Bergische Universität Wuppertal

Fakultät für Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 23/17

Tatiana Kossaczká, Matthias Ehrhardt and Michael Günther

Deep finite difference method for solving Asian option pricing problems

October 25, 2023

http://www.imacm.uni-wuppertal.de

Deep finite difference method for solving Asian option pricing problems

Tatiana Kossaczká[1], Matthias Ehrhardt[1], and Michael Günther[1]

Chair of Applied and Computational Mathematics, Bergische Universität Wuppertal, Gaußstrasse 20, 42119 Wuppertal, Germany,
kossaczka@uni-wuppertal.de

Abstract. Deep learning techniques have gained prominence in recent years for enhancing and optimizing existing numerical schemes. One crucial aspect in this context is the preservation of essential properties of these schemes, such as consistency and convergence.
We apply a recently developed innovative deep learning-based enhancement of the finite difference method to address the Asian option pricing problem, showcasing its efficacy through numerical results. Our research demonstrates that slight modifications to the scheme can yield even better results while preserving the essential theoretical order of convergence.

Keywords: finite difference scheme, deep learning, Asian options

1 Introduction

Recently, deep learning has been widely used to modify and improve existing numerical schemes. To do this, it is important to preserve the properties of the existing schemes, such as consistency and convergence. For example, in [1]-[4] the weighted essentially non-oscillatory (WENO) schemes, which belong to the class of modern finite difference methods (FDMs), have been successfully improved. The improved scheme exhibits better numerical results while maintaining the formal order of accuracy, which can be proven theoretically.

In this work, we use the recently developed deep learning based FDM [5]. As shown in [5], this deep learning extension of FDM is able to provide higher numerical accuracy and the method remains time efficient even when the deep learning part is added. We apply this method to the example of the Asian option pricing problem and show the numerical results using this method. Moreover, we show that we can obtain even better results by slightly modifying the scheme, while not destroying the theoretical order of convergence.

2 Deep FDM

We briefly introduce the Deep FDM (DFDM) as it was developed in [5]. We consider a one-dimensional PDE of a form

$$\begin{aligned} &\frac{\partial u}{\partial t} = \alpha(x)\frac{\partial^2 u}{\partial x^2} + \beta(x)\frac{\partial u}{\partial x} + \gamma(x)u, \qquad (x,t) \in \Omega_1 \times [0,T], \\ &u(x,0) = u_0(x), \end{aligned} \tag{1}$$

with the coefficients $\alpha, \beta, \gamma \colon \Omega_1 \subseteq \mathbb{R} \to \mathbb{R}$. We use the 1D spatial domain $\Omega_1 = [a, b]$ and introduce a uniform grid defined by the points $x_i = x_0 + i\Delta x$, $i = 0, 1, \ldots, I$. For the time domain $[0, T]$ we use the uniform discretization defined by the points $t_n = t_0 + n\Delta t$, $n = 0, 1, \ldots, N$. Let $u_i^n = u(x_i, t_n)$ be the value of the exact solution at the grid point (x_i, t_n) and $\hat{u}_i^n$ be the corresponding numerical approximation.

To modify the standard FDM, we use the Convolutional Neural Network (CNN). This ensures the spatial invariance of the method and the computational efficiency. Then, we define the neural network functions as $F(\cdot), G(\cdot) \colon \mathbb{R}^{2k+1} \to \mathbb{R}$, where $2k+1$ is the size of the receptive field of the CNN. Using the standard central finite difference scheme for the spatial discretizations and the explicit Euler scheme for the temporal discretization, we obtain

$$\begin{aligned} \hat{u}_i^{n+1} = \hat{u}_i^n + \Delta t \Big[\alpha(x_i) \Big(\frac{\hat{u}_{i+1}^n - 2\hat{u}_i^n + \hat{u}_{i-1}^n}{\Delta x^2} + \Delta x^2 F(\bar{u}_i^n) \Big) \\ + \beta(x_i) \Big(\frac{\hat{u}_{i+1}^n - \hat{u}_{i-1}^n}{2\Delta x} + \Delta x^2 G(\bar{u}_i^n) \Big) + \gamma(x_i)\, \hat{u}_i^n \Big]. \end{aligned} \tag{2}$$

The values $\bar{u}_i^n = \bar{u}^n(\bar{x}_i) = (\hat{u}^n(x_{i-k}), \ldots, \hat{u}^n(x_{i+k})) = (\hat{u}_{i-k}^n, \ldots, \hat{u}_{i+k}^n)$ represent the input to the neural network. $F(\bar{u}_i^n)$ and $G(\bar{u}_i^n)$ represent the output of a neural network trained so that the following approximations hold

$$F(\bar{u}_i^n) \approx \frac{1}{\Delta x^2}\, \epsilon_2, \qquad G(\bar{u}_i^n) \approx \frac{1}{\Delta x^2}\, \epsilon_1,$$

where $\epsilon_2 = O(\Delta x^2)$ and $\epsilon_1 = O(\Delta x^2)$ are the local truncation errors of both standard central finite difference schemes. For more details and detailed explanation we refer to [5].

3 Asian Option Pricing Problem

Let us consider Asian options of the European type depending on the arithmetic mean of the asset price. In this case, we obtain the equation for the price of the Asian option, which is given by the two-dimensional PDE. There are four different types of arithmetic Asian option with regards to the payoff function and we consider in this paper fixed-strike call option.

To avoid solving the two-dimensional PDE, Rogers and Shi [6] introduced a reduced PDE of the form

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sigma^2 x^2 \frac{\partial^2 u}{\partial x^2} - \Big(\frac{1}{T} + rx \Big) \frac{\partial u}{\partial x} = 0, \quad 0 \le t \le T, \quad x \in \mathbb{R}, \tag{3}$$

which can be solved equivalently for European style of Asian options. Here, r denotes an interest rate, σ the volatility, x is defined by

$$x = \frac{1}{S_t} \Big(K - \int_0^t S_\upsilon\, \mu(d\upsilon) \Big), \tag{4}$$

where μ is the probability measure with the density $\rho(t) = 1/T$ in our case. Further, S is the price of an underlying asset at time t and K the strike price. The price of an option V is then given by $V = S_0 u(K/S_0, 0)$ for some initial stock price S_0.

After transforming (3) to the forward-in-time PDE using $\tau = T - t$, we obtain

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}\sigma^2 x^2 \frac{\partial^2 u}{\partial x^2} - \left(\frac{1}{T} + rx\right)\frac{\partial u}{\partial x}, \quad 0 \le \tau \le T, \quad x \in \mathbb{R}, \tag{5}$$

which has a form of the PDE (1). We use the initial condition [6]

$$u(x, 0) = \max(0, -x), \tag{6}$$

and the boundary conditions [6]

$$u_0 = \frac{1 - e^{-r\tau}}{rT} - x_0 e^{-r\tau}, \qquad u_N = 0. \tag{7}$$

We select the computational domain $[x_l, x_r] = [-0.4, 4]$ and final time $T = 1$.

4 Modification of the Scheme and Training Procedure

The coefficient in front of a convection term $(\frac{1}{T} + rx)$ is always positive in our case and can become dominant. This means that to overcome the oscillations, which could appear in the solution, the left-biased stencil should be used to approximate the first derivative. This reads

$$\frac{\partial u}{\partial x}\Big|_{x_i} = \frac{u(x_i, t) - u(x_{i-1}, t)}{\Delta x} + O(\Delta x). \tag{8}$$

As we can see, the approximation is only of the first order, so for the enhanced DFDM we can use the formula

$$\hat{u}_i'^n = \frac{\hat{u}_i^n - \hat{u}_{i-1}^n}{\Delta x} + \Delta x G(\bar{u}_i^n) \tag{9}$$

to get the deep learning improved approximation of the first derivative. Now we insert the equation (9) into (2), replacing the central approximation of the first derivative, and use it as our final scheme.

Remark 1. Also one-sided second order finite difference approximation for the first derivative could be used. However, for this we would need to define one more boundary point on the left side of the computational domain. Moreover, wider stencil can again lead to numerical oscillations. In Section 5 we will compare the numerical results using (9) as well as using second-order finite difference approximations for the first derivative.

We also introduce a multiplication factor for the deep learning terms

$$\begin{aligned} F(\bar{u}_i^n) &= 10^3 \min\big(|\hat{u}_{i+1}^n - 2\hat{u}_i^n + \hat{u}_{i-1}^n|, 10^2\big)\, \mathcal{F}(\bar{u}_i^n), \\ G(\bar{u}_i^n) &= 10^2 \min\big(|\hat{u}_{i+1}^n - 2\hat{u}_i^n + \hat{u}_{i-1}^n|, 10^2\big)\, \mathcal{G}(\bar{u}_i^n). \end{aligned} \tag{10}$$

By adding these factors, we ensure that the bounded learned coefficients have significant effects only in the non-smooth (kinked) part of the solution. By ensuring that these factors are bounded, we do not violate the convergence properties.

For computational efficiency, we use a small CNN structure described in Figure 1. Two output channels in the last hidden layer represent the correction $\mathcal{F}(\bar{u}_i^n)$ of a diffusion term and the correction $\mathcal{G}(\bar{u}_i^n)$ of a convection term of (5).

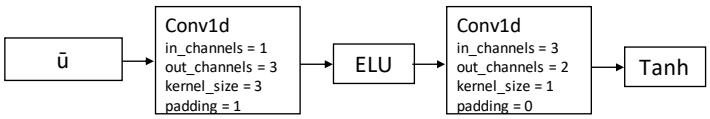


Fig. 1: The structure of the CNN used for the training.

We aim to obtain a numerical scheme, which can reliably solve the Asian option pricing problem for all possible combinations of σ, r and T. For this purpose, we first create a data set consisting of 100 reference solutions. We generate the parameters σ and r randomly

$$\sigma \in \mathcal{U}[0.1, 0.4], \qquad r \in \mathcal{U}[0.1, 0.3]. \tag{11}$$

For training, we fix $T = 1$, $K = 100$ and use the computational domain $[x_l, x_r] = [-0.4, 4]$. The reference solutions are computed using standard central finite difference schemes on a grid divided into 400 space points and the temporal step size is chosen such that $N = \max_{i=0,\dots,I} \Big((\tau\sigma^2 x_i^2)/(0.8\Delta x^2)\Big)$.

For the training, we use the training procedure described in [5]. We divide the spatial computational domain into 50 space steps. Then, we randomly select a problem from a data set. Afterwards, we compute successively the solution up to the fixed final time T. After each time step n we compute the loss with respect to the weights of CNN, update the weights and continue to the next time step $n+1$. This means, in each subsequent time step, a new updated solution according to (2) is input to the CNN. For the optimization we use the Adam optimizer with the learning rate 0.0001. For the training procedure, we use the mean squared error loss function

$$\mathrm{LOSS}_{\mathrm{MSE}}(u) = \frac{1}{I}\sum_{i=0}^{I}(\hat{u}_i^n - u_i^{n,\mathrm{ref}})^2, \tag{12}$$

where $\hat{u}_i^n$ is a numerical approximation of $u(x_i, t_n)$ obtained by DFDM, and $u_i^{n,\mathrm{ref}}$ denotes the reference solution.

After the training, we choose the model from a training step, in which the best performance on problems from the validation set is obtained. These are the problems with randomly generated initial parameters, which were not in the training data. This is our final DFDM and we present the numerical results using this method in the following section.

5 Numerical Results

Let us present the numerical results on a test set containing the problems with the randomly generated initial data.

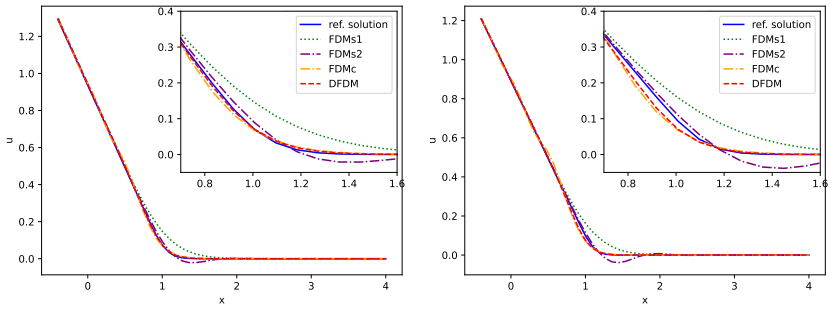
We compare the L_2 errors in Table 1. We computed the solution with the given r and σ parameters as given in the table. For the computation, we used the central finite difference formula for the diffusion term, and for the convection term, we distinguish among the following possibilities: second-order central finite difference scheme (FDMc), second-order one-sided finite difference scheme (FDMs2), first-order one-sided finite difference scheme (FDMs1), and deep learning improved first-order one-sided finite difference scheme (DFDM). As can be seen, DFDM has the smallest L_2 errors in all cases. Compared to the FDMs1, we obtain the largest improvement. The ratio denotes the error of the listed standard finite difference methods divided by the error of DFDM.

We illustrate the solution for two selected cases in Figure 2. As can be seen, FDMc and FDMs2 lead to spurious oscillations in the solution. FDMs1 does not cause any oscillations, but has a large error near the kink. DFDM produces the best solution among the methods.

We also computed the solution for the different final computation times T. The results are shown in Table 2. Let us note, that we only trained the method with the fixed $T = 1$, but we observe improving results also for different computation times. Based on the presented results it can be stated, that the longer computational time leads to even better numerical results using DFDM.

parameters		L_2				improvement ratios		
σ	r	FDMs1	FDMs2	FDMc	DFDM	ratio (FDMs1)	ratio (FDMs2)	ratio (FDMc)
0.22	0.12	0.050304	0.016930	0.017986	**0.008781**	5.73	1.93	2.05
0.18	0.13	0.050193	0.021800	0.021481	**0.011393**	4.41	1.91	1.89
0.32	0.24	0.047173	0.010243	0.010361	**0.003628**	13.00	2.82	2.86
0.2	0.21	0.049032	0.018337	0.017864	**0.008794**	5.58	2.09	2.03
0.32	0.15	0.046425	0.010367	0.010167	**0.003600**	12.89	2.88	2.82
0.23	0.16	0.049239	0.015483	0.015921	**0.007031**	7.00	2.20	2.26
0.35	0.24	0.046521	0.009166	0.009298	**0.003086**	15.08	2.97	3.01
0.16	0.23	0.047185	0.025895	0.022651	**0.015085**	3.13	1.72	1.50
0.27	0.18	0.047725	0.012656	0.012479	**0.004896**	9.75	2.58	2.55
0.15	0.28	0.047074	0.029096	0.025772	**0.017967**	2.62	1.62	1.43

Table 1: Comparison of L_2 error for the solution of (5) with various parameters σ and r using different finite difference methods, $I = 50$, $T = 1$.



(a) $\sigma = 0.22$, $r = 0.12$. (b) $\sigma = 0.16$, $r = 0.23$.

Fig. 2: Comparison of the solution of (5) using different finite difference methods, $I = 50$, $T = 1$.

parameters			L_2				improvement ratios		
σ	r	T	FDMs1	FDMs2	FDMc	DFDM	ratio (FDMs1)	ratio (FDMs2)	ratio (FDMc)
0.22	0.12	0.8	0.051032	0.019911	0.020737	**0.010952**	4.66	1.82	1.89
0.18	0.13	0.9	0.047113	0.023283	0.019891	**0.012144**	3.88	1.92	1.64
0.32	0.24	1.1	0.047800	0.009610	0.010600	**0.003927**	12.17	2.45	2.70
0.2	0.21	2	0.049840	0.011665	0.012847	**0.004838**	10.30	2.41	2.66
0.32	0.15	0.5	0.046685	0.016061	0.013966	**0.006675**	6.99	2.41	2.09
0.23	0.16	3	0.048125	0.007549	0.008789	**0.002720**	17.69	2.78	3.23
0.35	0.24	1.5	0.045988	0.006936	0.007465	**0.002200**	20.91	3.15	3.39
0.16	0.23	1.8	0.048583	0.017193	0.016256	**0.009683**	5.02	1.78	1.68
0.27	0.18	0.6	0.050456	0.018115	0.018972	**0.009517**	5.30	1.90	1.99
0.15	0.28	2.5	0.052187	0.014689	0.016639	**0.008149**	6.40	1.80	2.04

Table 2: Comparison of L_2 error for the solution of the equation (5) with various parameters σ, r and T using different finite difference methods, $I = 50$.

References

1. T. Kossaczká, M. Ehrhardt, and M. Günther, Enhanced fifth order WENO shock-capturing schemes with deep learning. Res. Appl. Math. 12 (2021), 100217.
2. T. Kossaczká, M. Ehrhardt, and M. Günther, A deep smoothness WENO method with applications in option pricing. In: M. Ehrhardt and M. Günther, M. (eds.), Progress in Industrial Mathematics at ECMI 2021. Springer (2021), 417-423.
3. T. Kossaczká, M. Ehrhardt, and M. Günther, A neural network enhanced WENO method for nonlinear degenerate parabolic equations. Phys. Fluids 34(2) (2022), 026604.
4. T. Kossaczká, A.D. Jagtap, and M. Ehrhardt, Deep smoothness WENO scheme for two-dimensional hyperbolic conservation laws: A deep learning approach for learning smoothness indicators. arXiv preprint 2309.10117, September 2023.
5. T. Kossaczká, M. Ehrhardt, and M. Günther, Deep FDM: Enhanced finite difference methods by deep learning. Franklin Open 4 (2023), 100039.
6. L.C.G. Rogers and Z. Shi, The value of an Asian option. J. Appl. Prob. 32(4) (1995), 1077-1088.