

This is an Open Access document downloaded from ORCA, Cardiff University's institutional repository: <https://orca.cardiff.ac.uk/id/eprint/162690/>

This is the author's version of a work that was submitted to / accepted for publication.

Citation for final published version:

Nwankwo, Chinonso, Umeorah, Nneka , Ware, Tony and Dai, Weizhong 2024. Deep learning and American options via free boundary framework. Computational Economics 64 , pp. 979-1022. 10.1007/s10614-023-10459-3

Publishers page: <http://dx.doi.org/10.1007/s10614-023-10459-3>

Please note:

Changes made as a result of publishing processes such as copy-editing, formatting and page numbers may not be reflected in this version. For the definitive version of this publication, please refer to the published source. You are advised to consult the publisher's version if you wish to cite this paper.

This version is being made available in accordance with publisher policies. See <http://orca.cf.ac.uk/policies.html> for usage policies. Copyright and moral rights for publications made available in ORCA are retained by the copyright holders.



Deep learning and American options via free boundary framework

Chinonso Nwankwo^{*1}, Nneka Umeorah², Tony Ware¹ and Weizhong Dai³

¹Department of Mathematics and Statistics, University of Calgary, Calgary T2N 1N4, Canada

²School of Mathematics, Cardiff University, Cardiff CF24 4AG, United Kingdom

³Department of Mathematics and Statistics, Louisiana Tech University, Ruston, LA 71272, USA

Abstract

We propose a deep learning method for solving the American options model with a free boundary feature. To extract the free boundary known as the early exercise boundary from our proposed method, we introduce the Landau transformation. For efficient implementation of our proposed method, we further construct an implicit dual solution framework consisting of a novel auxiliary function and free boundary equations. The auxiliary function is formulated to include the feed-forward deep neural network (DNN) output and further mimic the far boundary behaviour, smooth pasting condition, and remaining boundary conditions due to the second-order space derivative and first-order time derivative. Because the early exercise boundary and its derivative are not a priori known, the boundary values mimicked by the auxiliary function are in approximate form. Concurrently, we then establish equations that approximate the early exercise boundary and its derivative directly from the DNN output based on some linear relationships at the left boundary. Furthermore, the option Greeks are obtained from the derivatives of this auxiliary function. We test our implementation with several examples and compare them with the existing numerical methods. All indicators show that our proposed deep learning method presents an efficient and alternative way of pricing options with early exercise features.

Keywords: PDE, American options, Early exercise boundary, Front fixing method, DNN

1 Introduction

Different forms of neural networks (NN) have been developed to aid the solution of both low and high-dimensional partial differential equations (PDEs). These include Physics-Informed NN (Raissi et al., 2019, 2017; Wang et al., 2022); the feed-forward NN and its application in DeepXDE (Lu et al., 2021); convolutional NN (He and Xu, 2019), (Khoo et al., 2021); recurrent NN (Sabate-Vidales et al., 2020); Multi-layer perceptron (MLP) NN (He et al., 2000); radial basis function NN (Golbabai and Seifollahi, 2006; Jianyu et al., 2003); modified artificial NN (Hussian and Suhhiem, 2015); and the deep Galerkin Method for high-dimensional PDEs Sirignano and Spiliopoulos (2018). Implementing the artificial NN for option pricing has been traced back to the works of Malliaris and Salchenberger (1993) and Hutchinson et al. (1994), whose research serves as a benchmark for the subsequent studies on the machine learning (ML) applications in pricing and hedging of financial derivatives. The authors showed that the European-style option valuation based on the Black-Scholes model could be successfully recovered with NN. DNN has been extensively studied in the area of pricing, hedging, and calibration of financial derivatives and stochastic models (Chen and Wan, 2021); implied volatility smile with its calibration, algorithmic trading, credit risk analysis (Pacelli et al., 2011; Khashman, 2010), assessment of bond ratings (Dutta and Shekhar, 1988), forecasting of volatility, etc.

Free boundary PDEs often appear in diverse science, engineering, and mathematics areas and have a wider application in heat transfer models, population models, predator-prey models, finance, fluid

^{*}Corresponding author: chinonso.nwankwo@ucalgary.ca, <https://orcid.org/0000-0001-5526-1337>

flow problems, etc. They can be seen in melting and solidification problem (Alexiades and Cannon, 1980; Ceseri and Stockie, 2014; Hu and Argyropoulos, 1996; Kumar et al., 2006), semi-conductor problem (Morandi Cecchi and Russo, 2005; Schmeiser, 2020), forest fire simulation (Filippi et al., 2010), combustion processes (Andreucci and Gianni, 1994; Caffarelli and Vázquez, 1995; Frankel and Roytburd, 1994; Kim, 2003), fluid-structure interactions (Bänsch et al., 2013), logistic population models (Piqueras et al., 2017), and option pricing and hedging (MacKean Jr, 1965). Most free boundary PDE models do not have a closed-form solution. Hence, several numerical and semi-analytical approaches have been proposed in the existing literature for solving these models. Recently, Wang and Perdikaris (2021) proposed a deep learning approach for solving free boundary PDE related to the heat transfer problem. The authors presented a multi-network model based on physics-informed neural networks (PINN) and implemented it for solving one-dimensional one-phase Stefan problem, one dimensional two-phase Stefan problem, and the two-dimensional one-phase Stefan problem. Also, Zhao et al. (2021) solved a free boundary problem based on the Hele-Shaw model (Hele-Shaw, 1898). The authors further proved and established the existence of their numerical solution.

American options model can be seen as a free boundary PDE model. In contrast to the vanilla European options, the American options possess the free boundary, and the holder of the options is allowed to exercise its right before expiration. This advantage comes with a cost because the early exercise boundary is computed simultaneously with the value function and the option Greeks. Some standard numerical and analytical approximation methods for solving the American options model include the binomial tree method (Cox et al., 1979), least square method (Longstaff and Schwartz, 2001), (Létourneau and Stentoft, 2019), integral method (Carr, 1998), first passage approach (Bunch and Johnson, 2000; Gutiérrez, 2013), and mesh/meshless methods (Brennan and Schwartz, 1977). All these approaches are not without their advantages and limitations. In the PDE framework, American options can be formulated as a free boundary problem, in the linear complementary framework, or with a penalty term. The linear complementary framework and penalty method solve this model by introducing a constraint that removes the early exercise feature. It is important to mention that the early exercise boundary is an indispensable feature in American options valuation, which is very useful in the trading environment. Furthermore, the accuracy of the value function strongly depends on this early exercise feature. Computing the free boundary with high precision becomes paramount and not just a necessity.

The complication in solving American options becomes more substantial in a high dimensional context when precise early exercise boundary value is desired. For instance, American options with stochastic volatility involve a free boundary plane, a function of time and volatility. For American options with stochastic volatility and interest rate, a three-dimensional framework for the free boundary will be obtained: a function of time, volatility, and interest rate. Hence, it might be cumbersome and computationally expensive to obtain the free boundary structure for the basket FX options with stochastic volatility when the finite difference, element, or volume method is considered. Furthermore, options that involve two or multi-free boundaries, like the strangle and straddle options, can also pose challenges to approximate in a high-dimensional context. When the conventional finite difference, volume, or element are implemented, considering multi-dimensional basket stock American options under a linear complementary framework can be computationally intractable due to the curse of dimensionality. It is worth mentioning that NN and its extension have also been implemented for solving American-style options in both low and high-dimensional contexts (Anderson and Ulrych, 2022; Becker et al., 2020; Chen and Wan, 2021; Hirsä et al., 2019; Lin and Almeida, 2021; Reppen et al., 2022; Sirignano and Spiliopoulos, 2018; Villani, 2022).

Our research comes from the non-parametric aspect of option pricing and hedging as we seek to solve the American options model with a deep learning approach. In this research, we propose a DNN method for solving the American options model as a free boundary problem which entails extracting the early exercise features while approximating the options simultaneously. To this end, we first use the logarithmic Landau transformation to fix the free boundary. Furthermore, we introduce a novel

auxiliary function that both mimics the boundary values of the exact value function and in-cooperates the feed-forward DNN part. We formulate the auxiliary function to simultaneously, satisfy the far field boundary, the value matching condition, the high contact condition known as the smooth pasting, and the second-order in space and first-order in time boundary conditions. These formulations are under the condition that a linear relationship between the early exercise boundary and the feed-forward DNN output at the left boundary holds. This condition is actually the value-matching condition in approximate form. Training the NN is an optimization procedure that involves computation and minimization of the gradient of the loss function obtained from the auxiliary function with respect to the network parameters. The performance of our proposed DNN approach is investigated and compared extensively with other numerical methods in the result section. Thus, the major highlights of this research are summarized below:

- We propose a novel deep learning methodology via a dual solution framework consisting of an auxiliary function and early exercise boundary equations for solving a nonlinear American options pricing model.
- Due to the non-linearity and non-smoothness associated with the transformed American options model and a number of our trainable data, we opt for the feed-forward DNN algorithm in approximating our value function, option Greeks, and early exercise boundary with its first derivative.
- We perform some hyper-parameter tunings to determine the optimal architecture for our DNN model and justify our selection using some statistical analysis.
- Our implementation was tested with several examples that account for varying time to maturity T and strike price K . We further compared our results with existing methods (Egorova et al., 2014; Fazio et al., 2021; Nielsen et al., 2002) including the results obtained from a highly accurate sixth-order compact finite difference scheme with left boundary improvement (Nwankwo and Dai, 2022b).

The remaining sections of this work are organized as follows. In Section 2, we introduced the transformed American options using the logarithmic Landau transformation and further normalized the transformed model. Section 3 explains the concept of the DNN approximation. Section 4 presents the experimental results; a discussion and analysis of the results follow, and then a comparison with the existing method is illustrated. Finally, Section 5 concludes the study and provides guidelines for future research.

2 Normalized American Options Model with Landau Transformation

Let us consider the American put options written on an underlying asset with an asset price $S(u)$. Under risk neutral probability $\mathbb{Q}$, $S(u)$ is driven by geometric Brownian motion as follows:

$$dS(u) = (r - \delta)S(u)dt + \sigma S(u)dW(u), \quad (2.1)$$

where $\{W(u), u > 0\}$ is the standard Brownian motion. Furthermore, σ , δ , and r represents the volatility, dividend, interest rate, respectively. Let us denote $V(u, S)$ as the value with K and T representing the strike price and time to maturity, respectively. Furthermore, if we solve backward in time with $t = T - u$, then the free boundary PDE is given below.

$$V_t(t, S) - \frac{\sigma^2 S^2}{2} V_{SS}(t, S) - (r - \delta)SV_S(t, S) + rV(t, S) = 0, \quad S > s_f(t); \quad (2.2)$$

$$V(t, S) = K - S, \quad S < s_f(t). \quad (2.3)$$

The boundary and the initial conditions are as follows:

$$V(0, S) = \max(K - S, 0), \quad V(t, s_f(t)) = K - s_f(t) \quad V(t, \infty) = 0. \quad (2.4)$$

To compute a more precise value of the early exercise feature with the options, we consider the logarithmic Landau (Wu and Kwok, 1997) and time transformation. To this end, let

$$x = \ln S - \ln s_f(t), \quad V(t, S) = U(t, x). \quad (2.5)$$

Hence, the transformed fixed free boundary problem with the initial and boundary condition is given as follows:

$$U_t(t, x) - \frac{\sigma^2}{2} U_{xx}(t, x) - \left(r - \delta + \frac{s'_f(t)}{s_f(t)} - \frac{\sigma^2}{2} \right) U_x(t, x) + rU(t, x) = 0, \quad x > 0; \quad (2.6)$$

$$U(t, x) = K - e^x s_f(t), \quad U_x(t, x) = -e^x s_f(t), \quad x < 0; \quad (2.7)$$

$$U_t(t, x) = -e^x s'_f(t), \quad U_{xx}(t, x) = -e^x s_f(t), \quad x < 0. \quad (2.8)$$

The first derivative of the early exercise boundary with respect to t is denoted as $s'_f(t)$. Here, $U_x(t, x)$ and $U_t(t, x)$, and $U_{xx}(t, x)$ are the derivatives of the value function which represent the Greeks. The boundary and initial conditions are also given below.

$$U(0, x) = 0, \quad U(t, 0) = K - s_f(t), \quad U(t, \infty) = 0, \quad t > 0; \quad (2.9)$$

$$U_x(0, x) = 0, \quad U_x(t, 0) = -s_f(t), \quad U_x(t, \infty) = 0, \quad t > 0; \quad (2.10)$$

$$U_t(0, x) = 0, \quad U_t(t, 0) = -s'_f(t), \quad U_t(t, \infty) = 0, \quad t > 0. \quad (2.11)$$

Furthermore, some information regarding the second-order derivative of the value function can be obtained from the PDE when $x \rightarrow 0^+$, $x \rightarrow 0^-$, and $x \rightarrow \infty$ as follows:

$$\lim_{x \rightarrow 0^+} U_{xx}(t, x) = \frac{2rK}{\sigma^2} - \left(\frac{2\delta}{\sigma^2} + 1 \right) s_f(t), \quad \lim_{x \rightarrow 0^-} U_{xx}(t, x) = -s_f(t), \quad t > 0; \quad (2.12)$$

$$U_{xx}(t, \infty) = 0. \quad (2.13)$$

After fixing the free boundary, we are now focusing our interest on the continuation region and consider only the positive semi-infinite space domain. Hence the fixed space and time domain under consideration is given as $(t, x) \in ([0, T] \times [0, \infty))$. Because the option values vanish rapidly as we move further away from the fixed left boundary, if we consider the far field boundary condition with the right boundary given as $x_{\max}$, then the fixed domain is reduced to $[0, T] \times [0, x_{\max}]$.

For effective implementation of the DNN method for solving the transformed American options model, we reduced our fixed space and time domain from $[0, T] \times [0, x_{\max}]$ to $[0, 1] \times [0, 1]$. To this end, we introduce another transformation as follows:

$$\tau = \frac{t}{T}, \quad y = \frac{x}{x_{\max}}, \quad P(\tau, y) = \frac{1}{x_{\max}} U(x, \tau). \quad (2.14)$$

The normalized model with the fixed free boundary is then given below:

$$\begin{aligned} \frac{1}{T} P_\tau(\tau, y) - \frac{1}{x_{\max}^2} \frac{\sigma^2}{2} P_{yy}(\tau, y) - \frac{1}{x_{\max}} \left(r - \delta - \frac{\sigma^2}{2} \right) P_y(\tau, y) - \frac{1}{x_{\max} T} \frac{s'_f(\tau)}{s_f(\tau)} P_y(\tau, y) \\ + rP(\tau, y) = 0, \quad y > 0. \end{aligned} \quad (2.15)$$

For the initial and boundary conditions after the normalization of the transformed model,

$$P(\tau, y) = \frac{K}{x_{\max}} - \frac{e^{yx_{\max}}}{x_{\max}} s_f(\tau), \quad P_y(\tau, y) = -s_f(\tau), \quad y < 0; \quad (2.16)$$

$$P_\tau(\tau, y) = -\frac{e^{yx_{\max}}}{T x_{\max}} s'_f(\tau), \quad P_{yy}(\tau, y) = -x_{\max} s_f(\tau), \quad y < 0; \quad (2.17)$$

$$P(0, y) = 0, \quad P(\tau, 0) = \frac{K}{x_{\max}} - \frac{s_f(\tau)}{x_{\max}}, \quad P(\tau, 1) = 0; \quad (2.18)$$

$$P_y(0, y) = 0, \quad P_y(\tau, 0) = -s_f(\tau), \quad P_y(\tau, 1) = 0; \quad (2.19)$$

$$P_\tau(\tau, 0) = -\frac{s'_f(\tau)}{x_{\max}}, \quad P_\tau(\tau, 1) = 0, \quad \tau > 0. \quad (2.20)$$

Substituting $P(\tau, y)$ and its derivative into (2.15) when $y \rightarrow 0^+$ and $y \rightarrow 1^-$, we then obtain the boundary value of the second-derivative of the value function as follows:

$$P_{yy}(\tau, 0) = \left[\frac{2rK}{\sigma^2} - \left(\frac{2\delta}{\sigma^2} + 1 \right) s_f(\tau) \right] x_{\max}, \quad P_{yy}(\tau, 1) = 0, \quad \tau > 0. \quad (2.21)$$

For the left corner-point $(0, 0)$ which is one of the most sensitive points in our model, the following condition holds.

$$\lim_{y \rightarrow 0^+} U(0, y) = 0, \quad \lim_{y \rightarrow 0^+} U_y(0, y) = 0, \quad s_f(0) = \min\{a_r\delta, 1\}, \quad a_r\delta = \frac{r}{\delta}. \quad (2.22)$$

Furthermore, for the non-dividend paying American put options pricing model, we have it that the early exercise boundary is monotonically decreasing with (Musiela and Rutkowski, 2006)

$$s_f(0) = K, \quad 0 \leq s_f(\tau) \leq K, \quad s_f(\infty) = \frac{\xi}{\xi + 1}K, \quad \xi = \frac{2r}{\sigma^2}. \quad (2.23)$$

3 Front-Fixing DNN and the American Options Model

Deep learning is a sub-class of the ML algorithm (Deng et al., 2014), which uses multiple hidden layers to extract higher-level features from the specified raw input data. The learning can be supervised, unsupervised or semi-supervised and has applications in speech recognition, regression, image classification, game intelligence, etc. (Liu and Markowich, 2020). Deep learning procedures has attained remarkable success in different applied fields, though its application in PDEs has emerged recently. In this section, we present our proposed DNN method for solving the fixed free boundary American options, which simultaneously accounts for the early exercise feature, value function, and option Greeks.

3.1 DNN via an auxiliary function

This section introduces a novel function that solves the optimal stopping problem and, more precisely, leverages the position to approximate the early exercise boundary simultaneously with the value function and Greeks using the DNN techniques. We called this function an auxiliary function. Closely related implementation was first introduced in the work of Lagaris et al. (1998) for solving ODEs and PDEs, which they called the trial solution. Subsequently, some authors implemented Lagaris et al. (1998) approach for solving the options pricing model mainly in the European options framework (Eskiizmirliler et al., 2021; Hou et al., 2022) and in barrier options (Umeorah and Mba, 2022). Because we consider American options as a free boundary problem and implemented a front-fixing approach, several boundary values of the value function and its derivatives pose a more significant challenge using Lagaris et al. (1998) approach directly. This is by no means trivial, and we consider a slightly different approach that is similar in representation (but different in formulation) to the one presented in the works of Chen et al. (2020) and Liu et al. (2019). To this end, we present the novel auxiliary function as follows:

$$\mathcal{P}(\tau, y; \nu) = e^{-\frac{y^2 \gamma}{2\tau}} \mathcal{M}(\tau, y; \nu), \quad (\tau, y) \in ((0, 1], [0, 1]). \quad (3.1)$$

The exponential part of $\mathcal{P}(\tau, y; \nu)$ closely resembles that of the probability density function of the Brownian motion. Here,

$$\mathcal{M}(\tau, y; \nu) = \sum_{i=0}^m \frac{f_i(\tau; \nu) y^i}{i!} + z(\tau, y) \mathcal{N}(\tau, y; \nu), \quad \gamma = \frac{r\phi}{\sigma^2}, \quad \phi \in \mathbb{R}_+; \quad (3.2)$$

and $m = 2$ represents the highest order of the differential operator present in our model with

$$z(\tau, y) = yx_{\max}, \quad f_0(\tau; \nu) = \frac{K - \bar{s}_f(\tau; \nu)}{x_{\max}}, \quad f_1(\tau; \nu) = -K; \quad (3.3)$$

$$f_2(\tau; \nu) = x_{\max} \left(\left[\frac{2rK}{\sigma^2} - \left(\frac{2\delta}{\sigma^2} + 1 \right) \bar{s}_f(\tau; \nu) \right] + \frac{\gamma}{\tau x_{\max}^2} (K - \bar{s}_f(\tau; \nu)) - 2\mathcal{N}_y(\tau, 0; \nu) \right), \quad (3.4)$$

$$\bar{s}_f(\tau; \nu) = K - x_{\max} \mathcal{N}(\tau, 0; \nu). \quad (3.5)$$

The auxiliary function in (3.1) can be seen as a combination of sub-functions which satisfies the boundary conditions of the value function in the approximate form (due to the early exercise boundary) and further include the feed-forward DNN output. ν represents the adjustable weights, and biases from the DNN output $\mathcal{N}(\tau, y; \nu)$. $\mathcal{N}_y(\tau, 0; \nu)$ is the first derivative (in space) of the feed-forward DNN output when $y = 0$. Here, γ controls the speed at which the auxiliary function vanishes when $y \rightarrow 1$, hence ensuring that we satisfy the far boundary behaviour. It is dependent on the constant parameter ϕ . $\bar{s}_f(\tau; \nu)$ given in (3.5) is the DNN approximation of the early exercise boundary, which is not known a priori. As we will later describe in this work, our deep learning method based on this auxiliary function will simultaneously, approximate $\bar{s}_f(\tau; \nu)$, the value function and Greeks.

Remark. We select ϕ in such a way that γ is not too small to force the exponential part to decay too slowly and also not too large, which could result in extremely rapid decay. These two scenarios can hugely impact the far boundary behaviour, minimization of the loss function, and the DNN approximation. We envisage that further analysis can be performed on ϕ to always determine the optimal value. Heuristically, for the present auxiliary function in (3.1)-(3.5), we observed that $\phi = 100$ provides consistent and stable performance across the experiment we carried out in the result section.

Furthermore, we formulate $\mathcal{P}(\tau, y; \nu)$ so that it simultaneously satisfies the payoff conditions, high contact condition known as the smooth pasting condition, and the remaining boundary values involving the first order time derivative and second order space derivative of the value function. We verify this in (3.6), (3.11), (3.12), and (3.16) below. We want to note that even though we formulate the auxiliary function to satisfy the exact boundary conditions, our left boundary values are not exact because of the early exercise boundary. This is because we simultaneously approximate the latter from the feed-forward DNN output with the value function. Our approach for approximating the early exercise boundary equation and its velocity is similar but not exactly the same when compared with the PINN method (Raissi et al., 2017, 2019) for computing the boundary and initial values from the NN output. The slight discrepancy is because we have two unknowns in (3.5) consisting of the early boundary and the DNN output when $y = 0$.

The feed-forward DNN output and its derivatives are embedded both in the auxiliary function and free boundary equations. At this juncture, the inherent discontinuity in the option pricing model at the payoff is worth mentioning. To what extent the DNN output minimizes this challenge remains to be seen. From (2.15) and (3.1), one can easily observe that the transformed model and auxiliary function are dominated by the early exercise boundary and its first derivative. Furthermore, both the early exercise boundary and its derivative is approximated directly from the DNN output and its first derivative in time. Hence, the extent to which the DNN output learns and improves the inherent discontinuity in the model will influence the accuracy of the early exercise boundary and will further affect the precise computation of the option price and the Greeks. We give further details in the result section. It is easy to see from (3.1)-(3.4) that the auxiliary function mimics the boundary value of the value function in the approximate form (due to the early exercise boundary) as follows:

$$\mathcal{P}(\tau, 0; \nu) = \frac{K - \bar{s}_f(\tau; \nu)}{x_{\max}}, \quad \mathcal{P}(\tau, 1; \nu) \cong 0, \quad \mathcal{P}(0, y; \nu) = 0, \quad y > 0. \quad (3.6)$$

Moreover, $\mathcal{P}(\tau, 0; \nu)$ is the auxiliary function evaluated at $y = 0$. $\mathcal{P}(\tau, 0; \nu)$ will be the same with $\mathcal{P}(\tau, 0)$ if $\bar{s}_f(\tau; \nu) = s_f(\tau)$. Furthermore, it is easy to see that $\mathcal{P}(\tau, 0; \nu) = \mathcal{N}(\tau, 0; \nu)$. Hence, our

auxiliary function satisfies, in approximate form, the value-matching condition. Taking derivatives of the auxiliary function, we then obtain

$$\begin{aligned} \mathcal{P}_y(\tau, y; \nu) &= e^{\frac{-y^2\gamma}{2\tau}} [-K + yf_2(\tau; \nu) + x_{\max}\mathcal{N}(\tau, y; \nu) + yx_{\max}\mathcal{N}_y(\tau, y; \nu)] \\ &\quad - \frac{y\gamma}{\tau} e^{\frac{-y^2\gamma}{2\tau}} \left[\frac{K - \bar{s}_f(\tau; \nu)}{x_{\max}} - yK + \frac{y^2}{2} f_2(\tau; \nu) + yx_{\max}\mathcal{N}(\tau, y; \nu) \right]. \end{aligned} \quad (3.7)$$

$$\begin{aligned} \mathcal{P}_\tau(\tau, y; \nu) &= e^{\frac{-y^2\gamma}{2\tau}} \left[\frac{-\bar{s}'_f(\tau; \nu)}{x_{\max}} + \frac{y^2}{2} f'_2(\tau; \nu) + yx_{\max}\mathcal{N}_\tau(\tau, y; \nu) \right] \\ &\quad + \frac{y^2\gamma}{2\tau^2} e^{\frac{-y^2\gamma}{2\tau}} \left[\frac{K - \bar{s}_f(\tau; \nu)}{x_{\max}} - yK + \frac{y^2}{2} f_2(\tau; \nu) + yx_{\max}\mathcal{N}(\tau, y; \nu) \right]. \end{aligned} \quad (3.8)$$

$$f'_2(\tau; \nu) = \left(-\frac{\gamma}{\tau^2 x_{\max}^2} [K - \bar{s}_f(\tau; \nu) + \tau \bar{s}'_f(\tau; \nu)] - \left(\frac{2\delta}{\sigma^2} + 1 \right) \bar{s}_f(\tau; \nu) - 2\mathcal{N}_{\tau y}(\tau, 0; \nu) \right) x_{\max}. \quad (3.9)$$

$\mathcal{N}_\tau(\tau, y; \nu)$ is the first derivative (in time) of the feed-forward DNN output, and $\bar{s}'_f(\tau; \nu)$ is the first derivative of the early exercise boundary. From the above equations, the following boundary conditions are satisfied:

$$\mathcal{P}_y(\tau, 0; \nu) = -\bar{s}_f(\tau; \nu), \quad \mathcal{P}_y(\tau, 1; \nu) \cong 0, \quad \mathcal{P}_y(0, y; \nu) = 0, \quad y > 0; \quad (3.10)$$

$$\mathcal{P}_\tau(\tau, 0; \nu) = \frac{-\bar{s}'_f(\tau; \nu)}{x_{\max}}, \quad \mathcal{P}_\tau(\tau, 1; \nu) \cong 0, \quad \mathcal{P}_\tau(0, y; \nu) = 0, \quad y > 0. \quad (3.11)$$

The left boundary value of the first derivative of the auxiliary function is a Neumann boundary condition which is also known as the smooth-pasting or high contact condition. Furthermore, we investigate the left boundary values of the value function and its first derivative in space. It is easy to see from (2.18) and (2.19) that we can construct another boundary condition as follows:

$$P_y(\tau, 0) - x_{\max}P(\tau, 0) = -K. \quad (3.12)$$

The boundary condition in (3.12) is called the Robin boundary condition. It is worth mentioning that considering (3.6) and (3.10), our auxiliary function has already mimicked this Robin boundary condition as follows:

$$\mathcal{P}_y(\tau, 0; \nu) - x_{\max}\mathcal{P}(\tau, 0; \nu) = -K. \quad (3.13)$$

Finally, we differentiate the auxiliary function twice and obtain

$$\begin{aligned} \mathcal{P}_{yy}(\tau, y; \nu) &= e^{\frac{-y^2\gamma}{2\tau}} [f_2(\tau; \nu) + 2x_{\max}\mathcal{N}_y(\tau, y; \nu) + yx_{\max}\mathcal{N}_{yy}(\tau, y; \nu)] \\ &\quad - 2\frac{y\gamma}{\tau} e^{\frac{-y^2\gamma}{2\tau}} [-K + yf_2(\tau; \nu) + x_{\max}\mathcal{N}(\tau, y; \nu) + yx_{\max}\mathcal{N}_y(\tau, y; \nu)] \\ &\quad + \left(\frac{y^2\gamma^2}{\tau^2} - \frac{\gamma}{\tau} \right) e^{\frac{-y^2\gamma}{2\tau}} \left[\frac{K - \bar{s}_f(\tau; \nu)}{x_{\max}} - yK + \frac{y^2}{2} f_2(\tau; \nu) + yx_{\max}\mathcal{N}(\tau, y; \nu) \right], \end{aligned} \quad (3.14)$$

where $\mathcal{N}_{yy}(\tau, y; \nu)$ is the second derivative (in space) of the feed-forward DNN output. With L'Hôpital's rule and fixed y , one can observe that

$$\lim_{\tau \rightarrow 0} \frac{y\gamma}{\tau} e^{\frac{-y^2\gamma}{2\tau}} = 0, \quad \lim_{\tau \rightarrow 0} \frac{y^2\gamma}{\tau^2} e^{\frac{-y^2\gamma}{2\tau}} = 0, \quad \lim_{\tau \rightarrow 0} \frac{y^2\gamma^2}{\tau^2} e^{\frac{-y^2\gamma}{2\tau}} = 0. \quad (3.15)$$

This is because the exponential part tends to zero faster than τ . Furthermore, it is easy to see from 3.14 that the far right boundary condition for the second derivative of the value function is satisfied. Moreover, we are now focused on the continuation region because we fix the free boundary. Hence, the discontinuity in the value function's first and second derivative (in space) is no longer visible; at most for $\tau > 0$. We refer the reader to the work of Ballestra (2018). To ensure that the second derivative of the auxiliary function satisfies the left boundary condition, we impose the boundary value of the second derivative of the value function on $\mathcal{P}_{yy}(\tau, 0; \nu)$ as follows:

$$\mathcal{P}_{yy}(\tau, 0; \nu) = \left[\frac{2rK}{\sigma^2} - \left(\frac{2\delta}{\sigma^2} + 1 \right) \bar{s}_f(\tau; \nu) \right] x_{\max}, \quad (3.16)$$

and then obtain

$$\left[\frac{2rK}{\sigma^2} - \left(\frac{2\delta}{\sigma^2} + 1 \right) \bar{s}_f(\tau; \nu) \right] x_{\max} = f_2(\tau; \nu) - \frac{\gamma}{\tau} \left[\frac{K - \bar{s}_f(\tau; \nu)}{x_{\max}} \right] + 2x_{\max} \mathcal{N}_y(\tau, 0; \nu). \quad (3.17)$$

Thus, in order to satisfy (3.17), we define $f_2(\tau; \nu)$ as presented in (3.4). If we observe the value function when $\tau = 0$, $K - s_f(0) = 0$. Moreover, from the work of Chen et al. (2013), a relation is established when $0 \leq \delta \leq r$ and $u \simeq T$ as follows:

$$K - s_f(u) \propto \sqrt{|t| \log(t)}. \quad (3.18)$$

Similarly, for $\delta > r$, Wilmott et al. (1995) further established a relation as given below

$$K - s_f(u) \propto \sqrt{(t)}, \quad (3.19)$$

Hence, for $\tau \simeq 0$,

$$K - s_f(\tau) = o(\tau), \quad (3.20)$$

resulting to infinite velocity at $\tau = 0$. We may further refer the reader to the work of van Moerbeke (1974). This attribute further ascertains the regularity of the expressions

$$\frac{\gamma}{\tau} \left[\frac{K - \bar{s}_f(\tau; \nu)}{x_{\max}} \right], \quad (3.21)$$

in (3.4) under exact representation. Furthermore, the exponential part in $\mathcal{P}(\tau, y; \nu)$ further ensures that the latter and its derivative tend to zero faster than τ . We observe during experiment that we can achieve an accurate DNN solution if we infinitesimally avoid $\tau = 0$. In this research, a stable solution is obtained up to $\tau_{\min} = 10^{-8}$. Finally, the auxiliary function can be reformulated as follows:

$$\mathcal{P}(\tau, y; \nu) = \mathcal{H}_1(\tau, y)K - \mathcal{H}_2(\tau, y)\bar{s}_f(\tau; \nu) + e^{\frac{-y^2\gamma}{2\tau}} y x_{\max} [\mathcal{N}(\tau, y; \nu) - y \mathcal{N}_y(\tau, 0; \nu)], \quad (3.22)$$

where

$$\begin{aligned} \mathcal{H}_1(\tau, y) &= e^{\frac{-y^2\gamma}{2\tau}} \left[\frac{1}{x_{\max}} - y + \left(\frac{\gamma}{\tau x_{\max}} + \frac{2rx_{\max}}{\sigma^2} \right) \frac{y^2}{2} \right], \\ \mathcal{H}_2(\tau, y) &= e^{\frac{-y^2\gamma}{2\tau}} \left(\frac{1}{x_{\max}} + \left[\frac{\gamma}{\tau x_{\max}} + \left(\frac{2\delta}{\sigma^2} + 1 \right) x_{\max} \right] \frac{y^2}{2} \right). \end{aligned}$$

It is also important to observe that at the left boundary, we have reduced our approximation only to the early exercise boundary equations. If the DNN approximation of the early exercise boundary converges to the exact early exercise boundary, the left boundary values of the auxiliary function and its derivative will converge to the left boundary values of the value function and its derivatives. Furthermore, with the formulation of (3.22), we have ensured that the boundary values associated with the value functions and their derivatives present in (2.15) are fully mimicked in pseudo-exact form (due to the early exercise boundary) by the auxiliary function and its derivatives such that

$$\frac{1}{T} \mathcal{P}_\tau(\tau, 0; \nu) - \frac{\sigma^2}{2x_{\max}^2} \mathcal{P}_{yy}(\tau, 0; \nu) - \frac{\xi_\tau}{x_{\max}} \mathcal{P}_y(\tau, 0; \nu) + r \mathcal{P}(\tau, 0; \nu) = 0, \quad (3.23)$$

$$\frac{1}{T}\mathcal{P}_\tau(\tau, 1; \nu) - \frac{\sigma^2}{2x_{\max}^2}\mathcal{P}_{yy}(\tau, 1; \nu) - \frac{\xi_\tau}{x_{\max}}\mathcal{P}_y(\tau, 1; \nu) + r\mathcal{P}(\tau, 1; \nu) \cong 0, \quad (3.24)$$

$$\frac{1}{T}\mathcal{P}_\tau(\tau_{\min}, y; \nu) - \frac{\sigma^2}{2x_{\max}^2}\mathcal{P}_{yy}(\tau_{\min}, y; \nu) - \frac{\xi_{\tau_{\min}}}{x_{\max}}\mathcal{P}_y(\tau_{\min}, y; \nu) + r\mathcal{P}(\tau_{\min}, y; \nu) \cong 0, \quad (3.25)$$

$$\xi_{\tau_{\min}} = r - \delta - \frac{\sigma^2}{2} + \frac{\bar{s}'_f(\tau_{\min}; \nu)}{T\bar{s}_f(\tau_{\min}; \nu)}, \quad \xi_\tau = r - \delta - \frac{\sigma^2}{2} + \frac{\bar{s}'_f(\tau; \nu)}{T\bar{s}_f(\tau; \nu)}. \quad (3.26)$$

Some of the Greek sensitivities are obtained directly from the auxiliary function and its derivatives after training. Furthermore, the early exercise boundary and its first derivative can easily be obtained from (3.5). Below, we highlight the limitation of the PINN method of Raissi et al. (2019) and the trial solution of Lagaris et al. (1998) and the significance of our proposed method for solving the presented non-linear and fixed-free boundary American options model:

- Several boundary values associated with the transformed model (value matching condition, smooth pasting condition, far right boundaries of the value function and its derivatives.) make it challenging to implement the trial solution method.
- The boundary values involve the early exercise boundary, which is not a priori known. Hence, we cannot achieve the exact nature of the trial solution method.
- We cannot account for the boundary conditions directly with the PINN method because of two unknowns $[\bar{s}_f(\tau; \nu)$ and $\mathcal{N}(\tau, 0; \nu)]$.
- Implicitly, our dual solution framework seeks to address this limitation. To this end, the first solution method comes from the auxiliary function, which include the DNN output and mimics in approximate form the boundary values of the value function due to the early exercise boundary.
- By reducing the boundary values of the auxiliary function (and its derivatives) to the early exercise boundary and its first derivative in time (see (3.6), (3.11), (3.12), and (3.16) above) we then implicitly construct a second solution method based on the value-matching condition for accounting for the free boundary. Please see (3.5).

3.2 Grid sampling and refinements

Solving the corresponding PDE on a discrete grid entails discretizing the domains and transforming the problem into an unconstrained minimization problem Lagaris et al. (1998). Generally, uniform distribution and equispaced grid sampling techniques are popular in literature; others include Latin hypercube sampling, Sobol sequence, Halton sequence, and Hammersley sequence (Wu et al., 2023). This research considers uniform grid sampling (with and without stretching), randomly structured grid, and random grid sampling (with and without stretching) techniques for generating our training dataset (See Figure 1). It is important to note that the distribution of these training points greatly impacts our model's computational efficiency and accuracy, which in turn affects the minimization of the loss function, early exercise boundary, value function, and Greeks. For the randomly structured grid, we first obtain a randomly spaced one-dimensional grid for the y and τ input from the uniform distribution. We then use the NumPy `meshgrid` function to duplicate the grids and form a two-dimensional domain. Hence, this makes the randomly generated grid to be structured. The stretched grid sampling was generated from the uniform grid. In mesh methods like the finite difference, several methods have been proposed for improving the accuracy and convergence rate of the options pricing model. Some of these approaches use different grid refinement strategy that includes local mesh refinement, adaptive mesh refinement, and grid stretching based on coordinate transformation. The idea primarily concentrates on the grid points close to the left boundary where singularity and discontinuity exist in the model. In this work, we implement a stretched grid sampling with a simple stretching as follows:

$$y_{\text{stretched}} = y_{\text{uniform}}^{\sqrt{p}}, \quad \tau_{\text{stretched}} = \tau_{\text{uniform}}^p, \quad p > 1. \quad (3.27)$$

Finally, we present another grid sampling that is random in y -direction and uniform in τ -direction and further stretched the sampled random grid using (3.27).

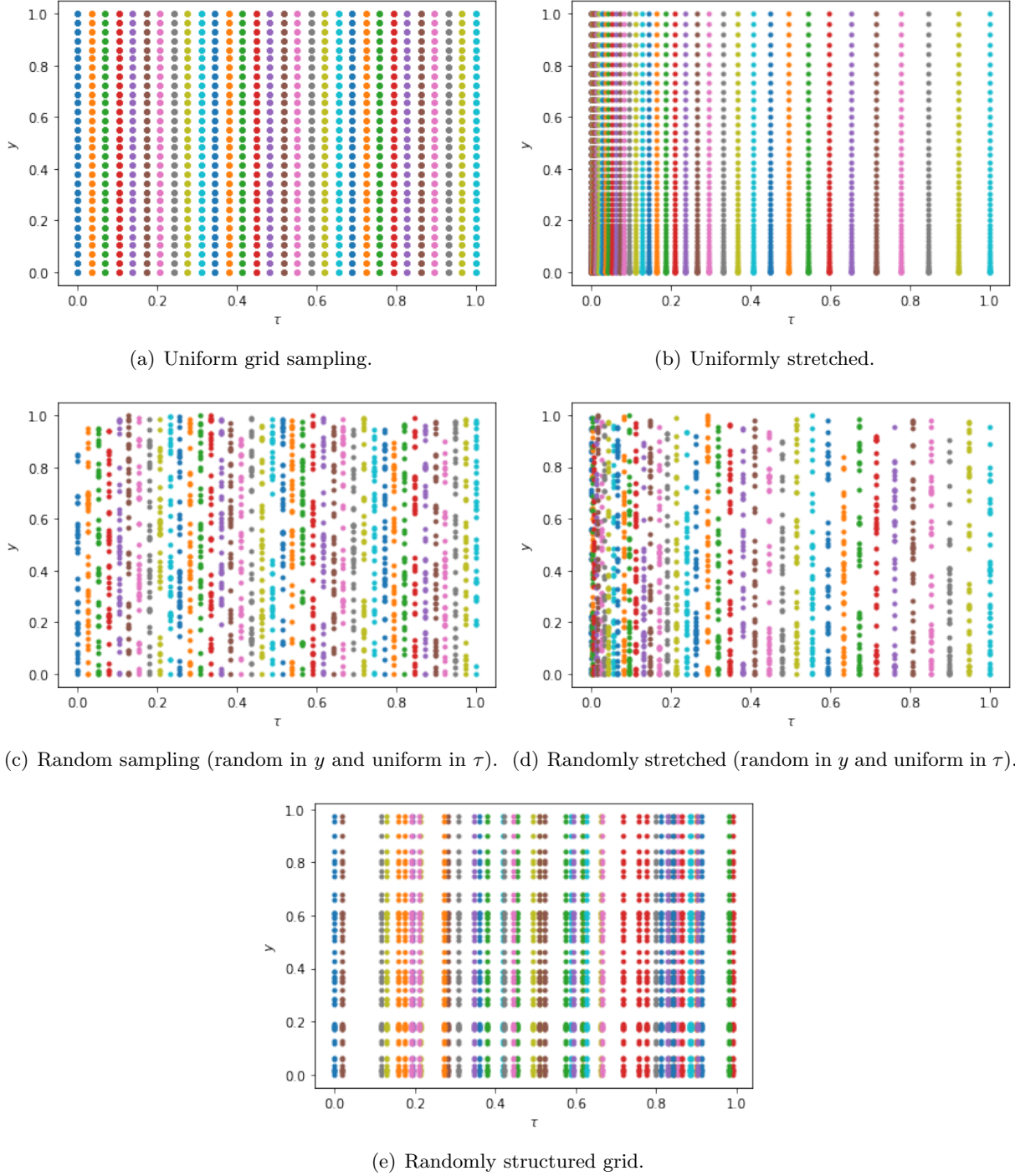


Figure 1: Grid sampling and refinement.

3.3 Loss function and optimization

This section introduces the loss function and the optimization techniques employed in minimizing the loss function and training the DNN. We estimate the unknown adjustable DNN parameter ν by minimizing the following loss function:

$$\mathcal{L}(\nu) = \frac{1}{N_\tau N_y} \sum_{j=1}^{N_y} \sum_{i=1}^{N_\tau} \left[\frac{1}{T} (\mathcal{P}_\tau)_{j,i} - \frac{\sigma^2}{2x_{\max}^2} (\mathcal{P}_{yy})_{j,i} - \frac{1}{x_{\max}} \xi_i (\mathcal{P}_y)_{j,i} + r \mathcal{P}_{j,i} \right]^2, \quad (3.28)$$

$$\xi_i = r - \delta - \frac{\sigma^2}{2} + \frac{1}{T\bar{s}_{f(i)}} \frac{d\bar{s}_{f(i)}}{d\tau},$$

for $i = 0, 1, \dots, N_\tau$, $j = 0, 1, \dots, N_y$. The loss or cost function is denoted as $\mathcal{L}(\nu)$ while N_τ and N_y are the total numbers of sample points in time and space domain, respectively. The output of the auxiliary function at the sample point (i, j) is given as $\mathcal{P}_{i,j}$ and will approximate the exact value function after training. Similarly $(\mathcal{P}_\tau)_{i,j}$, $(\mathcal{P}_y)_{i,j}$, and $(\mathcal{P}_{yy})_{i,j}$ represent the output of the derivatives of the auxiliary functions at the sample point (i, j) which will approximate the option Greeks after training. The process can be constructed into an optimization problem

$$\arg \min_{\nu} \mathcal{L}(\nu). \quad (3.29)$$

Training the DNN requires computing the gradient of the error function with respect to the input vectors and the DNN parameters. The partial derivatives of $\mathcal{P}(\tau, y; \nu)$ with respect to y , yy and τ are already presented in equations (3.7), (3.8), and (3.14). Consider a feed-forward multi-layer perceptron with one hidden layer; then the linear output is given by

$$\mathcal{N}(\tau, y; \nu) = \sum_{i=1}^M \zeta_i f(w_i \tau + n_i y + B_i), \quad (3.30)$$

where M is the total number of units in the DNN; ζ_i is the weight of the i^{th} hidden unit; w_i is the coefficient of the τ input to the i^{th} hidden unit; n_i is the coefficient of the y input to the i^{th} hidden unit; and B_i is the bias value of the i^{th} hidden neuron. Finally, for the hidden layers implemented in this work, we used $f(d) = (1 + e^{-d})^{-1}$, the sigmoid activation function. The derivatives of $\mathcal{N}(y, \tau; \nu)$ with respect to y , yy and τ are given below:

$$\mathcal{N}_\tau(\tau, y; \nu) = \sum_{i=1}^M \zeta_i w_i f(d_i)(1 - f(d_i)), \quad \mathcal{N}_y(\tau, y; \nu) = \sum_{i=1}^M \zeta_i n_i f(d_i)(1 - f(d_i)); \quad (3.31)$$

$$\mathcal{N}_{yy}(\tau, y; \nu) = \sum_{i=1}^M \zeta_i n_i^2 f(d_i)(1 - f(d_i))(1 - 2f(d_i)). \quad (3.32)$$

Figures 2 and 3 describe our deep learning procedure and less dense architecture, respectively. The following pseudocode (Algorithm 1) explains the techniques for approximating the early exercise boundary and its first derivatives, value function and Greeks.

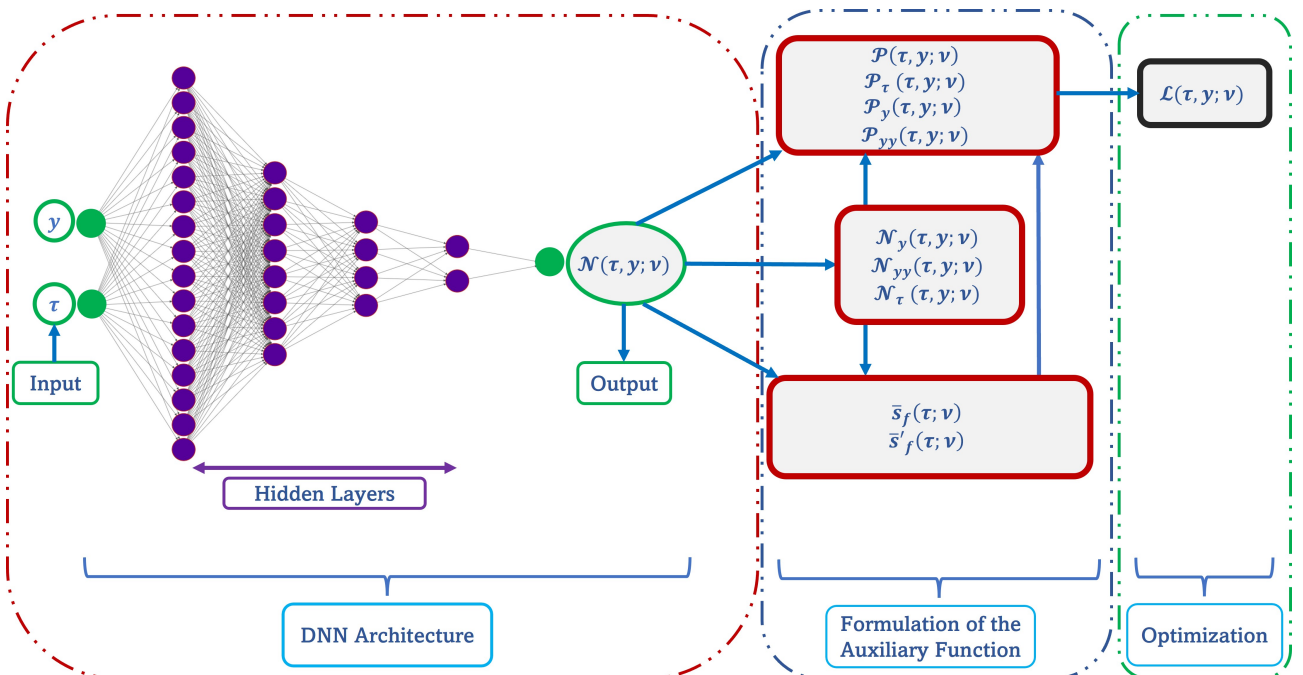
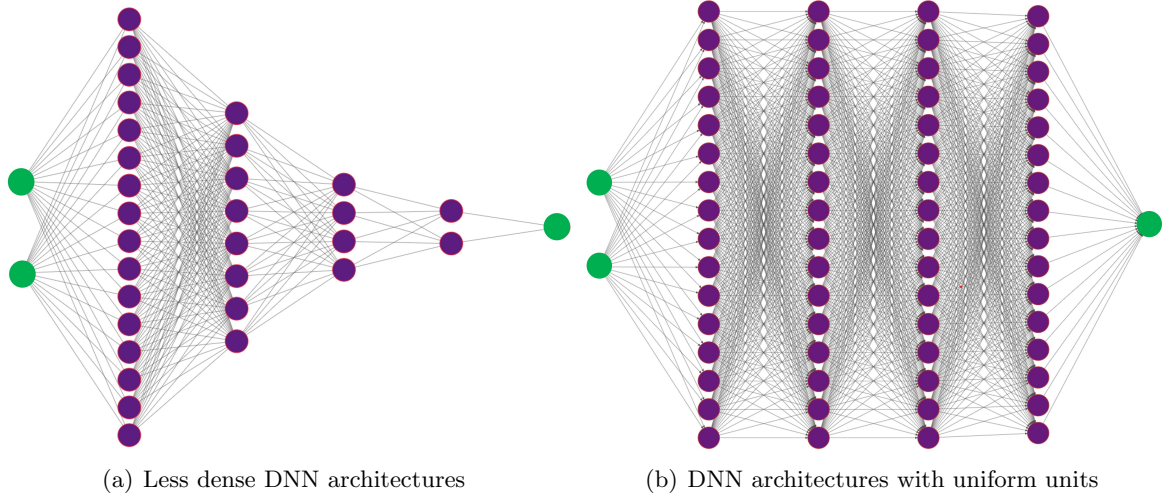


Figure 2: Flowchart of the DNN procedure.

Figure 3: Implemented DNN architectures ¹**Algorithm 1** Pseudocode for the optimal stopping problem implemented in Python

Import the `tensorflow` library and other packages. Initialize the American options parameters.

1: Building the DNN Architecture:

- Initialize and discretize the training and the prediction dataset into the required grid structure.
- Initialize the parameters of the feed-forward DNN, such as the learning rate, number of the input, hidden, and output units.
- Build the MLP and define the feed-forward flow $\mathcal{N}(\tau, y; \nu)$ with the prescribed architecture.

2: Functions Description:

- Extract the early exercise boundary function from the DNN output, i.e., $\bar{s}_f(\tau; \nu) = K - x_{\max} \mathcal{N}(\tau, 0; \nu)$.
- Extract the derivative of the early exercise boundary function from the first derivative (in time) of the DNN output, i.e., $\bar{s}'_f(\tau; \nu) = -x_{\max} \mathcal{N}_\tau(\tau, 0; \nu)$.
- Construct the auxiliary function $\mathcal{P}(\tau, y; \nu) = e^{\frac{-x^2 \gamma}{2\tau}} \mathcal{M}(\tau, y; \nu)$ as shown in equation 3.1; and construct the corresponding PDE as shown in equation 2.15 with their derivatives.

3: Training:

- Define the Loss function $\mathcal{L}(\nu)$ using the expression in 3.28; and determine the derivatives of $\mathcal{L}(\nu)$ with respect to the parameters.
- Train the DNN using the exponential decay learning rate and find optimal parameter ν' : For the learning rate $q \in (0, 1)$, adjust the parameters. Here, w = original weight, w' = adjusted weight, B = bias, B' = adjusted bias, $\mathcal{L}_w(\nu)$ = derivative of the loss function with respect to the weight, and $\mathcal{L}_B(\nu)$ = derivative of the loss function with respect to the bias.

$$\begin{cases} w \rightarrow w' &= w - q\mathcal{L}_w(\nu), \\ B \rightarrow B' &= B - q\mathcal{L}_B(\nu). \end{cases}$$

- 4: Results:** Generate the test data points and compute the errors, value function, option Greeks, and the early exercise boundary.
-

¹The authors credit the diagram for the DNN architecture in Figure 3 to this link source: <http://alexlenail.me/NN-SVG/index.html>

Remark. During the construction and training of the DNN, weight initialization is essential to drive the convergence of the model efficiently. Failure to do this might lead to vanishing or exploding gradient issues. Also, since the NN methods are applied to a series of problems that are both highly non-linear and complex, then diverse initialization techniques tend to give rise to fluctuating performances. These techniques are presented in Dolezel et al. (2016) and Narkhede et al. (2022). Weights are typically initialized to all zeros, all ones, from a normal distribution, uniform distribution, or even from techniques like the Xavier (Glorot) initialization (Glorot and Bengio, 2010). Following Umeorah and Mba (2022), this paper considers the random initialization from the normal distribution (Narkhede et al., 2022)

4 Experiment and Discussion

The experiments presented in this work were performed on a computer with an 11th Gen Intel Core i7-1165G7 processor running at 2.80 GHz on a 64-bit Windows 10 operating system. The only exception is the experiment in Table 15 which was conducted on a computer with a 12th Gen Intel(R) Core (TM) i7-12700H running at 2.30 GHz with 64-bit Windows 11 operating system. Furthermore, we used the free and open-source software library of toolboxes **Tensorflow** (v 2.3.0) designed by Abadi et al. (2016) to construct the DNN architecture and to train and optimize the model.

To extensively verify the performance of our proposed DNN method for solving the fixed free boundary American option model, we will conduct experiments with the dividend and non-dividend paying options parameters listed in Tables 1-3.

Table 1: First set of parameters for the non-dividend paying American options pricing problem.

Parameters	K	T	r	σ	$x_{\max}$
Short Time to Maturity (STM)	100	0.50	0.05	0.20	6.00
Medium Time to Maturity (MTM)	100	1.00	0.10	0.30	6.00
Long Time to Maturity (LTM)	100	3.00	0.08	0.20	6.00

Table 2: Second set of parameters for the dividend paying American options pricing problem.

Parameters	values
K	100
T	0.5, 1.0, 3.0
σ	0.1, 0.2, 0.3
δ	0.05
r	0.1
$x_{\max}$	6.0

Table 3: Third set of parameters for the non-dividend paying American options pricing problem.

Parameters	K	T	r	σ	$x_{\max}$
Values	1.00	1.00	0.10	0.20	2.00

The reason for investigating our model's performance with several examples presented above is to verify and ensure that it can efficiently handle problems with a short, medium, and long time to maturity using varying parameters while validating its solution accuracy.

4.1 Performance of grid samplings on the DNN solution

Here, we investigate the performance of the uniform, stretched uniform, randomly structured, random, and stretched random grid sampling on the solution accuracy of the proposed DNN method based on

the above parameters. For the stretched uniform grid sampling, we verify its performance using several grid sets. In all the experiments involved in this work, we use a training step of 20000. Furthermore, for the stretched uniform grid sampling, we ensure that $\tau_{\min}^p = 10^{-8}$. Moreover, for the stretched uniform grid solutions, across all subsections in this section, we use the stretched grid for training and uniform grid sampling for predicting. The only exception is in Table 8 and Figures 4 and 6 where we carried out both training and predicting procedure with the stretched grid to ensure we concentrate more predicting points near $\tau_{\min}$. In this subsection, our DNN architecture consists of four hidden layers. The number of units decreases across the hidden layers progressively. We considered 1 input layer (2×1 neurons); 4 hidden layers (1×512 neurons, 1×256 neurons, 1×128 neurons, 1×64 neurons); and 1 output layer (1×1 neuron). The only exception is Figures 5 and 4 where we use 1 input layer (2×1 neurons); 4 hidden layers (1×256 neurons, 1×128 neurons, 1×64 neurons, 1×32 neurons); and 1 output layer (1×1 neuron). Moreover, we observed that combining the Adam optimizer and sigmoid activation function with exponential learning rate decay minimizes our loss function more efficiently. For the Adam optimizer, the following parameters were used $\beta_1 = 0.99$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. It is important to mention, and as we will also see in the next subsection, that a slight change in β_1 greatly impacts the minimization of the loss function. The exponential learning rate decay further ensures that the minimization of our loss function is consistent, especially after large training steps have been carried out. For the exponential learning rate decay, we used the decay step of 2000 and the decay rate of 0.85, which entails decaying every 2000 steps with a decay rate of 0.85.

In our first experiment, we verify the solution accuracy of the early exercise boundary and its first derivative obtained from our proposed DNN method when $t = T$ and $\delta = 0$ using the parameters in Table 1. The results are compared with the one obtained from the sixth-order compact finite difference scheme and third-order Runge-Kutta adaptive time stepping with free boundary improvement (Nwankwo and Dai, 2022b). For the sixth-order compact finite difference scheme, we use a step size of $h = 0.005$ across all examples and comparisons. The results are display in Tables 4-9 and Figure 4.

Table 4: Early exercise boundary with (100×100) grid set ($t = T$, $\delta = 0$).

	Uniform	Randomly structured	Uniformly stretched ($p = 2.0$)	Sixth-order compact scheme
STM	84.27	84.52	84.02	83.92
MTM	76.19	76.23	76.16	76.16
LTM	81.84	81.78	81.78	81.78

Table 5: Early exercise boundary with different stretched (60×60) grid set ($t = T$, $\delta = 0$).

p	1.2	1.5	1.7
STM	84.20	84.04	83.91
MTM	76.26	76.17	76.16
LTM	81.88	81.80	81.78

Table 6: Early exercise boundary with stretched grid ($p = 2.5$, $t = T$, $\delta = 0$).

Grid set	(20×20)	(30×30)	(40×40)
STM	84.22	84.06	84.03
MTM	76.19	76.18	76.17
LTM	81.76	81.78	81.77

Table 7: Early exercise boundary with random grid sampling (40×40) grid set and MTM ($t = T$).

Grid sampling	Randomly stretched ($p = 2.5$)	Random ($p = 1$)
$s_f(t)$	76.16	76.18

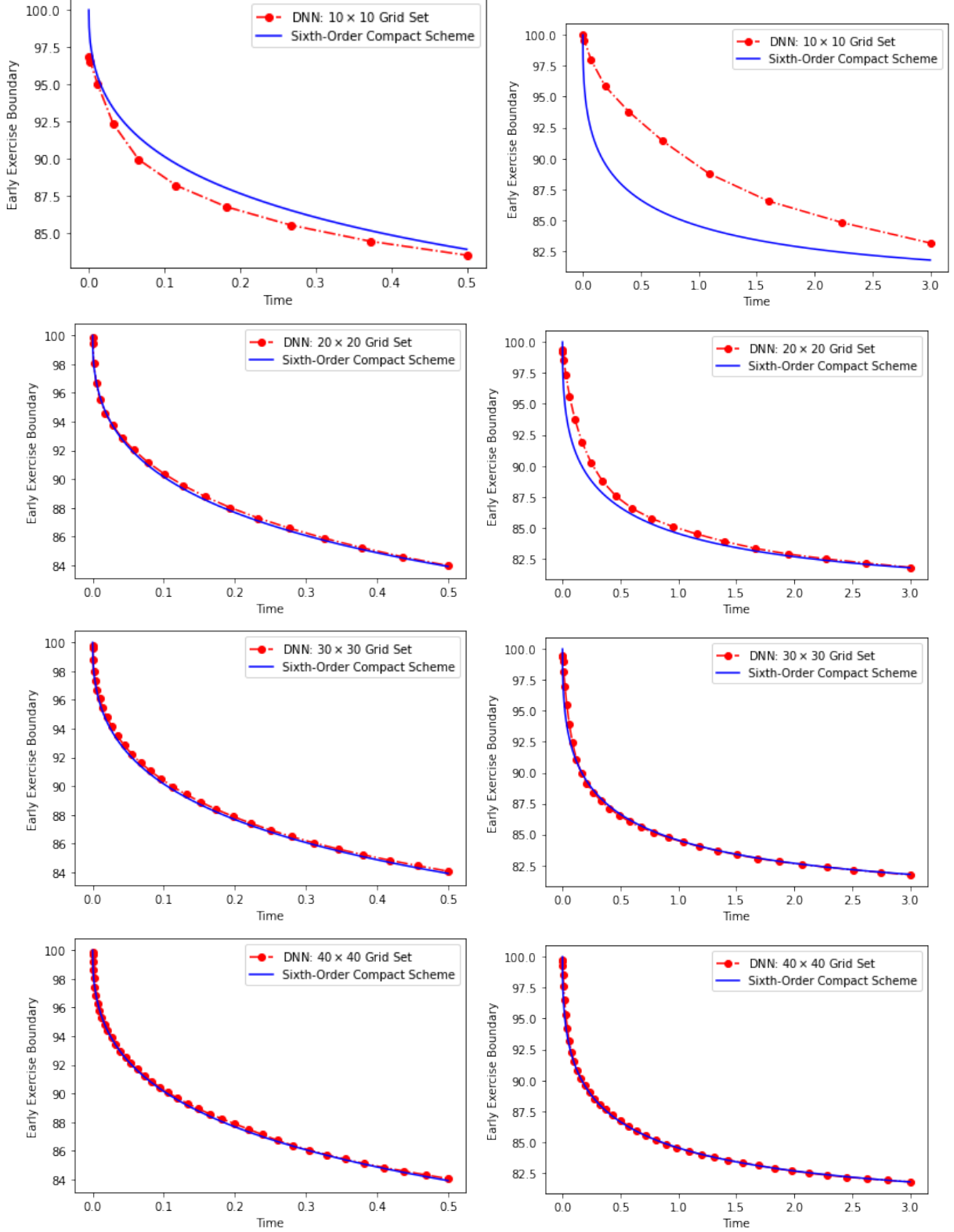


Figure 4: Early exercise boundary $s_f(t)$ in the LTM and STM scenarios using a uniformly stretched grid with $p = 2.5$. STM is shown on the left side, and LTM is on the right side. See Table 1 for a description of the scenarios. Early exercise boundary is plotted backward in time.

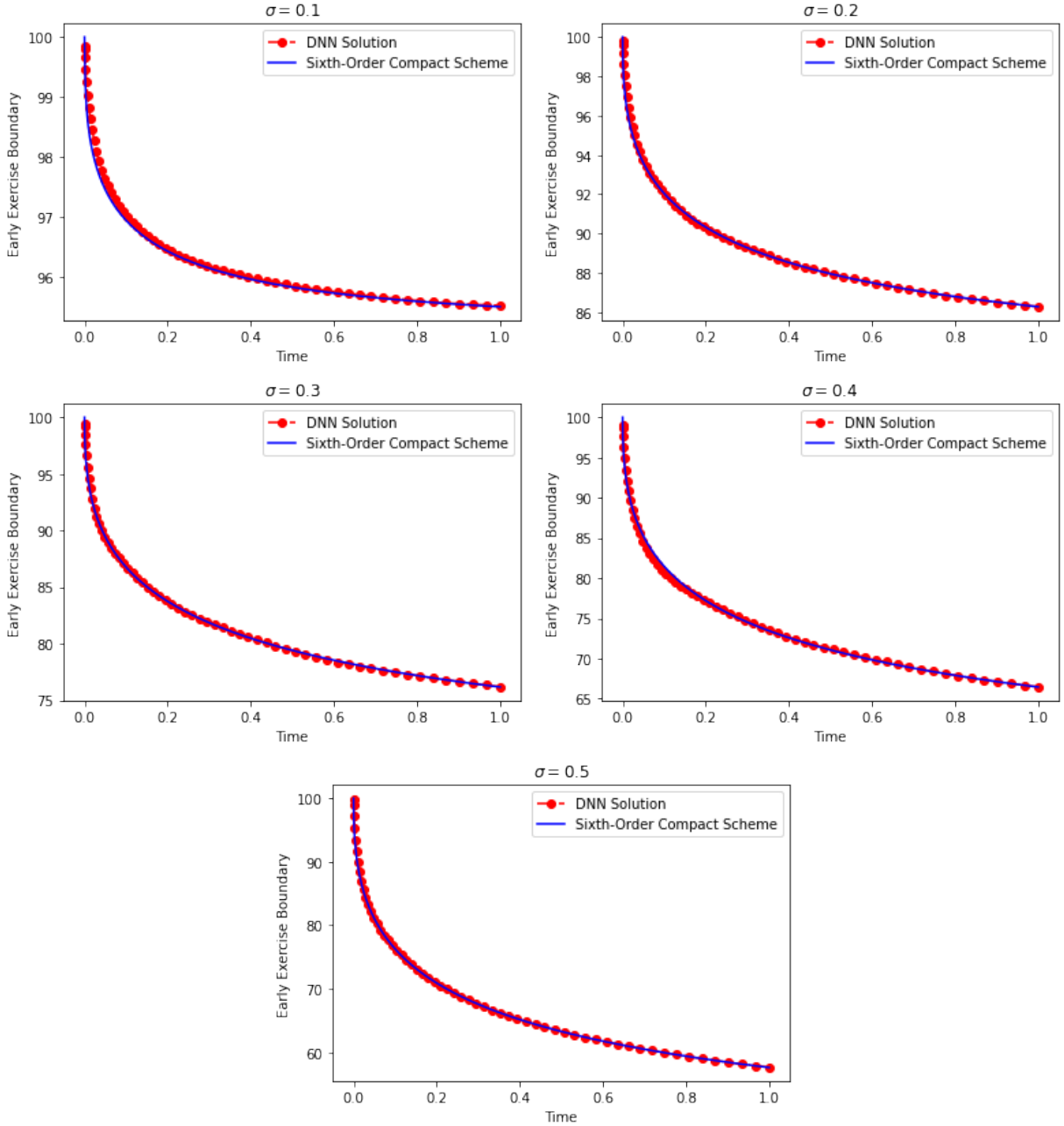


Figure 5: Comparing the early exercise boundary in the MTM scenarios with varying volatility $\sigma = 0.1, 0.2, 0.3, 0.4$ and 0.5 . For the DNN solution, we considered (60×60) uniformly stretched grid ($p = 2.0$). Early exercise boundary is plotted backward in time.

From Tables 4-7, the result using the stretched grid outperforms the ones obtained from the uniform grid and randomly structured grid sampling. For instance, the result obtained from the stretched grid with (60×60) grid set is much closer to the benchmark result when compared with the uniform and structured grid sampling using the (100×100) grid set. Even with a small training dataset, as displayed in Table 6 and 9, the result from the stretched grid is still reasonably accurate compared to the benchmark value. We further observed from 7 that the random grid sampling performs reasonably well with a small grid set of (40×40) . Moreover, from 7, we also observed that the result from the random grid sampling with (40×40) grid set is much better and the same as the result obtained from the sixth-order compact finite difference scheme when stretched. From the above findings, we can infer that even though random and uniform grid sampling provides reasonable DNN approximation, we can achieve better accuracy with a small training dataset if we concentrate more of those training points around the neighbourhood of the left boundary. Hence, grid sampling with stretching presents an optimal choice for our proposed DNN method.

Furthermore, from Figure 4, we observe that the plot of the early exercise boundary from the DNN output keeps converging around the one obtained from the highly accurate sixth-order compact finite difference scheme with free boundary improvement as we increase the dataset. With grid stretching, a little increase in the training dataset is required to achieve reasonable convergence. It is interesting to see that with a stretched grid set of (30×30) , we have a reasonable match between the early exercise boundary obtained from our DNN approximation when compared with the one obtained from the sixth-order compact finite difference scheme.

Also, using (60×60) grid set and varying volatility $\sigma = 0.1, 0.2, 0.3, 0.4$, and 0.5 , we also observe that there is a reasonable match between the result obtained from the DNN solution (with stretched grid) and the sixth-order compact finite difference scheme. Hence, we have shown that our proposed method gives reasonable accuracy with STM, MTM and LTM. When we fix the time to maturity (MTM) and vary the volatility, we also obtain a reasonable accuracy. In our future work, we hope to investigate pure random grid sampling with low discrepancy as described in the work of Wu et al. (2023). We hope to further investigate other grid refinement strategies that could improve the overall solution accuracy of our DNN approximation.

Moreover, it is possible for our results to vary slightly after each training. Here, we want to first ensure that our solution accuracy does not change significantly after each training. Secondly, we want to also understand how the statistical error changes when we increase the grid set. To this end, for a fixed grid set, we conduct a few experiments and generate sample solutions of the early exercise boundary when $t = T$. With the few experimental data obtained, we then compute statistical error estimates as shown in Table 8.

From Table 8, we observed that the mean values obtained from the samples of $s_f(T)$ across the presented grid sets are close to the result from the sixth-order compact finite difference scheme. Moreover, the mean value obtained with the (40×40) grid set gives the best result that is 4-digit accurate when compared with the result from the sixth-order compact finite difference scheme. Furthermore, the standard deviation, RMSE given as $= \sqrt{\frac{\sum_{i=1}^N [s_f(T) - \hat{s}_f(T)]^2}{N}}$, and absolute error (from the mean value of the DNN solution samples and the sixth-order compact scheme) obtain from the (40×40) is the smallest when compared with the (20×20) and (30×30) grid sets presented in Table 8. Another indication to observe here is that with grid stretching, an accurate result can always be achieved with small grid sets. From Figure 6, we also observe that as the grid set increases, the samples of $s_f(T)$ keep stabilizing and converging around the mean value and the result from the sixth-order compact finite difference scheme which reflect the reason why the obtained standard deviation, RMSE, and the absolute error is smallest with the (40×40) grid set. It is also very important to mention that as the grid set increases, it might be pertinent to either increase the number units in each hidden layer or reduce the stretching parameter p . Here, we can validate that the 40×40 grid set with $p = 2.5$ will provide very reasonable accuracy. What remains to be seen is whether we can further enhance accuracy by implementing an optimal DNN architecture and hyperparameter tuning. In conclusion, we can further validate that the bias and the standard deviation can be controlled by using more training points in the DNN architecture.

Table 8: Statistical error estimate with No. of samples $N = 25$. $\check{s}_f(T) = 81.7773$ is the benchmark value obtained from the sixth-order compact scheme.

Uniformly stretched grid set ($p = 2.5$)	(20×20)	(30×30)	(40×40)
Mean $[\hat{s}_f(T)]$	81.7648	81.7580	81.7754
Standard Deviation (SD)	0.06662	0.05663	0.02527
RMSE	0.06645	0.05874	0.02483
$ \hat{s}_f(T) - \check{s}_f(T) $	0.01247	0.01926	0.00190

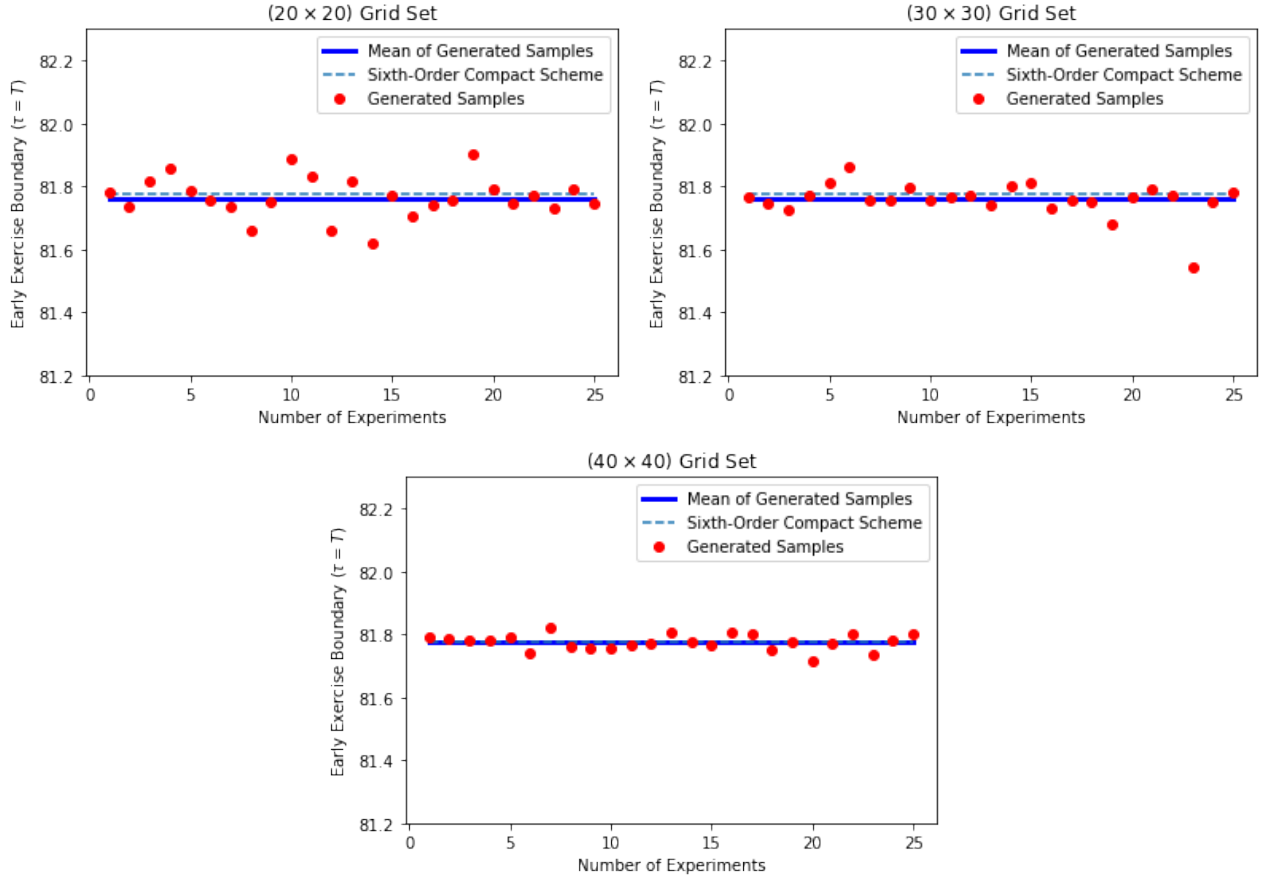


Figure 6: Distribution of computed samples of $s_f(T)$ around its mean value and solution of the sixth-order compact scheme.

In the second experiment, we consider dividend paying options with the parameters presented in Table 2 and verify the result of the early exercise boundary from the DNN approximation with the one obtained from fourth-order compact finite difference scheme in space and fifth-order 5(4) Runge-Kutta time integration method with free boundary improvement Nwankwo and Dai (2022a). The sixth-order finite difference scheme did not account for the dividend paying options, hence, the reason for comparing with the fourth-order compact finite difference scheme. The results for this experiment is listed in Table 9. We observe reasonable consistency with our DNN approximation of the early exercise boundary when compared with the result from the fourth-order compact finite difference scheme.

Table 9: Early exercise boundary for the dividend paying American options using uniformly stretched (30×30) grid set ($p = 2.5$, $t = T$, $r = 0.1$, $\delta = 0.05$).

Time to maturity	DNN solution			Compact scheme		
	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$
STM	93.84	84.60	75.58	93.86	84.57	75.40
MTM	93.13	81.84	71.16	92.99	81.92	71.17
LTM	92.05	78.26	65.29	91.99	78.29	65.31

Finally, for this subsection, we present the DNN result we obtained with the parameter in Table 3, which we compared with some of the well performing existing methods. The result is displayed in Table 10. We observed that our result with a stretched grid sampling ($p = 2.5$) is the same as the ones in the works of Fazio et al. (2021) and Egorova et al. (2014) up to 5 digits and Nielsen et al. (2002) up to 4 digits. It is also interesting that we achieved this result with a very small training data set i.e., (20×20) grid set. Moreover, it is worth mentioning that our result could insignificantly change

if retrained, as one can observe from Table 8 and Figure 6 due to the statistical error. In general, we can further validate from Table 10, the importance of the stretched grid sampling on the solution accuracy of our DNN approximation.

Table 10: Early exercise boundary with uniformly stretched (20×20) grid set ($p = 2.5$, $t = T$).

Methods	Nielsen et al. (2002)	Fazio et al. (2021)	Egorova et al. (2014)	DNN solution
$s_f(t)$	0.8622	0.8628	0.8628	0.8628

4.2 Hyperparameter tuning, exponential learning rate decay and the loss function

Hyperparameter tuning is an indispensable step in the ML algorithm, as this is essential for the ML algorithm to output the result with the highest accuracy. There are diverse ways of tuning the hyperparameters, including manual search, grid search, Bayesian optimization, etc. We employ the manual search optimization technique for this research, despite being intensive and exhaustive. Here, we check the performance of the Adam optimizer² on the loss function minimization for our DNN model. We slightly tune the values of the Adam optimizer and investigate how it impacts the loss function and solution accuracy. The first set of approaches we employed in this research was to examine the impact of using/not using the exponential learning rate decay on the training optimizer, and this was performed on the same number of the hidden layer, neurons, and across the uniformly stretched (60×60) grid set with $p = 1.7$. (For the architecture, see Table 11). Furthermore, it is worth mentioning that we use uniform grid sampling with stretching for this experiment.

We visualized these results across all time to maturity, i.e., the STM, MTM, and the LTM. The result we obtained from the comparison showed that the exponential rate decay significantly impacted the loss function and the solution accuracy. For Methods 2 and 3, the oscillations were quite intense, and this behaviour affected the final minimized values and the accuracy of the solution. Thus, it was observed that the exponential learning rate decay trained the network with the specified significant learning rate, then slowly decayed the rate until the realization of the local minima. Furthermore, it can be seen that the conventional Adam parameter might not be optimal for all models, implying that it could be model dependent. Hence, Method 1 performed better when compared to Methods 2 and 3, as displayed in Figure 7. The substantial improvement observed in the plot profile of Method 1 based on the implementation of the exponential learning rate and Adam parameter tuning was also reported in the work of Anderson and Ulrych (2022).

Table 11: Hyperparameter tuning with uniformly stretched (60×60) grid set ($p = 1.7$).

Hyperparameter	Method 1	Method 2	Method 3
Hidden Layer	4	4	4
Units	(512,256,128,64)	(512,256,128,64)	(512,256,128,64)
Learning rate	0.001	0.001	0.001
Training step	20000	20000	20000
Display step	1000	1000	1000
Decay step	2000	none	2000
Decay rate	0.850	none	0.850
β_1	0.990	0.900	0.900
β_2	0.999	0.999	0.999
ϵ_{adam}	10^{-8}	10^{-8}	10^{-8}

Next, we investigate the impact of the loss function when we implement Method 1 and then compare it with the results of implementing Methods 4-8. For Table 12, we vary and increase the hidden layers

²Adam is an optimization algorithm based on adaptive estimation of momentums of the first order and second order (Kingma Diederik and Adam, 2014). It has the reputation of converging rapidly and being computationally efficient.

and the units for Methods 4, 5, and 6. We equally fix the hidden layers and progressively decrease the number of units across the hidden layers for Methods 7 and 8, as displayed in Table 13. After the training of the NN, we choose the last loss function values for Method 1 and Methods 4-8 above. We compare the performance based on the lowest loss function values and the computational time of each of the algorithms and display these results in Figure 8. Regarding the plot for the loss function values (STM) in Figure 8, the value for Method 4 was significantly larger compared to the other methods, whereas Method 7 gave the least loss value. Thus, there is a percentage decrease of 94.78% when Method 4 (the highest) was compared to Method 7 (the lowest). Furthermore, we measured the impact of loss values on the MTM; we observed that the loss values were relatively small across all methods, though Method 1 gave the least of all. Similarly, for the LTM, the loss values were significantly small across all methods, though Method 5 gave the least of all.

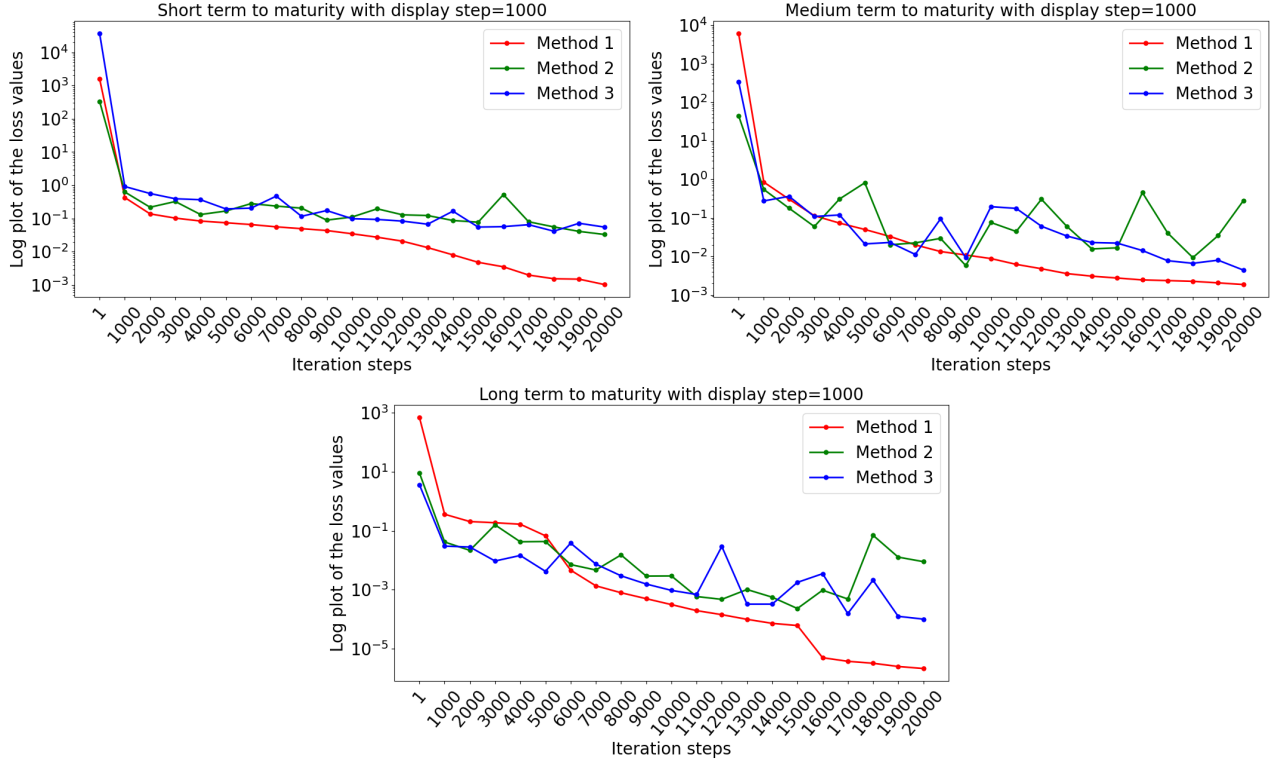


Figure 7: Log plot of loss function (3.28) values for various time to maturity. We used a uniformly stretched (60×60) grid set with $p = 1.7$.

Table 12: DNN architecture with fixed units and varying hidden layers. We used a uniformly stretched (60×60) grid set ($p = 1.7$).

Hyperparameter	Method 4	Method 5	Method 6
Hidden Layer	4	5	6
Units	(200,200,200,200)	(200,200,200,200,200)	(200,200,200,200,200,200)
Learning rate	0.001	0.001	0.001
Training step	20000	20000	20000
Display step	1000	1000	1000
Decay step	2000	2000	2000
Decay rate	0.850	0.850	0.850
β_1	0.990	0.990	0.990
β_2	0.999	0.999	0.999
ϵ_{adam}	10^{-8}	10^{-8}	10^{-8}

Table 13: DNN architecture with fixed hidden layer and varying units. We used a uniformly stretched (60×60) grid set ($p = 1.7$).

Hyperparameter	Method 7	Method 8
Hidden Layer	4	4
Units	(256,128,64,32)	(768,384,192,96)
Learning rate	0.001	0.001
Training step	20000	20000
Display step	1000	1000
Decay step	2000	2000
Decay rate	0.850	0.850
β_1	0.990	0.990
β_2	0.999	0.999
ϵ_{adam}	10^{-8}	10^{-8}

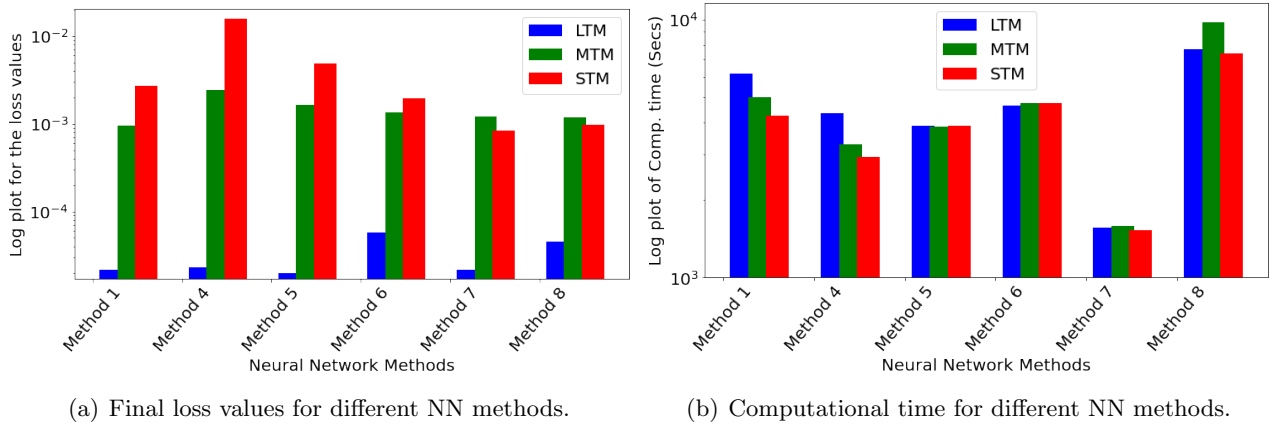


Figure 8: Comparison of loss function values and computational time with uniformly stretched (60×60) grid set ($p = 1.7$).

For the computational time, irrespective of the time to maturity, Method 7 was computationally inexpensive compared to other methods, and this was a result of fewer hidden layers and units used in building the model. Consider the MTM as an example; the following percentage decrease was observed when all the other methods were compared to Method 7: 68.4% (for Method 1), 51.02% (for Method 4), 58.9% (for Method 5), 66.8% (for Method 6) and 83.9% (for Method 8). Furthermore, we observe the impact of using fixed or varying units across the hidden layers. It is easy to observe the effect of this implementation based on the result obtained from Methods 4 and 7 in Figure 8. We observed that when we use a fixed unit of $(200,200,200,200)$ based on Method 4, the computational time is very high compared with Method 7 with varying units of $(256,128,64,32)$. However, both of them gave almost the same final loss function value. When Method 4 was further compared to Method 7, the following percentage decrease was observed: 94.78%, 51.02%, and 6.55% for STM, MTM, and LTM, respectively ³.

In the last experiment, Method 7 presents a better choice in terms of the overall computational time and final loss function value as described in Figure 8. However, we cannot validate that it is the optimal choice without carrying out further statistical analysis. Here, we want to further conduct an experiment to generate a few samples of the early exercise boundary (when $t = T$). With these sample solutions, we then obtain the RSME and standard deviation of these methods. This is in a similar fashion to the experiment in Table 8. Our interest here is to match the RSMEs and standard

³It is essential to mention that all the plot profiles in section 4.2 are in log scale. It enables better visualization of the LTM profile for the final loss function value and the computational time.

deviation obtained with these methods with their computational time and final loss function. It will enable us to ascertain the method that presents an optimal choice. To this end, we carry out this experiment using data from LTM with a uniformly stretched (60×60) grid set with $p = 1.7$. The obtained result is displayed in Table 14.

Table 14: Comparing the performance of different DNN architectures with LTM and uniformly stretched (60×60) grid set ($p = 1.7$). $s_f(t = T) = 81.7773$ is the benchmark value obtained from the sixth-order compact scheme.

Parameters	Method 1	Method 2	Method 3	Method 4
Mean $[\hat{s}_f(T)]$	81.77037	81.55878	81.75503	81.78272
SD	0.00524	0.41917	0.04476	0.02384
RSME	0.00479	0.38265	0.04086	0.02176
Final loss value	0.00443	0.56389	0.02934	0.00237
Training time(s)	4307.66	5032.72	4427.24	2879.55
	Method 5	Method 6	Method 7	Method 8
Mean $[\hat{s}_f(T)]$	81.77130	81.86920	81.76935	81.77661
SD	0.01771	0.15675	0.01371	0.00492
RSME	0.01616	0.14309	0.00979	0.00450
Final loss value	0.00119	0.00194	0.00179	0.00203
Training time(s)	4602.22	4811.10	1596.34	9763.41

We observed from Table 14 that Methods 1, 7, and 8, present the best results in terms of the mean value of the early exercise, RSME and standard deviation⁴. This is expected, and it further validates our first choice of implementing a DNN architecture whose unit progressively decays exponentially across the layer, as can be observed from Methods 1, 7, and 8 in Tables 11 and 13. More importantly, there is some level of consistency in the values generated from these Methods 1, 7, and 8. For instance, Method 7 (256, 128, 64, 32) with the smallest number of neurons has the lowest computational time and highest SD and RSME when compared with Method 1 (512, 256, 128, 64) and Method 8 (768, 384, 192, 96). Method 8 has the lowest SD and RSME, but the computational time is the highest when compared with the other Methods. The mean value of Method 8 is the closest to the result from the sixth-order compact finite difference and the same with the latter up to 5-digit accuracy. Method 4 has the second lowest computational time but performs substantially lower in terms of RSME and SD when compared with Method 1, which has the third lowest computational time. As a trade-off, we can finally validate that Method 1, which we used to carry out the experiment in the previous subsection, presents the optimal choice of DNN architecture. It is also worth mentioning that Method 2 presents the worst performance. During the training process, we observed the oscillating nature of the loss function values across each display step. The loss values kept oscillating and this impacted its accuracy upon the completion of the training process. The absence of the exponential learning rate decay in Method 2 accounted for its poor performance, and thus, we can further validate the importance of exponential learning rate decay in enhancing solution accuracy in NN methods.

Furthermore, we have also shown in Tables 6 and 10 and Figure 4, that with Method 1 and uniformly stretched (20×20) grid sets, we can still obtain reasonable accuracy. We also observed during experiment that with stretched (20×20) grid sets and Method 1, we can further reduce the number of unit across the 4 hidden layer and still obtain reasonable training time, prediction time, and solution accuracy. Also see Table 15.

⁴After performing several experiments, we obtained the average of the optimal exercise boundary (denoted by Mean $[\hat{s}_f(T)]$), SD, final loss value, and training time. The averaged values over the number of sample data are displayed in Table 14.

Table 15: Early exercise boundary for the dividend paying American options using Method 1 and varying varying smaller units and uniformly stretched (20×20) grid set ($p = 2.0$, $r = 0.1$, $\sigma = 0.3$, $T = 0.5$, and $\delta = 0.05$). $s_f(t = T) = 75.40$ is the benchmark obtained from the fourth-order compact finite difference scheme.

	DNN solution		
Units	(64,32,16,8)	(128,64,32,16)	(256,128,64,32)
$s_f(T)$	74.64	75.47	75.46
Training time (s)	167.60	380.65	691.80
Prediction time (s)	0.07	0.06	0.03

With an effective DNN architecture, what remain to be seen from our proposed method is that if all the options input parameters (r , σ , δ , K , T , etc.) and variables (S and t) are considered as input, can we use fewer dataset (with stretched, adaptive sampling, or other form of sampling and feature selections) to train the free boundary American option model faster and further make reasonable prediction faster after training? Achieving both gains will largely ascertain the effectiveness and practicability of our proposed deep learning methodology. We hope to carry out such comprehensive experiment in our future work. This is because on a broad-scale, one of the main advantage of deep learning procedure over existing methods lies in its predicting potential over range of option parameters after training. Once a given model is trained over large several inputs dataset that comprises the options parameters (r , σ , δ , K , T , etc.) and variables (S and t), the weights and biases can be re-used for future simultaneous prediction of the option value, early exercise boundary and hedge sensitivities. In conclusion, the main impression for the experiment in this subsection is to show that with an efficient DNN architecture, deep learning procedures can solve free boundary pricing problem with a very reasonable accuracy when compared with a more precise solution methods.⁵

Remark. We note that the NN presented in this paper are trained for one specific set of option parameters (r, δ, σ, T, K) at a time. Our goal has been to demonstrate the viability of our DNN solution framework for solving free boundary and nonlinear models. Future work will extend this approach to NNs that are trained to take options parameters ($r_1, \dots, r_{n_1}$, $\sigma_1, \dots, \sigma_{n_2}$, $\delta_1, \dots, \delta_{n_3}$, K , T , etc.) and variables ($S_1, \dots, S_{n_4}$, and t) as input parameters (for some positive integer number $n_1, \dots, n_4$) and thus once trained, be generally applicable (for pricing and risk management applications).

4.3 Impact of far-field boundary and near payoff conditions

It is well known that some errors are introduced when a far-field boundary is considered in the American options model. Moreover, by introducing $\tau_{\min}$; a point very close to the terminal point, another source of error could be introduced further. Here, with fixed $\tau_{\min} = 10^{-8}$, we first vary the value of $x_{\max}$ and observe its impact on the solution accuracy of the early exercise boundary. Next, we vary the value of $\tau_{\min}$ and carry out the same observation.

Furthermore, we consider the domain for which $(\tau, y) \in ([0, 1], [0, 1])$. In such case, we want to include $\tau = 0$ in our training. To this end, we first introduce an $\epsilon > 0$ such that

$$\mathcal{P}(\tau, y; \nu) = e^{\frac{-y^2\gamma}{2(\tau+\epsilon)}} \mathcal{M}(\tau, y; \nu), \quad (\tau, y) \in ((0, 1], [0, 1]); \quad (4.1)$$

with

$$f_2(\tau; \nu) = x_{\max} \left(\left[\frac{2rK}{\sigma^2} - \left(\frac{2\delta}{\sigma^2} + 1 \right) \bar{s}_f(\tau; \nu) \right] + \frac{\gamma}{(\tau + \epsilon)x_{\max}^2} (K - \bar{s}_f(\tau; \nu)) - 2\mathcal{N}_y(\tau, 0; \nu) \right). \quad (4.2)$$

⁵Based on symmetry properties, the American call options can be obtain from our DNN approximation of the American put options. Similarly, the early exercise boundary for the American call options can also be obtained from the DNN approximation of the early exercise boundary for the American put options. We refer the reader to the work of Detemple (2001).

Hence, in our experiment, we will further include another scenario for which $\tau_{\min} = 0$ which represent the implementation in (4.1) and (4.2).

For this experiment, we use Method 7. We further compare the result with the one obtained with Method 1 and the sixth-order compact finite difference scheme as shown in Tables 4 and 5. The result is displayed in Table 16. When we compare the result in Table 16 with the one obtained from the sixth order compact finite difference, it is easy to see that with fixed $\tau_{\min} = 10^{-8}$, we have reasonable accuracy when $x_{\max} \geq 4$. Furthermore, if we fix $x_{\max} = 6$, we observed that $\tau_{\min} = 0$ and $\tau_{\min} = 10^{-8}$ perform much better across different time to maturities given as STM, MTM, and LTM.

Table 16: Early exercise boundary with varying $x_{\max}$ and $\tau_{\min}$ ($t = T$). Implemented on a uniformly stretched (60×60) grid set ($p = 1.7$).

Maturity time	$x_{\max} = 6$	$x_{\max} = 5$	$x_{\max} = 4$	$x_{\max} = 3$	$x_{\max} = 2$
STM	83.91	84.02	84.04	83.81	83.95
MTM	76.17	76.15	76.17	76.19	79.92
LTM	81.80	81.77	81.79	83.19	90.39
	$\tau_{\min} = 0$	$\tau_{\min} = 10^{-8}$	$\tau_{\min} = 10^{-6}$	$\tau_{\min} = 10^{-4}$	$\tau_{\min} = 10^{-2}$
STM	83.97	83.91	83.86	83.88	83.89
MTM	76.15	76.17	76.21	76.20	76.22
LTM	81.76	81.80	81.78	81.76	81.83

4.4 Boundary reformulation and the auxiliary functions

Here, we verify the importance of our proposed novel auxiliary function, which satisfies, in pseudo-exact form, the boundary conditions imposed in the transformed and normalized American options. Our focus is on the left boundary point. Two common approaches exist in the literature for accounting for the boundary and initial condition in the loss function. One accounts for the boundary in the exact form, and the other is the PINN approach, where the boundary and initial values are generated directly from the NN output. Chen et al. (2020) mentioned that though the latter is easier to implement and extend to the high-dimensional problem and complicated domain, this approach does not always lead to exact satisfaction of the boundary and initial conditions and might require more computational effort. Our formulation, as presented in (3.1) and (3.2), is a novel hybrid method that integrates both approaches mentioned above for the efficient and simultaneous approximation of the early exercise boundary, value function, and Greeks. Simply, we use the first approach to present boundary values that are in pseudo-exact form due to the early exercise boundary and implicitly, implement a second solution method based on the value-matching condition for the approximate early exercise boundary and its first derivative directly from the feed-forward DNN output. The second solution method is similar to the PINN method for accounting for the boundary values in the loss function (though we have two unknowns). In this subsection, we present an alternative implementation where we assume we do not know the boundary value of the second derivative of the value function when $y = 0$ and extract the information directly from the DNN output by differentiating the latter twice with respect to y . Moreover, for this implementation, we take cognizant of the fact that the Greek variables are inherently sensitive, which will be more pronounced at the left boundary. To this end, we then present two different boundary scenarios and compare their performance extensively. Case 1 represents our formulation in (3.1)-(3.9), where we satisfy the boundary values in the pseudo-exact form as follows:

$$\mathcal{P}_y(\tau, 0; \nu) = -\bar{s}_f(\tau; \nu), \quad \mathcal{P}_{yy}(\tau, 0; \nu) = \left[\frac{2rK}{\sigma^2} - \left(\frac{2\delta}{\sigma^2} + 1 \right) \bar{s}_f(\tau; \nu) \right] x_{\max}. \quad (4.3)$$

In Case 2, the left boundary value of the second derivative of our auxiliary function is directly obtained from the feed-forward DNN output as follows:

$$\mathcal{P}_y(\tau, 0; \nu) = -\bar{s}_f(\tau; \nu), \quad \mathcal{P}_{yy}(\tau, 0; \nu) = \mathcal{N}_{yy}(\tau, 0; \nu); \quad (4.4)$$

with

$$f_2(\tau; \nu) = x_{\max} \left(\frac{\gamma}{\tau x_{\max}} (K - \bar{s}_f(\tau; \nu)) + \mathcal{N}_{yy}(\tau, 0; \nu) - 2x_{\max} \mathcal{N}_y(\tau, 0; \nu) \right), \quad (4.5)$$

$$f'_2(\tau; \nu) = x_{\max} \left(-\frac{\gamma}{\tau x_{\max}} (K - \bar{s}_f(\tau; \nu) + \tau \bar{s}'_f(\tau; \nu)) + \mathcal{N}_{\tau yy}(\tau, 0; \nu) - 2x_{\max} \mathcal{N}_{\tau y}(\tau, 0; \nu) \right). \quad (4.6)$$

We compare the performance of these two boundary scenarios on the solution accuracy and the final loss function value using LTM, Method 1, (60×60) stretched grid set with $p = 2.5$ and 20000 training steps. The results are listed in Table 17.

Table 17: Solution accuracy and final loss function value based on (4.3) and (4.4) ($t = T$).

Parameter	Case 1 (4.3)	Case 2 (4.4)
$s_f(t)$	81.773	81.788
Final loss function value	4.437×10^{-3}	4.170×10^{-2}

From Table (17), we observe that the final loss function value is lower in Case 1 when compared with Case 2. Chen et al. (2020) further mentioned in their literature that due to the sensitivity of the boundary and initial conditions, our approach in Case 1 can reduce the effort required during the training of the NN. Furthermore, the value of the early exercise boundary obtained using the sixth-order compact finite difference scheme is 81.777. We observed that the early exercise boundary obtained from Case 1 in (4.3) is more accurate when compared with the one obtained from Case 2 in (4.4). Hence, we validate that our first implementation in (3.1) provides a more accurate solution. However, the importance of Case 2 cannot be undermined, especially when the boundary value of the second derivative is not known or accounted for. For example, in the existing literature that solves the American-style stochastic volatility model with the front-fixing approach, the boundary value for the second derivative was not accounted for (Zhu and Chen, 2011). The approach implemented in Case 2 can be used to generate the boundary value of the second derivative from the feed-forward DNN output, and the latter can be compared with any existing exact boundary value under investigation. This phenomenon is left for future research, and we hope to study it further when solving free boundary options involving stochastic volatility and interest rate using our proposed deep learning method.

4.5 Investigating the sub-functions in the auxiliary operator with utility functions

Our main focus in this subsection is to further investigate if our implemented auxiliary operator that approximates the free boundary, value function and Greeks is final. Here, we seek to understand if we can reformulate $z(\tau, y)$ and $f_i(\tau; \nu)$ and what impact can it have on the model and solution accuracy. To this end, we present two sets of investigations. In the first investigation, our interest is to reformulate $z(\tau, y)$ with four utility functions

$$1 - e^{-yx_{\max}}, \quad e^{yx_{\max}} - 1, \quad \sinh(yx_{\max}), \quad \log(1 + yx_{\max}); \quad (4.7)$$

such that $f_1(\tau; \nu) = -K$ (similar to initial implementation) and

$$\mathcal{P}_y(\tau, 0; \nu) = -K + x_{\max} \mathcal{N}(\tau, 0; \nu) = -\bar{s}_f(\tau; \nu). \quad (4.8)$$

Here, $f_2(\tau; \nu)$ will vary for each set-up. Even though these utility functions tend to 0 as $y \rightarrow 0$, which is similar to our initial implementation ($z(\tau, y) = yx_{\max}$), it is important to mention that the speed and direction to which they tend to zero (which is one of the critical points in our input domain) differs. Please see Figure 9. Moreover, the derivative of these various functions differs, which is the main reason why $f_2(\tau; \nu)$ varies for each set-up.

Furthermore, careful observation of the first derivative of the auxiliary function and its boundary value in (3.7) and (4.8) reveals that part of the boundary value is included in the DNN component of the

auxiliary operator. Our objective in the second investigation is to reformulate $z(\tau, y)$ with another two utility functions

$$\frac{x_{\max}^2 y^2}{2}, \quad \ln(\cosh(yx_{\max})). \quad (4.9)$$

such that $f_1(\tau; \nu) = -s_f(\tau)$. $f_2(\tau; \nu)$ will be the same for each set-up in the second investigation. The above last two utility functions enable us to detach the DNN output part in the auxiliary function from the boundary value(s) part. We envisage that it will be useful when we encounter more complicated boundary value(s) such that we can avoid interaction of the latter with the DNN component present in the auxiliary function. The rest of the phenomenon is left for future research. In Tables 18 and 19, we display our findings for these two investigations. Furthermore, we present the plots of these utility functions close to $y = 0$ in Figure 9.

Table 18: Early exercise boundary with the first investigation ($\phi = 100$, $t = T$). Implemented with stretched (20×20) grid set and LTM.

$z(\tau, y)$	$1 - e^{-yx_{\max}}$	$e^{yx_{\max}} - 1$	$\sinh(yx_{\max})$	$\log(1 + yx_{\max})$
$s_f(t)$	81.83	81.69	81.43	81.79

Table 19: Early exercise boundary with the second investigation ($t = T$). Implemented with stretched (20×20) grid set and LTM.

$z(\tau, y)$	$\frac{x_{\max}^2 y^2}{2}(\phi = 30)$	$\ln(\cosh(yx_{\max}))(\phi = 40)$
$s_f(t)$	81.81	81.93

From Tables 18 and 19, we observe a reasonable solution accuracy with these utility functions using a very small grid set. Considering the overall observations and investigations, we intend to strongly mention that our initial implementation as described in (3.1)-(3.4) is not final, and the sub-functions present in the auxiliary function can be reformulated which could be model specific. In our future research, we hope to further investigate this phenomenon when solving other options pricing models.

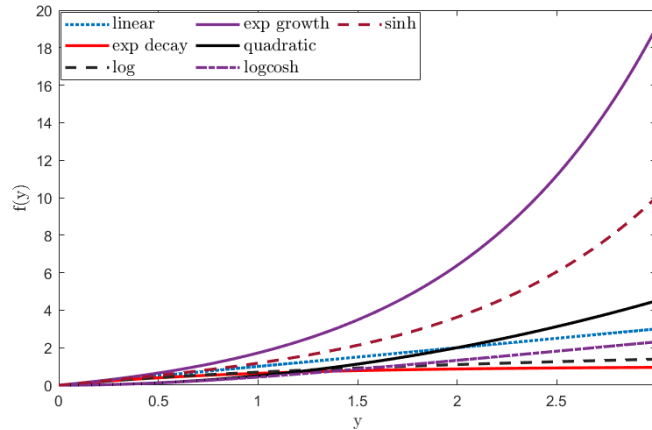
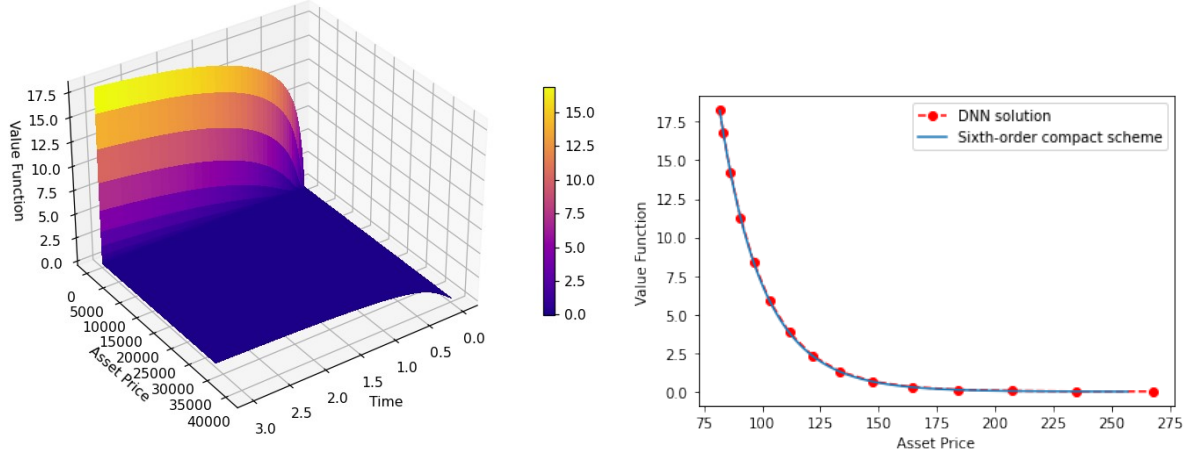


Figure 9: Plots of presented utility functions close to $y = 0$.

4.6 Value function and Greek sensitivities

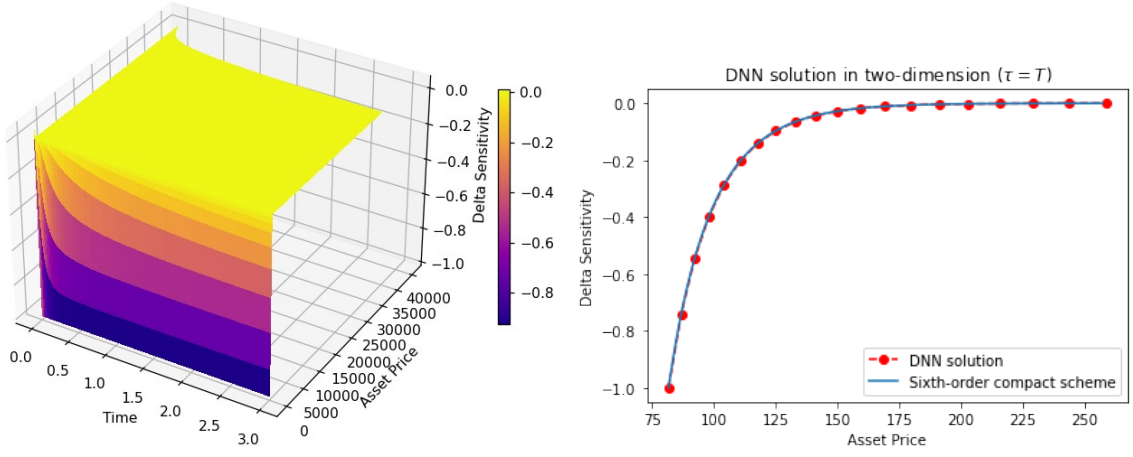
In this last subsection, we verify the solution accuracy of our proposed DNN method with respect to the option value and Greek sensitivities. We further compare our DNN result with the one obtained from the sixth-order compact finite difference scheme. The value function and option Greeks are computed from the auxiliary function and its derivatives based on (3.1)-(3.14). The plot profile of the value function and the option Greeks are displayed in Figures 10-12.



(a) Value function in three-dimensional axis.

(b) Comparison of the value function when $t = T$.

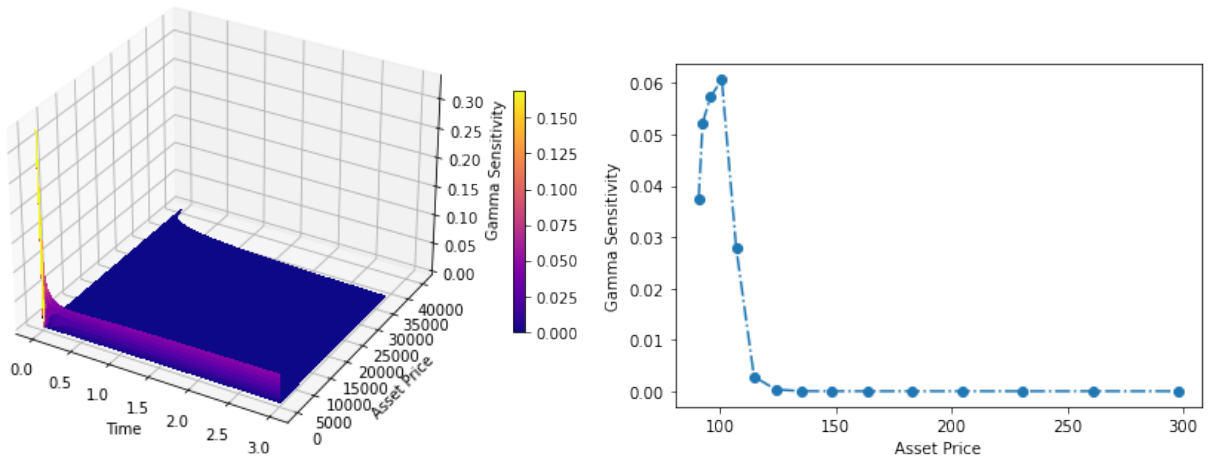
Figure 10: Solution of the value function with LTM.



(a) Delta sensitivity in three-dimensional axis.

(b) Comparison of the value function when $t = T$.

Figure 11: Solution of the Delta sensitivity with LTM.



(a) Gamma sensitivity in three-dimensional axis.

(b) Gamma sensitivity when $t = 3.3 \times 10^{-2}$.

Figure 12: Solution of the Gamma sensitivity with LTM.

From Figures 10 and 11, the plot of the value function and Delta sensitivity from the DNN solution coincides reasonably well with the one from the sixth-order compact finite difference scheme. Similarly, we observed the same for the delta sensitivity. We did not compare the result of the Gamma sensitivity; however, the smoothness of the latter can easily be observed from Figure 12. In conjunction with the result we obtain from other subsections, we can validate that our proposed DNN method could provide a more efficient approach for solving the free boundary option pricing PDE model, which could be very useful in a high-dimensional context.

4.7 Extending model normalizations

Here, we want to investigate the importance of normalizing the transformed model as presented in (2.15). Here, we present two different implementations where we only normalize in the τ direction with the new domain given as $(\tau, x) \in ([\tau_{\min}, 1], [0, x_{\max}])$.

In the first implementation here, the transformed fixed free boundary PDE for the American options model is given as

$$\frac{1}{T}P_\tau(\tau, x) - \frac{\sigma^2}{2}P_{xx}(\tau, x) - \left(r - \delta - \frac{\sigma^2}{2}\right)P_x(\tau, x) - \frac{1}{T}\frac{s'_f(\tau)}{s_f(\tau)}P_x(\tau, x) + rP(\tau, x) = 0, \quad x > 0. \quad (4.10)$$

Here, for brevity, we skip the boundary and initial value. The auxiliary operator in this scenario is given below

$$\mathcal{P}(\tau, x; \nu) = e^{\frac{-x^2\gamma}{2\tau}} \left(\sum_{i=0}^m \frac{f_i(\tau; \nu)x^i}{i!} + z(\tau, x)\mathcal{N}(\tau, x; \nu) \right), \quad (\tau, x) \in ((\tau_{\min}, 1], [0, x_{\max}]); \quad (4.11)$$

$$z(\tau, x) = x, \quad f_0(\tau; \nu) = K - \bar{s}_f(\tau; \nu), \quad f_1(\tau; \nu) = -K; \quad (4.12)$$

$$f_2(\tau; \nu) = \frac{2rK}{\sigma^2} - \left(\frac{2\delta}{\sigma^2} + 1\right)\bar{s}_f(\tau; \nu) + \frac{\gamma}{\tau}(K - \bar{s}_f(\tau; \nu)) - 2\mathcal{N}_x(\tau, 0; \nu). \quad (4.13)$$

For the second implementation, we further carry out another transformation such that

$$W(\tau, x) = \frac{P(\tau, x)}{K}, \quad b_f(\tau) = \frac{s_f(\tau)}{K}, \quad \ln S - \ln b_f \quad (\tau, x) \in ([\tau_{\min}, 1], [0, x_{\max}]). \quad (4.14)$$

The reason for the second transformation in (4.14) is to keep all asset price variables in a unit dollar. The fixed free boundary PDE equation is given below

$$\frac{1}{T}W_\tau(\tau, x) - \frac{\sigma^2}{2}W_{xx}(\tau, x) - \left(r - \delta - \frac{\sigma^2}{2}\right)W_x(\tau, x) - \frac{1}{T}\frac{b'_f(\tau)}{b_f(\tau)}W_x(\tau, x) + rW(\tau, x) = 0, \quad x > 0. \quad (4.15)$$

The auxiliary operator in this scenario is given below

$$\mathcal{P}(\tau, x; \nu) = e^{\frac{-x^2\gamma}{2\tau}} \left(\sum_{i=0}^m \frac{f_i(\tau; \nu)x^i}{i!} + z(\tau, x)\mathcal{N}(\tau, x; \nu) \right), \quad (\tau, x) \in ((\tau_{\min}, 1], [0, x_{\max}]); \quad (4.16)$$

$$z(\tau, x) = x, \quad f_0(\tau; \nu) = 1 - \bar{b}_f(\tau; \nu), \quad f_1(\tau; \nu) = -1; \quad (4.17)$$

$$f_2(\tau; \nu) = \frac{2r}{\sigma^2} - \left(\frac{2\delta}{\sigma^2} + 1\right)\bar{b}_f(\tau; \nu) + \frac{\gamma}{\tau}(1 - \bar{b}_f(\tau; \nu)) - 2\mathcal{N}_x(\tau, 0; \nu). \quad (4.18)$$

Here, $\bar{b}_f(\tau; \nu)$ is the DNN approximation of the early exercise boundary in a unit dollar. Also, the option value here is in a unit dollar. After approximation, we can recover the actual early exercise boundary by multiplying $\bar{b}_f(\tau; \nu)$ with K , i.e., $\bar{s}_f(\tau; \nu) = \bar{b}_f(\tau; \nu)K$. This formulation will further enable the reusing of the optimized parameters (weight and biases) of the DNN approximation. Because we no longer consider a unit domain in the space direction, we implemented a slightly different grid

stretching approach. To concentrate points in the locality of $x = 0$, we implement grid stretching based on sinh function as follows

$$x_{stretched} = c_k \sinh \left[\operatorname{arcsinh} \left(\frac{x_{\max}}{c_k} \right) \left(\frac{x}{x_{\max}} \right) \right], \quad (4.19)$$

where c_k is a positive constant. In Figure 13 below, we display the uniformly stretched grid with sinh function in the x direction as shown in (4.19).

Another significant change in this last normalization procedure is that our feed-forward DNN output was constrained to values in the range of 0 and 1. We noticed that an efficient DNN approximation is consistently achieved with this constraint. In Table 20, we present the result of the early exercise boundary obtained using these transformations by considering the second experimental data as given in Table 3. Here we use 4 hidden layers with (128,64,32,16) units. We also considered (40×40) grid set that is stretched with $p = 2.5$ in the τ direction and $c_k = 0.4$ in the x direction. For convenience, we will denote our first implementation here in (4.11)-(4.13) as Scenario 1 with the second implementation in (4.16)-(4.18) as Scenario 2. For the sake of brevity, we will only investigate the result from Scenario 2 and chose $\gamma = 1/\sigma\sqrt{T}$.

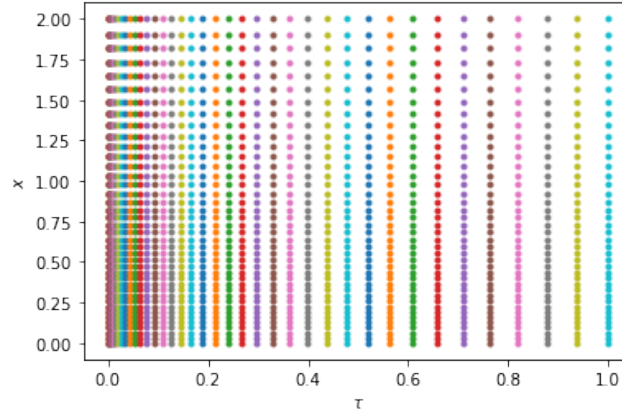


Figure 13: Stretching with sinh function in x direction ($c_k = 0.4$) and power function in τ direction ($p = 2.5$).

Table 20: $b_f(t)$ using stretched (40×40) grid set with $p = 2.5$, $c_k = 0.4$, $\gamma = 1/\sigma\sqrt{T}$, $T = 3$, and $x_{\max} = 2$ ($t = T$).

Parameters	$(\sigma = 0.2, r = 0.08)$	$(\sigma = 0.6, r = 0.08)$	Sixth order compact scheme $(\sigma = 0.6, r = 0.08)$
$b_f(t)$	0.8176504	0.377393	0.377586
Final loss value	3.3×10^{-6}	2.1×10^{-4}	None
$s_f(t)$ with $K = 100$	81.76504	37.73930	37.75856

It is worth observing the solution accuracy we obtain from Table 20 even though we used a very small number of neurons and grid set for this particular experiment when we compare it with our previous experiments. This is intentional as we observe that this particular normalization procedure enables reasonable solution accuracy with a small number of neurons and grid set. Moreover, the final loss function values were reasonably small in both low- and high-volatility cases presented in Table 20. The reason for investigation with low and high volatility values is to ensure that there is no susceptibility with varying volatility parameters.

4.8 American options with transaction cost

In this subsection, we further extend our approach for solving the American options pricing model with transaction cost which can be seen as a free boundary local volatility pricing model. We would

love to mention here that our implementations is to demonstrate the capacity and potential of our proposed method in addressing strongly nonlinear options pricing problems. For the transaction cost, we will consider risk-adjusted pricing methodology (RAPM) model.

In the original setting with $t = T - u$, the options model is given below:

$$V_t(t, S) - \bar{\sigma}^2 [t, S, V(t, S), \dots, V_{SS}(t, S)] \frac{S^2}{2} V_{SS}(t, S) - (r - \delta)SV_S(t, S) + rV(t, S) = 0, \quad S > s_f(t). \quad (4.20)$$

Here,

$$\bar{\sigma}^2(t, S, V(t, S), \dots, V_{SS}(t, S)) = \sigma^2 \left(1 + v [SV_{SS}(t, S)]^{\frac{1}{3}}\right), \quad (4.21)$$

where σ is the constant volatility presented in the American options pricing model without transaction cost, and v is a constant positive number.

Here, we want to demonstrate that with the Case 2 approach, which we described in (4.4) of subsection 4.4, we can efficiently solve the model in (4.20). Fixing the free boundary with the Landau logarithmic transformation with further normalization similar to our initial implementation in (2.5) and (2.14), we will obtain as follows:

$$\begin{aligned} \frac{x_{\max}}{T} P_\tau(\tau, y) - \frac{1}{x_{\max}} \frac{\bar{\sigma}^2(\tau, y, P(\tau, y), \dots, P_{yy}(\tau, y))}{2} P_{yy}(\tau, y) - \left(r - \delta - \frac{\bar{\sigma}^2}{2}\right) P_y(\tau, y) \\ - \frac{1}{T} \frac{s'_f(\tau)}{s_f(\tau)} P_y(\tau, y) + r x_{\max} P(\tau, y) = 0, \end{aligned} \quad (4.22)$$

with

$$\bar{\sigma}^2(\tau, y, P(\tau, y), \dots, P_{yy}(\tau, y)) = \sigma^2 + \sigma^2 v \left(\frac{1}{e^{y x_{\max} s_f(\tau)}} \left[\frac{1}{x_{\max}} P_{yy}(\tau, y) - P_y(\tau, y) \right] \right)^{\frac{1}{3}}. \quad (4.23)$$

Substituting (4.23) to (4.22), we will obtain

$$\begin{aligned} \frac{x_{\max}}{T} P_\tau(\tau, y) - \frac{1}{2x_{\max}} \left[\sigma^2 + \sigma^2 v \left(\frac{1}{e^{x_{\max} s_f(\tau)}} \left[\frac{1}{x_{\max}} P_{yy}(\tau, y) - P_y(\tau, y) \right] \right)^{\frac{1}{3}} \right] P_{yy}(\tau, y) \\ - \left(r - \delta - \frac{1}{2} \left[\sigma^2 + \sigma^2 v \left(\frac{1}{e^{x_{\max} s_f(\tau)}} \left[\frac{1}{x_{\max}} P_{yy}(\tau, y) - P_y(\tau, y) \right] \right)^{\frac{1}{3}} \right] \right) P_y(\tau, y) \\ - \frac{1}{T} \frac{s'_f(\tau)}{s_f(\tau)} P_y(\tau, y) + r x_{\max} P(\tau, y) = 0. \end{aligned} \quad (4.24)$$

Simplifying further, we obtain as follows:

$$\begin{aligned} \frac{x_{\max}}{T} P_\tau(\tau, y) - \frac{1}{x_{\max}} \frac{\sigma^2}{2} P_{yy}(\tau, y) - \left(r - \delta - \frac{\sigma^2}{2} \right) P_y(\tau, y) - \frac{1}{T} \frac{s'_f(\tau)}{s_f(\tau)} P_y(\tau, y) + r x_{\max} P(\tau, y) \\ - \frac{\sigma^2 v}{2} \left(\frac{1}{e^{x_{\max} s_f(\tau)}} \left[\frac{1}{x_{\max}} P_{yy}(\tau, y) - P_y(\tau, y) \right] \right)^{\frac{1}{3}} \left(\frac{1}{x_{\max}} P_{yy}(\tau, y) - P_y(\tau, y) \right). \end{aligned} \quad (4.25)$$

Finally, we obtain as follows:

$$\begin{aligned} & \frac{1}{T}P_\tau(\tau, y) - \frac{1}{x_{\max}^2} \frac{\sigma^2}{2} P_{yy}(\tau, y) - \frac{1}{x_{\max}} \left(r - \delta - \frac{\sigma^2}{2} \right) P_y(\tau, y) - \frac{1}{x_{\max} T} \frac{s'_f(\tau)}{s_f(\tau)} P_y(\tau, y) + rP(\tau, y) \\ & - \frac{\sigma^2 v}{2x_{\max}} \left(\frac{1}{e^{x_{\max} s_f(\tau)}} \right)^{\frac{1}{3}} \left[\frac{1}{x_{\max}} P_{yy}(\tau, y) - P_y(\tau, y) \right]^{\frac{4}{3}}. \end{aligned} \quad (4.26)$$

We would love to mention that the reformulated model in (4.26) is more suitable to implement using our proposed DNN method, which is equivalent to (4.24). Furthermore, observing (4.26), we could see that the last expression is actually a nonlinear differential term in a fractional form which is added to the dividend paying American options model in (2.15). If $v = 0$, we will recover the transformed American options model in (2.15) without transaction cost. Hence, our presented American options model with transaction cost as presented in (4.26) is strongly nonlinear. It may be a bit challenging to account for the second derivative of the value function in its auxiliary counterpart using Case 1 in (4.3). Here, we resort to Case 2 as presented in (4.4). Below we consider MTM and compare the plots of the early exercise boundary with and without transaction cost ($v = 0$). For the transaction cost, we consider the coefficient v as presented in the work of (Jandačka and Ševčovič, 2005) and (Egorova et al., 2017) with

$$v = 3 \left(\frac{C_{tr}^2 R}{2\pi} \right)^{\frac{1}{3}}. \quad (4.27)$$

In Figure 14 below, we present two plot profiles of the early exercise boundary. In the first plot, we vary $R = 0, 40, 80, 120, 160$ and fix $C_{tr} = 0.01$. In the second plot, we vary $C_{tr} = 0.1, 0.01, 0.001, 0$, and fix $R = 160$.

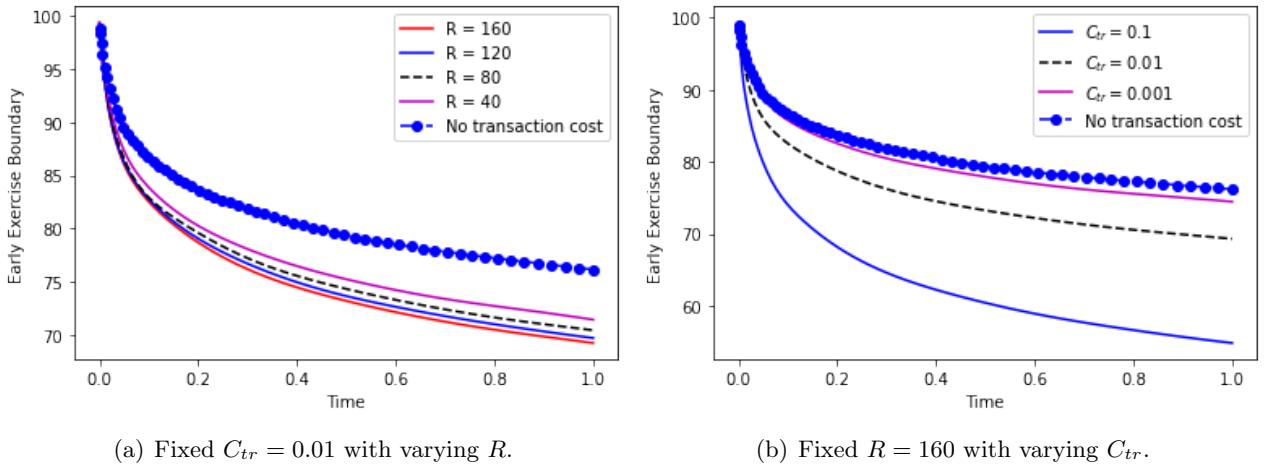


Figure 14: Plots of $s_f(t)$ with and without transaction cost using MTM and (60×60) uniformly stretched grid set ($p = 1.7$). Early exercise boundary is plotted backward in time.

It is important to mention that for $R = 0$ and $C_{tr} = 0$, we have it that $v = 0$. Here, we compute the result directly from the non-dividend paying American options pricing model. The reason is that we want to verify if the plot of the early exercise boundary obtained from the American options pricing model with transaction cost will start converging around the one obtained from the original model without transaction cost. From Figure 14, we observed that when $C_{tr} \rightarrow 0$, $v \rightarrow 0$ faster and the early exercise boundary from the model with transaction cost start converging around the one obtained from the model without transaction cost. Hence, we can validate that our proposed method efficiently approximates the strongly non-linear options pricing model as presented in (4.26). In our future work, we hope to comprehensively, explore and investigate the American options model with other forms of transaction cost, which include the Leland model (Leland, 1985), Wilmott model (Wilmott et al., 1994), and Barles and Soner (Barles and Soner, 1998) model using our proposed method.

5 Conclusion

We have proposed a deep learning method for solving the American options model as a free boundary problem. For the efficient computation of the early exercise boundary and its derivative simultaneously with the value function and the Greeks, we first implement a logarithmic Landau transformation. We then introduce a novel auxiliary function that includes the feed-forward DNN part and mimic the boundary behaviours. We trained this auxiliary function and its derivative to simultaneously account for the smooth pasting condition, far boundary behaviour, and other boundary conditions associated with the derivatives of the value function under the further condition (value matching condition) that a linear relationship between the early exercise boundary and feed-forward DNN output at the left boundary holds.

The performance of our DNN method was rigorously tested with several examples and compared with the existing methods. All indicators show that our proposed DNN method provides a more efficient and alternative way of solving free boundary style options and could be very useful in a high-dimensional context. Moreover, our approach can easily be extended for solving other free boundary PDEs like the Stefan problem (Liu et al., 2020), logistic population model (Piqueras et al., 2017), etc., where Landau transformation can be used for extracting the free boundary features. Furthermore, the potential of our proposed method for solving a wider range of options pricing models is tested on a couple of nonlinear free boundary options pricing models with special market features. Our results in this context are both promising and preliminary. More importantly, it provides a medium for further comprehensive studies, applicability, and extensions.

Even though our experiment is almost exhaustive, in our future work, we will further enhance the performance of our proposed method by implementing a more rigorous hyperparameter tuning approach that will give us an optimal DNN architecture such that we can obtain a more precise solution and efficient minimization with lesser computational time. Also, we will conduct a convergence analysis of our deep learning method for solving our model and carry out an extensive sensitivity analysis to extract the remaining Greeks. We also hope to extend our approach to American options with jumps, regime switching, stochastic volatility and interest rate, energy-related options with one or multi-free boundaries, free boundary Quanto and basket FX options, and other exotic options that involve more than one free boundary like straddle and strangle options.

Funding

The first author is funded in part by an NSERC Discovery Grant. The second author was funded in part by PIMS and the School of Mathematics, Cardiff University, during her research visit to the University of Calgary.

Code and Data Availability

The code and data could be shared at a reasonable request.

Conflict of interest

The authors declare that they have no conflict of interest

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv Preprint arXiv:1603.04467*.

- Alexiades, V. and Cannon, J. R. (1980). Free boundary problems in solidification of alloys. *SIAM Journal on Mathematical Analysis*, 11(2):254–264.
- Anderson, D. and Ulrych, U. (2022). Accelerated American option pricing with deep neural networks. *Swiss Finance Institute Research Paper*, (22-03).
- Andreucci, D. and Gianni, R. (1994). Classical solutions to a multidimensional free boundary problem arising in combustion theory. *Communications in Partial Differential Equations*, 19(5-6):803–826.
- Ballestra, L. V. (2018). Fast and accurate calculation of American option prices. *Decisions in Economics and Finance*, 41(2):399–426.
- Bänsch, E., Paul, J., and Schmidt, A. (2013). An ALE finite element method for a coupled Stefan problem and Navier–Stokes equations with free capillary surface. *International Journal for Numerical Methods in Fluids*, 71(10):1282–1296.
- Barles, G. and Soner, H. M. (1998). Option pricing with transaction costs and a nonlinear Black-Scholes equation. *Finance & Stochastics*, 2(4).
- Becker, S., Cheridito, P., and Jentzen, A. (2020). Pricing and hedging American-style options with deep learning. *Journal of Risk and Financial Management*, 13(7):158.
- Brennan, M. J. and Schwartz, E. S. (1977). The valuation of American put options. *The Journal of Finance*, 32(2):449–462.
- Bunch, D. S. and Johnson, H. (2000). The American put option and its critical stock price. *The Journal of Finance*, 55(5):2333–2356.
- Caffarelli, L. A. and Vázquez, J. L. (1995). A free-boundary problem for the heat equation arising in flame propagation. *Transactions of the American Mathematical Society*, 347(2):411–441.
- Carr, P. (1998). Randomization and the American put. *The Review of Financial Studies*, 11(3):597–626.
- Ceseri, M. and Stockie, J. M. (2014). A three-phase free boundary problem with melting ice and dissolving gas. *European Journal of Applied Mathematics*, 25(4):449–480.
- Chen, F., Sondak, D., Protopapas, P., Mattheakis, M., Liu, S., Agarwal, D., and Di Giovanni, M. (2020). Neurodiffq: A python package for solving differential equations with neural networks. *Journal of Open Source Software*, 5(46):1931.
- Chen, X., Cheng, H., and Chadam, J. (2013). Nonconvexity of the optimal exercise boundary for an American put option on a dividend-paying asset. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 23(1):169–185.
- Chen, Y. and Wan, J. W. (2021). Deep neural network framework based on backward stochastic differential equations for pricing and hedging American options in high dimensions. *Quantitative Finance*, 21(1):45–67.
- Cox, J. C., Ross, S. A., and Rubinstein, M. (1979). Option pricing: A simplified approach. *Journal of Financial Economics*, 7(3):229–263.
- Deng, L., Yu, D., et al. (2014). Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387.
- Detemple, J. (2001). American options: symmetry properties. *Option Pricing, Interest Rates and Risk Management*, pages 67–104.
- Dolezel, P., Skrabanek, P., and Gago, L. (2016). Weight initialization possibilities for feedforward neural network with linear saturated activation functions. *IFAC-PapersOnLine*, 49(25):49–54.

- Dutta, S. and Shekhar, S. (1988). Bond rating: a non-conservative application of neural networks. In *IEEE Int Conf on Neural Networks*, pages 443–450. Publ by IEEE.
- Egorova, V., Jódar, L., et al. (2014). Solving American option pricing models by the front fixing method: Numerical analysis and computing. In *Abstract and Applied Analysis*, volume 2014. Hindawi.
- Egorova, V. N., Tan, S.-H., Lai, C.-H., Company, R., and Jódar, L. (2017). Moving boundary transformation for american call options with transaction cost: finite difference methods and computing. *International Journal of Computer Mathematics*, 94(2):345–362.
- Eskiizmirli, S., Günel, K., and Polat, R. (2021). On the solution of the Black–Scholes equation using feed-forward neural networks. *Computational Economics*, 58(3):915–941.
- Fazio, R., Insana, A., and Jannelli, A. (2021). A front-fixing implicit finite difference method for the American put options model. *Mathematical and Computational Applications*, 26(2):30.
- Filippi, J.-B., Morandini, F., Balbi, J. H., and Hill, D. R. (2010). Discrete event front-tracking simulation of a physical fire-spread model. *Simulation*, 86(10):629–646.
- Frankel, M. L. and Roytburd, V. (1994). A free boundary problem modeling thermal instabilities: stability and bifurcation. *Journal of Dynamics and Differential Equations*, 6(3):447–486.
- Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.
- Golbabai, A. and Seifollahi, S. (2006). Numerical solution of the second kind integral equations using radial basis function networks. *Applied Mathematics and Computation*, 174(2):877–883.
- Gutiérrez, Ó. (2013). American option valuation using first-passage densities. *Quantitative Finance*, 13(11):1831–1843.
- He, J. and Xu, J. (2019). MgNet: a unified framework of multigrid and convolutional neural network. *Science China Mathematics*, 62(7):1331–1354.
- He, S., Reif, K., and Unbehauen, R. (2000). Multilayer neural networks for solving a class of partial differential equations. *Neural Networks*, 13(3):385–396.
- Hele-Shaw, H. S. (1898). Flow of water. *Nature*, 58(1509):520–520.
- Hirsa, A., Karatas, T., and Oskoui, A. (2019). Supervised deep neural networks (DNNs) for pricing/calibration of vanilla/exotic options under various different processes. *arXiv Preprint arXiv:1902.05810*.
- Hou, M., Fu, H., Hu, Z., Wang, J., Chen, Y., and Yang, Y. (2022). Numerical solving of generalized Black-Scholes differential equation using deep learning based on blocked residual connection. *Digital Signal Processing*, 126:103498.
- Hu, H. and Argyropoulos, S. A. (1996). Mathematical modelling of solidification and melting: a review. *Modelling and Simulation in Materials Science and Engineering*, 4(4):371.
- Hussian, E. A. and Suhhiem, M. H. (2015). Numerical solution of partial differential equations by using modified artificial neural network. *Network and Complex Systems*, 5(6):11–21.
- Hutchinson, J. M., Lo, A. W., and Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889.
- Jandačka, M. and Ševčovič, D. (2005). On the risk-adjusted pricing-methodology-based valuation of vanilla options and explanation of the volatility smile. *Journal of Applied Mathematics*, 2005(3):235–258.

- Jiangu, L., Siwei, L., Yingjian, Q., and Yaping, H. (2003). Numerical solution of elliptic partial differential equation using radial basis function neural networks. *Neural Networks*, 16(5-6):729–734.
- Khashman, A. (2010). Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. *Expert Systems with Applications*, 37(9):6233–6239.
- Khoo, Y., Lu, J., and Ying, L. (2021). Solving parametric PDE problems with artificial neural networks. *European Journal of Applied Mathematics*, 32(3):421–435.
- Kim, I. C. (2003). A free boundary problem arising in flame propagation. *Journal of Differential Equations*, 191(2):470–489.
- Kingma Diederik, P. and Adam, J. B. (2014). A method for stochastic optimization. *arXiv Preprint arXiv:1412.6980*.
- Kumar, V., Durst, F., and Ray, S. (2006). Modeling moving-boundary problems of solidification and melting adopting an arbitrary Lagrangian-Eulerian approach. *Numerical Heat Transfer, Part B: Fundamentals*, 49(4):299–331.
- Lagaris, I. E., Likas, A., and Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on Neural Networks*, 9(5):987–1000.
- Leland, H. E. (1985). Option pricing and replication with transactions costs. *The Journal of Finance*, 40(5):1283–1301.
- Létourneau, P. and Stentoft, L. (2019). Bootstrapping the early exercise boundary in the least-squares Monte Carlo method. *Journal of Risk and Financial Management*, 12(4):190.
- Lin, J. and Almeida, C. (2021). American option pricing with machine learning: An extension of the Longstaff-Schwartz method. *Brazilian Review of Finance*, 19(3):85–109.
- Liu, H. and Markowich, P. (2020). Selection dynamics for deep neural networks. *Journal of Differential Equations*, 269(12):11540–11574.
- Liu, S., Du, Y., and Liu, X. (2020). Numerical studies of a class of reaction–diffusion equations with Stefan conditions. *International Journal of Computer Mathematics*, 97(5):959–979.
- Liu, Z., Yang, Y., and Cai, Q.-D. (2019). Solving differential equation with constrained multilayer feedforward network. *arXiv Preprint arXiv:1904.06619*.
- Longstaff, F. A. and Schwartz, E. S. (2001). Valuing American options by simulation: A simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147.
- Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. (2021). Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228.
- MacKean Jr, H. (1965). A free boundary problem for the heat equation arising from a problem in mathematical economics. *Industrial Management Review*, 6:32–39.
- Malliaris, M. and Salchenberger, L. (1993). A neural network model for estimating option prices. *Applied Intelligence*, 3(3):193–206.
- Morandi Cecchi, M. and Russo, R. (2005). Numerical approximation of free boundary problem by variational inequalities. Application to semiconductor devices. In *Variational Analysis and Applications*, pages 697–722. Springer.
- Musiela, M. and Rutkowski, M. (2006). *Martingale methods in financial modelling*, volume 36. Springer Science & Business Media.
- Narkhede, M. V., Bartakke, P. P., and Sutaone, M. S. (2022). A review on weight initialization strategies for neural networks. *Artificial Intelligence Review*, 55(1):291–322.

- Nielsen, B. F., Skavhaug, O., and Tveito, A. (2002). Penalty and front-fixing methods for the numerical solution of American option problems. *Journal of Computational Finance*, 5(4):69–98.
- Nwankwo, C. and Dai, W. (2022a). On the efficiency of 5(4) RK-embedded pairs with high order compact scheme and Robin boundary condition for options valuation. *Japan Journal of Industrial and Applied Mathematics*, 39(2):753–775.
- Nwankwo, C. and Dai, W. (2022b). Sixth-order compact differencing with staggered boundary schemes and 3(2) Bogacki-Shampine pairs for pricing free-boundary options. *arXiv Preprint arXiv:2207.14379*.
- Pacelli, V., Azzollini, M., et al. (2011). An artificial neural network approach for credit risk management. *Journal of Intelligent Learning Systems and Applications*, 3(02):103.
- Piqueras, M.-A., Company, R., and Jódar, L. (2017). A front-fixing numerical method for a free boundary nonlinear diffusion logistic population model. *Journal of Computational and Applied Mathematics*, 309:473–481.
- Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv Preprint arXiv:1711.10561*.
- Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707.
- Reppen, A. M., Soner, H. M., and Tissot-Daguette, V. (2022). Neural optimal stopping boundary. *arXiv Preprint arXiv:2205.04595*.
- Sabate-Vidales, M., Šiška, D., and Szpruch, L. (2020). Solving path dependent PDEs with LSTM networks and path signatures. *arXiv Preprint arXiv:2011.10630*.
- Schmeiser, C. (2020). Free boundaries in semiconductor devices. In *Emerging Applications in Free Boundary Problems*, pages 268–272. Chapman and Hall/CRC.
- Sirignano, J. and Spiliopoulos, K. (2018). Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364.
- Umeorah, N. and Mba, J. C. (2022). Approximation of single-barrier options partial differential equations using feed-forward neural network. *Applied Stochastic Models in Business and Industry*, 38(6):1079–1098.
- van Moerbeke, P. (1974). Optimal stopping and free boundary problems. *The Rocky Mountain Journal of Mathematics*, 4(3):539–578.
- Villani, G. (2022). A neural network approach to value R&D compound American exchange option. *Computational Economics*, 60(1):305–324.
- Wang, S. and Perdikaris, P. (2021). Deep learning of free boundary and Stefan problems. *Journal of Computational Physics*, 428:109914.
- Wang, X., Li, J., and Li, J. (2022). A deep learning based numerical PDE method for option pricing. *Computational Economics*, pages 1–16.
- Wilmott, P., Hoggard, T., and Whalley, A. E. (1994). Hedging option portfolios in the presence of transaction costs. *Advances in Futures and Options Research*, 7.
- Wilmott, P., Howison, S., and Dewynne, J. (1995). *The mathematics of financial derivatives*. Cambridge university press.

- Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. (2023). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671.
- Wu, L. and Kwok, Y.-K. (1997). A front-fixing finite difference method for the valuation of American options. *Journal of Financial Engineering*, 6(4):83–97.
- Zhao, X. E., Hao, W., and Hu, B. (2021). Convergence analysis of neural networks for solving a free boundary problem. *Computers & Mathematics with Applications*, 93:144–155.
- Zhu, S.-P. and Chen, W.-T. (2011). A predictor–corrector scheme based on the ADI method for pricing american puts with stochastic volatility. *Computers & Mathematics with Applications*, 62(1):1–26.