n} dB_t, & V_0^{n,i} = 0, \\
g^n(t) &= V_0 + \lambda\theta \sum_{i=1}^{n} c_i^n \int_0^t e^{-\gamma_i^n (t-s)} ds,
\end{aligned}
$$

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

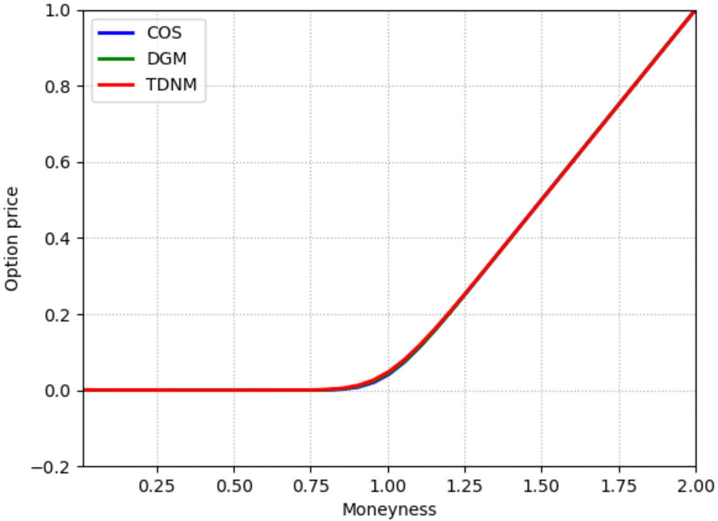
Lifted Heston, $n = 1$



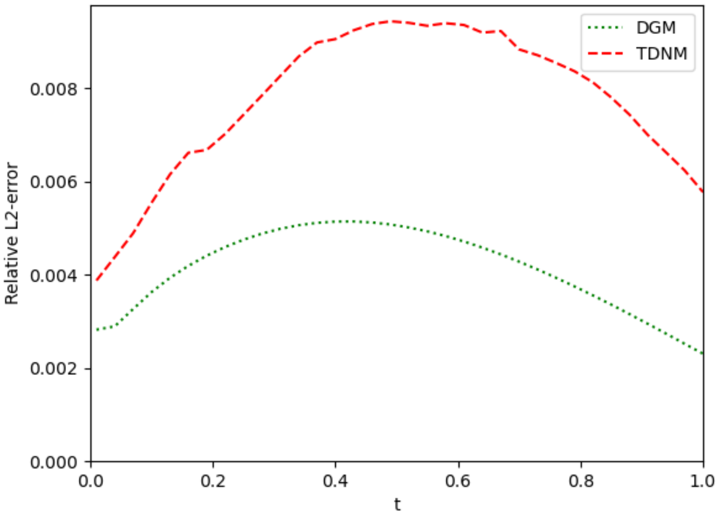
Lifted Heston, $n = 1$



Lifted Heston, $n = 20$



Lifted Heston, $n = 20$



Running times

Method	Heston	LH, n=1	LH, n=20
DGM	1.3	1.3	6.0
TDNM	0.77	0.66	1.0

Table: Training time (10^4 seconds)

Running times

Method	Heston	LH, n=1	LH, n=20
DGM	1.3	1.3	6.0
TDNM	0.77	0.66	1.0

Table: Training time (10^4 seconds)

Method	Heston	LH, n=1	LH, n=20
COS	10^{-2}	8.9	10.4
DGM	10^{-2}	10^{-2}	10^{-2}
TDNM	10^{-2}	10^{-2}	10^{-2}

Table: Computing time (seconds)

Conclusion

	Accurate	Fast
Heston	$\times$	✓
Lifted Heston	✓	$\times$

Conclusion

	Accurate	Fast
Heston	$\times$	$\checkmark$
Lifted Heston	$\checkmark$	$\times$
Lifted Heston with neural networks	$\checkmark$	$\checkmark$

Deep Gradient Flow Methods for Option Pricing in Diffusion Models

Finance Research Day

Jasper Rou

December 15, 2023

j.g.rou@tudelft.nl www.jasperrou.nl