Physics-Informed Neural Networks with Adaptive Sampling for Option Pricing

Hidde Agterberg[1]

Supervisor(s): Dr. Jing Sun[1]**, Dr. Alexander Heinlein**[1]**, Dr. Tiexing Wang**[2]

[1]EEMCS, Delft University of Technology, The Netherlands
[2]Shearwater Geoservices, United Kingdom

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 26, 2025

Name of the student: Hidde Agterberg
Final project course: CSE3000 Research Project
Thesis committee: Dr. Jing Sun, Dr. Alexander Heinlein, Dr. Tiexing Wang, Dr. Hayley Hung

Abstract

Today, machine learning has an accelerated impact in quantitative finance. Current models require large amounts of data, which can be expensive. A notable area of research, physics-informed neural networks (PINNs), has proven to be effective in approximating problems that are described by partial differential equations (PDEs). During training, the PDE is embedded in the loss function and evaluated at the residual points. This allows these types of neural networks to solve problems where data is scarce or noisy. Recent studies have shown that the method for sampling residual points has a great influence on training efficiency. Residual-based adaptive distribution (RAD) sampling is the adaptive sampling method used throughout this paper. This research applies PINNs with RAD sampling to solve the Black-Scholes PDE. Here, the Black-Scholes model is used to determine the price of options in a financial market. The fundamental goal of this paper is to study the difference in training performance between non-adaptive and RAD sampling. The types of options that are being considered in this study, are the European call options and the American put options. The results shown suggest that both types of options benefit from using RAD sampling compared to non-adaptive sampling. With a loss decrease of 39.33%, American put options improve more using RAD sampling than European call options. Although European call options still show a decrease in loss of 7.57%.

1 Introduction

The world of finance is increasingly dependent on machine learning methods to model markets and asset prices [9]. These models are often very complex because financial markets are driven by many factors. This means that a large amount of data is required to train these models to predict future value [11].

The latter is where physics-informed neural networks (PINNs) could add value in the industry. These types of neural networks represent a relatively recent advancement in the field of artificial intelligence. PINNs are used to solve problems described by partial differential equations (PDEs) [25]. PINNs work by incorporating the relevant PDE, including initial and boundary conditions, as part of the loss function. They need much less data than ordinary neural networks and work in cases where the data is noisy [16].

Since PINNs are a relatively new topic of research, there are still many aspects that can be refined to improve performance. Recent studies have reported that the method of sampling residual points plays a major role in the training efficiency of PINNs [20]. These residual points are the points sampled to compute the loss of the PDE associated with the PINN.

A new way of sampling is based on the research by Wu et al. [30], where they introduce residual-based adaptive distribution (RAD) sampling. This works by constructing a probability density function (PDF) based on the loss function. It computes the probability of a point being sampled for the next training iterations. If the loss is high at a certain residual point, then the probability of sampling this point increases.

This research touches on the problem of pricing options. The most popular model for estimating the value of an option is the Black-Scholes model, which was originally introduced in 1973 [3]. Their model makes several assumptions about the underlying asset, one of which is that the value of the underlying asset follows a geometric Brownian motion.

The main focus of this paper is to investigate how RAD sampling affects the performance of PINNs solving option pricing compared to random non-adaptive sampling. Furthermore, numerical methods [1][6][4] for solving Black-Scholes are discussed. Lastly, the type of option that benefits most from PINNs and RAD sampling is identified.

The application of PINNs to solve the Black-Scholes equation is an important area of research, as current (numerical) methods for solving Black-Scholes are computationally expensive and relatively less scalable [9]. However, the implementation of PINNs for option pricing remains largely unexplored.

The structure of the paper is as follows. Section 2 describes the methodology on which the rest of the paper and the experiment are based. It shows how PINNs, RAD sampling, and Black-Scholes work. Subsequently, Section 3 shows the experimental setup and the results are followed in Section 4. Section 5 discusses the ethical aspects of the results and shows the reproducibility. Moreover, Section 6 discusses the results from the previous sections. Finally, Section 7 concludes the paper and summarizes the work.

Related literature

The topic of PINNs is a new field in the academic literature. In addition, the topics of sampling methods and option pricing for PINNs have been very little studied as of today. The following section covers the research that is related to adaptive sampling and option pricing with PINNs.

The RAD sampling method, which is used for this research, was originally introduced in the study by Wu et al. [30]. They compared two different groups of sampling approaches: non-adaptive and adaptive sampling. Non-adaptive sampling methods include: uniform sampling, random sampling, Latin hypercube sampling [21], Halton sequence [12], Hammersley sequence [13], and Sobol sequence [26]. These non-adaptive sampling methods and uniform points with resampling are compared against three adaptive sampling techniques. The first is Residual-based Adaptive Refinement (RAR), which is a greedy algorithm first introduced in Lu et al. [20]. The second and third methods are RAD sampling and residual-based adaptive refinement with distribution (RAR-D) sampling, which were both proposed by Wu et al. [30]. Their results show a significant improvement in performance of the PINNs when using the RAD or RAR-D sampling methods.

Liu et al. introduced another adaptive sampling algorithm called EI-RAR [19]. This method builds on the previously

mentioned RAR and RAR-D algorithms. The EI-RAR algorithm adds a new expected improvement (EI) function, which, compared to RAR-D, places a greater emphasis on boundary points. In addition, Liu et al. introduce EI-Grad, a second algorithm that considers the gradients of the residuals when making sampling decisions.

The work by Wang on Relative Residual Resampling [28] takes a different approach to improve residual sampling. Their research is based on the conception that there is often an unbalance in the number of points sampled for boundaries and initial conditions versus the samples within the PDE's domain. They introduce Relative Residual Resampling (R3), which dynamically changes the number of sampling points for each type of point in training.

As for related work, which focuses on PINNs for option pricing, Tanios [27] was one of the first to adopt PINNs for this application. They focus on multi-asset European options, solving forward and inverse cases of the Black-Scholes model.

In the same period, Bai et al. [2] proposed their improved physics-informed neural network (IPINN) method for option pricing in finance. They were successfully able to improve the performance of PINNs using a local adaptive activation function [15].

In contrast to the previous papers, Gatta et al. created a method for American option pricing [9]. They introduced a trick for the free boundary problem [29] that occurs with American options. The free boundary is trained as a separate entity with its own initial condition, in addition it is related to the PINN's Dirichlet and Neumann boundary conditions.

2 Methodology

All the relevant background information and methods on how the model is constructed, is described in this section. Starting with Section 2.1, illustrating the approach that PINNs take. Section 2.2 shows the workings of the RAD sampling method. Finally, Section 2.3 describes the problem of option pricing and introduces the Black-Scholes model as a solution.

2.1 Physics-Informed Neural Networks

The first introduction of PINNs was in 2019 by Raissi et al. [25]. They developed a framework that can be used to solve supervised problems that are described by partial differential equations. This framework can be used to simulate in a situation where all parameters of the PDE are known. On the contrary, when not all parameters are available but simulations are provided, it allows for the retrieval of the parameters in an inverse manner.

This paper focuses on the forward problem using PINNs. The fundamental concept is that one can use a neural network adding the PDE to the loss function, then with automatic differentiation and backpropagation capabilities (included in most machine learning libraries) one can calculate the loss and gradients [16]. The loss function for the PINNs in this paper takes on the following structure:

$$\mathcal{L} = w_{pde}\mathcal{L}_{pde} + w_{init}\mathcal{L}_{init} + w_{b0}\mathcal{L}_{b0} + w_{b1}\mathcal{L}_{b1} \tag{1}$$

Here in Eq. 1, $\mathcal{L}$ denotes the total loss consisting of the PDE loss $\mathcal{L}_{pde}$, the loss of the initial condition $\mathcal{L}_{init}$, and boundary boundary losses $\mathcal{L}_{b0}$ and $\mathcal{L}_{b1}$. The loss components can be balanced using the weights w_{pde}, w_{init}, w_{b0}, and w_{b1}. The loss components are measured with the mean squared error (MSE) [7], for N points it is defined by:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2}$$

In Eq. 2, y_i denotes the observed value and $\hat{y}_i$ is the predicted value. The PDE loss component is computed by the following:

$$\mathcal{L}_{pde} = \frac{1}{N_{pde}}\sum_{i=1}^{N_{pde}}(\varepsilon(t^i_{pde}, x^i_{pde}))^2 \tag{3}$$

In Eq. 3, $\{t^i_{pde}, x^i_{pde}\}_{i=1}^{N_{pde}}$ is the set of residual points for $\varepsilon(t, x)$. Where $\varepsilon(t, x)$ denotes the residual value at time t and point x according to the PDE. The initial and boundary loss components are all computed in a similar manner, using the following equation:

$$\mathcal{L}_{bnd} = \frac{1}{N_{bnd}}\sum_{i=1}^{N_{bnd}}(u(t^i_{bnd}, x^i_{bnd}) - \hat{u}^i_{bnd})^2 \tag{4}$$

Here, $\{t^i_{bnd}, x^i_{bnd}, \hat{u}^i_{bnd}\}_{i=1}^{N_{bnd}}$ denotes the data for the initial or boundary points and $\hat{u}^i_{bnd}$ marks the predicted value for an initial or boundary point at $u(t, x)$. Where $u(t, x)$ computes the value at a certain initial or boundary point at time t and point x.

2.2 Residual-based Adaptive Distribution Sampling

The adaptive sampling technique called RAD sampling was introduced by Wu et al. [30]. Their inspiration came from Nabian et al. [23] who developed a sampling strategy which resamples the residual points using a PDF. These methods start with a large non-adaptive sample space, for each point the residual is calculated. Finally, sample the desired number of points according to the PDF. Their improved PDF of the RAD methods is as follows:

$$p(x) \propto \frac{\varepsilon^k(x)}{\mathbb{E}[\varepsilon^k(x)]} + c \tag{5}$$

Eq. 5 calculates the probability of sampling $p(x)$, which includes two non-negative hyperparameters k and c. Wu et al. [30] advise $k = 1$ and $c = 1$ as a default choice. In Eq. 5, $\varepsilon(x)$ is the residual value at x and $\mathbb{E}[\varepsilon^k(x)]$ is an approximation of the mean of $\varepsilon^k(x)$, which can be estimated using numerical integration.

The algorithm of the RAD sampling method developed by Wu et al. is shown in Algorithm 1:

Algorithm 1 RAD

1: Randomly sample N initial residual points R
2: Run the training on R for n iterations
3: **while** the total number of iterations is not reached **do**
4: $R \leftarrow N$ new points picked by Eq. 5
5: Run the training on R for n iterations
6: **end while**

The initial N residual points (line 1 of Algorithm 1) are randomly sampled from a uniform distribution over the entire domain. In line 4 of Algorithm 1 the algorithm needs to pick new points according to Eq. 5. With a low-dimensional x, like in the case of this research, this can be done with the following steps [30]:

1. Sample a large set of points S_0 using random sampling
2. Calculate $p(x)$ for all points in S_0
3. Create the probability mass function $P(x) = \frac{p(x)}{A}$ where $A = \sum_{x \in S_0} p(x)$
4. Sample points from S_0 using $P(x)$

2.3 Black-Scholes Option Pricing

The Black-Scholes model [3] has been widely used for pricing financial derivatives and in particular stock options. There are many types of stock options, however, the most common characteristic is that they give the holder the right but not the obligation to buy or sell a stock at a later time for a predetermined price. An option that allows one to buy a stock is called a call option [22], while an option that allows someone to sell a stock is called a put option [5]. All options have an expiration date, underlying stock price, and a strike price (the price for which the owner can buy or sell). The most widely known types of options are European and American options. The only distinction between the two is that with European options, the owner can only exercise its right to buy or sell on the expiration date. In contrast, with American options, the owner has the right to exercise at any moment before the expiration date.

The Black-Scholes model makes several assumptions about the options [3], the following are the most relevant:

1. The short-term interest rate is constant
2. The underlying price follows a random walk
3. The stock does not pay any dividends

An important property for this research of the Black-Scholes model is that it can be described by a PDE [3]:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} = rV - rS\frac{\partial V}{\partial S} \tag{6}$$

In the PDE from Eq. 6 the parameters are the option price V, underlying price S, time t, volatility σ, and risk-free interest rate r.

2.3.1 European call option

An additional assumption that the Black-Scholes originally makes is that the option is European. The European call and put options are very similar, the European put option is left out because it is not used throughout this research. For European options there exists an analytical solution. The initial and boundary conditions for European call options are defined below.

$$\begin{aligned} &C(0,t) = 0 \\ &C(S,t) = S - K \quad \text{when} \quad S \to \infty \\ &C(S,T) = \max(S-K, 0) \end{aligned} \tag{7}$$

In Eq. 7, $C(S,t)$ is the value of a call option, with underlying stock price S, time t, time at maturity T, and strike price K. Note that the last condition is the initial condition.

The analytical solution for European call options from the Black-Scholes method:

$$\begin{aligned} C(S,t) &= S\Phi(d_1) - Ke^{r(T-t)}\Phi(d_2) \\ d_1 &= \frac{\ln(S/K) + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}} \\ d_2 &= \frac{\ln(S/K) + (r - \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}} \end{aligned} \tag{8}$$

Here in Eq. 8, Φ denotes the cumulative distribution function of the standard normal distribution. An example of an analytical solution is visualized in the following figure:

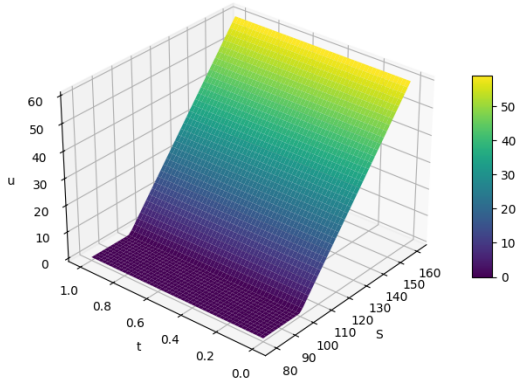

Figure 1: The analytical solution to an example of the European call option problem. Where the option is described by $K = 100$, $r = 0.05$ and $\sigma = 0.1$. The solution is computed according to the method from Eq. 8. It shows the value u at time t and underlying stock price S.

Figure 1 show the price of the option u over time t and underlying price S.

2.3.2 American put option

American call options are near equivalent to European call options [17], therefore the American call version is not considered in this paper.

Unlike the European call option, pricing American put options with the Black-Scholes model cannot be done analytically. This is due to its early exercise ability that allows the option owner to exercise at any time before the expiration. There are several numerical methods to solve this pricing problem, a few examples are finite elements [1], finite differences [6], and Monte-Carlo simulation [4]. These methods

aim to solve the same problem using different approaches. Finite elements and finite differences, unlike Monte-Carlo, discretize the PDE. On the other hand, Monte-Carlo simulations use random sampling to make an approximation. For this research Monte-Carlo simulations are used, because they are both quick and intuitive to implement. It works by the assumption that stock prices fluctuate, following a geometric Brownian motion with a drift. The value of a stock at some moment can be modeled as the following:

$$V_{t+1} = V_t e^{r - \frac{1}{2}\sigma^2 + \sigma X} \tag{9}$$

In Eq. 9, V_t is the value of a stock at time t, with risk-free interest rate r, volatility σ, and a standard normal random variable X. An example of simulating the price of an American put option using a Monte-Carlo method is demonstrated below.

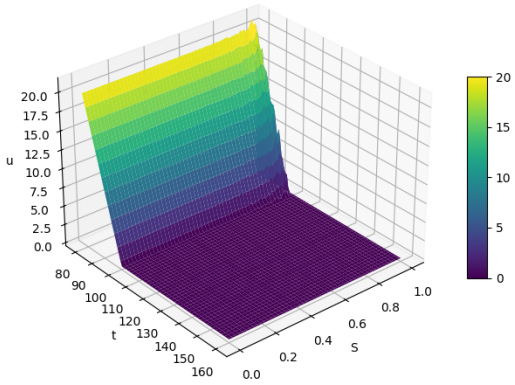

Figure 2: A numerical solution for an example of am American put option using Monte-Carlo simulation. The option is described by $K = 100$, $r = 0.05$ and $\sigma = 0.1$. The simulation was computed using Eq. 9. It shows the value u at time t and underlying stock price S.

Here in Figure 2, u denotes the price of the option over time t and underlying value S.

The initial and boundary conditions for American put options are the following:

$$\begin{aligned} &P(0,t) = K \\ &P(S,t) = 0 \quad \text{when} \quad S \to \infty \\ &P(S,T) = \max(K - S, 0) \end{aligned} \tag{10}$$

Here, $P(S,t)$ is the value of the American put option. Again, note that the last boundary is the initial boundary.

The main difference that the American put problem introduces is the free boundary problem [9]. It is the problem of deciding on a price low enough to exercise the put option before the expiration. The answer to this problem is approximated by $B(t)$ using a neural network. This neural network has its own loss function and the initial condition is:

$$B(T) = K \tag{11}$$

The free boundary also has Dirichlet and Neumann boundary conditions, these are respectively:

$$\begin{aligned} &P(B(t),t) = K - B(t) \\ &\frac{\partial P}{\partial S}(B(t),t) = -1 \end{aligned} \tag{12}$$

3 Experimental setup

This section covers the experimental setup, which first explains how the general results were produced in Section 3.1. Secondly, Section 3.2 explains the implementation of the RAD sampling method. Finally, Sections 3.3 and 3.4 illustrate the setups for European call and American put options.

3.1 General

Throughout this experiment, the work was supported by the PyTorch machine learning library [24], a widely used library that allows for automatic differentiation. All PINNs in the experiments used a feed-forward neural network with four layers in total. The input layer consists of two nodes, which represent the underlying price S and time t. These are then fed through layers two and three, which both contain 16 hidden nodes. The output layer has a single node for the approximated price. The current size of the network should be large enough to handle the complexity while training efficiently. For all layers, the biases are initialized to zero, which is a common starting point [10]. The weights are initialized according to the uniform Xavier initialization [10], which draws each weight from a random uniform distribution and ensures stable training. The activation function that is used for the input and hidden layers is the tangent hyperbolic (tanh) function [8]. The tanh function is used for approximating continuous and non-linear functions, and thus suitable for the Black-Scholes equation (Eq. 6). According to Xavier et al. [10], tanh networks perform well in combination with Xavier initialization.

Each loss component is measured using the MSE function, as shown in Eq. 2. The model was trained using the Adam optimizer [18], which can be considered a standard, 0.001 is the value used as the learning rate.

The following option parameters were used in the experiments:

$$\begin{aligned} &K = 100 \quad S \in [80, 160] \\ &r = 0.05 \quad t \in [0, 1] \\ &\sigma = 0.1 \end{aligned} \tag{13}$$

Initially, the PINNs start with a mesh of shape 2×2000 which represents the residual points. The mesh is filled with pseudo-random values that span the domains of S and t. The initial and boundary conditions all have a shape 2×200, and span either S or t, but not both. The initial condition is at maturity T (also $t = 1$), the first boundary condition is $S = 0$, and the second boundary is when $S \to \infty$. The weights associated with the PDE loss w_{pde} and the boundary conditions w_{bnd} are equal and set to $w_{pde}, w_{bnd} = 1$.

3.2 Residual-based Adaptive Distribution sampling

There are three hyperparameters for the RAD sampling method, the default values are:

$$k = 1, \quad c = 1, \quad \text{rf} = 50 \tag{14}$$

Here, rf (resampling frequency) is the number of training iterations between RAD resampling steps. Wu et al. [30] resample either every 1000 or 2000 iterations. In this research

however, we consider multiple possible resampling frequencies. These resampling frequencies range from 25 to 1000 iterations, lower than in Wu et al. [30] because they use more training iterations. There is an additional mesh of residual points, which uniformly spans the domains of S and t with the shape 2×10000. RAD resampling starts by running the network on this large mesh. Subsequently, calculate $p(x)$ using Eq. 5 for each point. Using the probabilities from the PDF, construct a probability mass function $P(x) = \frac{p(x)}{A}$ where $A = \sum p(x)$. Finally, the original mesh (with shape 2×2000) is replaced by picking 2000 samples from the large mesh according to the probability mass function.

Each experiment is run for 16,000 iterations. The performance of the network is evaluated for each iteration using uniform residual, initial, and boundary points. A threshold is used to discard training runs that do not converge, because these runs give a skewed view of the results. The threshold is equal to half of the starting loss. A run with a final loss higher than the threshold will be discarded and rerun. This

3.3 European call problem

In each iteration of training, the loss is computed according to Eq. 1 with the initial and boundary conditions for the European call options from Eq. 7.

3.4 American put problem

The American put problem requires an additional neural network for the free boundary problem. The network structure itself is similar to that of the PINN. Like the network mentioned above, it uses uniform Xavier initialization for the weights, biases initialized to zero, the tanh activation function and the Adam optimizer. The difference is that the input has one input (t) and the network contains only a single hidden layer of eight nodes. Because this is a less complex problem, the network can be smaller compared to the network for the PINN.

While training the American put problem, there are two loss terms, one for determining the price u and a second one for the free boundary problem $B(t)$. Firstly, the loss is computed similar to Eq. 1 where the initial and boundary conditions are described in Eq. 10. For the free boundary problem, the loss $\mathcal{L}_{bnd}$ is calculated with the initial condition from Eq. 11 and boundary conditions from Eq. 12. Upon ending each training iteration, both networks adopt the Adam optimizer.

4 Results

The following section shows the results of the experiment. The general goal is to show the performance of the RAD sampling method, compared to non-adaptive random sampling. To show this, the loss is initially analyzed in several combinations of values for k and c. Following with the best-performing combination of k and c, the loss is reviewed at a few resampling frequencies. With the best values for k, c and the resampling frequency, the performance is tested against a non-adaptive random sampling strategy.

This method for testing performance is applied to both European call and American put options as shown in Sections 4.1 and 4.2 respectively.

4.1 European call options

European call options are the first type of pricing problem for which the RAD sampling method is compared to the non-adaptive random sampling. RAD sampling requires two important hyperparameters k and c each resampling step. Wu et al. [30] advise to use the combination $k = 1$ and $c = 1$ as default. Additionally, they demonstrate values between 1 and 2 for k and between 0 and 1 for c performing well in different cases. For this reason, all integer combinations of these values are assessed on their performance. During training, the residual points are resampled every 50 training iterations. The MSE loss is computed by running each combination of k and c 10 separate times for 16,000 training iterations. Finally, the average of 10 runs is computed and plotted in Figure 3.

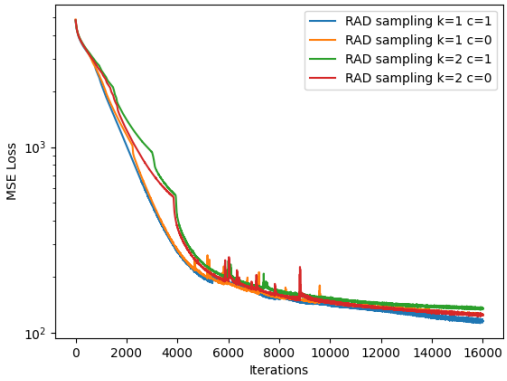

Figure 3: The comparison of MSE loss with different values for k and c using the RAD sampling method. Each combination is an average of 10 runs. With the MSE loss in a log scale and the number of training iterations on the horizontal axis.

k	c	loss	std dev
1	1	121.29	19.68
1	0	124.73	15.59
2	1	134.41	17.96
2	0	123.64	21.84

Table 1: The average loss and standard deviation (std dev) of the loss for the combinations of values for k and c. Where each combination is trained 10 individual times.

Table 1 reveals that the combination of $k = 1$ and $c = 1$ is the best performing out of the four. However, the difference in the loss from other combinations of k and c is small.

The following hyperparameter that needs to be tuned is the resampling frequency. The chosen range of frequencies lies between 25 and 1000, Wu et al. [30] resampled every 1000 iteration, however they trained the PINNs for 100.000 iterations. Figure 4 shows the MSE loss for each resampling frequency.

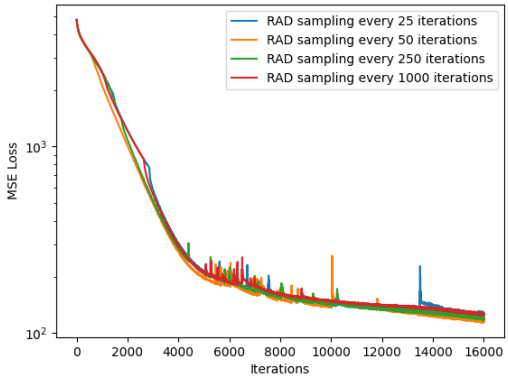

Figure 4: The comparison of MSE loss with different values for the resampling frequency using the RAD sampling method. Each loss curve resembles the average of 10 runs. With the MSE loss in a log scale and the number of training iterations on the horizontal axis.

rf	loss	std dev
25	128.08	17.69
50	116.55	15.33
250	118.22	16.63
1000	126.03	20.04

Table 2: The average loss and standard deviation (std dev) of the loss for different resample frequencies (rf). Where each instance is trained for 10 runs.

Here, Table 2 shows that, with $k = 1$ and $c = 1$, the best performing resampling frequency is 50. Additionally, the standard deviation for $\mathrm{rf} = 50$ was the lowest out of the different frequencies.

Lastly, to assess the performance of the RAD sampling method versus the non-adaptive random sampling, the RAD model with $k = 1$, $c = 1$ and $\mathrm{rf} = 50$ is run along the random sampling. Both the sampling strategies are trained 10 times and the average MSE loss is visualized in Figure 5.

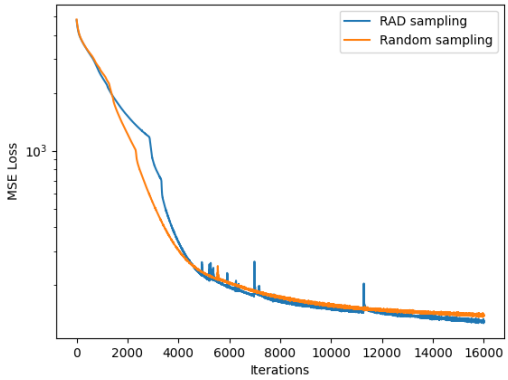

Figure 5: The comparison of MSE loss for RAD and non-adaptive random sampling. Each loss curve resembles the average of 10 runs. With the MSE loss in a log scale and the number of training iterations on the horizontal axis.

	loss	std dev
RAD sampling	128.21	22.25
Random sampling	138.72	14.57

Table 3: The average loss and standard deviation (std dev) of the loss for RAD and non-adaptive random sampling. Where each instance is trained for 10 runs.

Figure 5 and Table 3 show an improvement in performance with the RAD strategy. The average loss for RAD sampling is 7.57% lower than the non-adaptive random sampling. However, the standard deviation is higher with RAD sampling.

4.2 American put options

The second type of option pricing problem is the American put option. The RAD sampling method is evaluated against the non-adaptive random sampling, following a similar approach as described in Section 4.1. Starting with the hyperparameters k and c. Using the integer combinations of k ranging from 1 to 2, and c ranging from 0 to 1. While training the residual points, the points are resampled every 50 iterations. The loss is computed by training each combination of k and c 10 times, with each run containing 16,000 iterations. Lastly, for each combination, the average of 10 runs is computed and visualized in Figure 6.

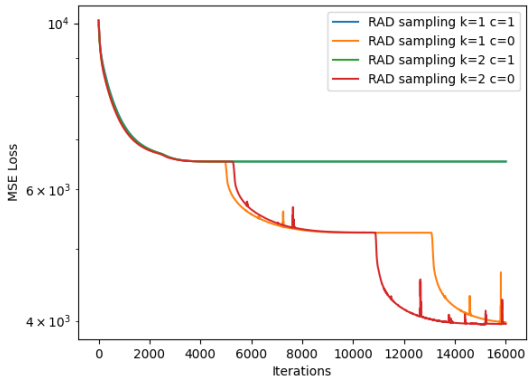

Figure 6: The comparison of MSE loss with different values for k and c using the RAD sampling method. Every combination is an average of 10 runs. With the MSE loss in a log scale and the number of training iterations on the horizontal axis.

k	c	loss	std dev
1	1	4066.82	3025.97
1	0	3980.89	3126.39
2	1	6533.48	0.02
2	0	3962.87	3148.35

Table 4: The average loss and standard deviation (std dev) of the loss for the combinations of values for k and c. Where each combination is trained 10 individual times.

Table 4 shows that the combination of $k = 2$ and $c = 0$ is the best performing values for k and c. With these chosen values, the subsequent hyperparameter that needs to be tuned

is the resampling frequency. The selected values for this frequency are 25, 50, 250 and 1000. Figure 7 shows the loss for each resampling frequency.

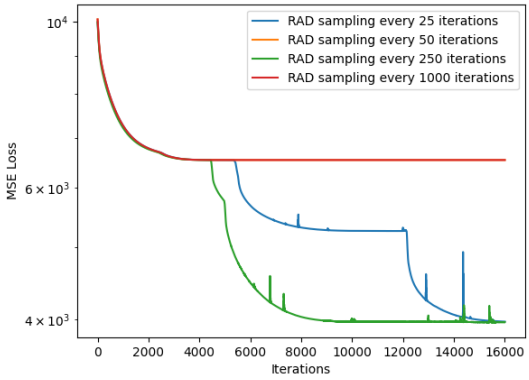

Figure 7: The comparison of MSE loss with different values for the resampling frequency using the RAD sampling method. Each loss curve resembles the average of 10 runs. With the MSE loss in a log scale and the number of training iterations on the horizontal axis.

rf	loss	std dev
25	3972.76	3136.26
50	6533.52	0.05
250	3965.82	3144.74
1000	6533.51	0.02

Table 5: The average loss and standard deviation (std dev) of the loss for different resample frequencies (rf). Where each instance is trained for 10 runs.

Table 5 shows that, using $k = 2$ and $c = 0$, the best performing resampling frequency is 250. To assess the performance of the RAD sampling method versus the non-adaptive random sampling, the RAD model with $k = 2$, $c = 0$, $\text{rf} = 250$ is run along the random sampling. Both the sampling strategies are trained 10 times and the average loss is visualized in Figure 8.

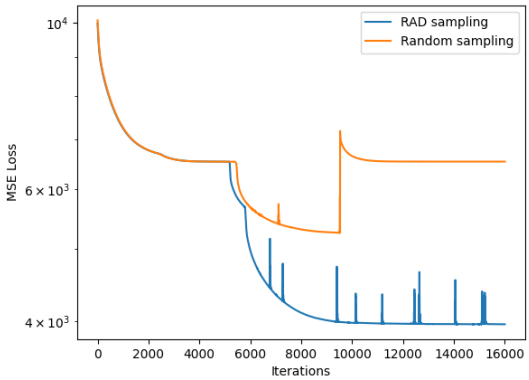

Figure 8: The comparison of MSE loss for RAD and non-adaptive random sampling. Each loss curve resembles the average of 10 runs. With the MSE loss in a log scale and the number of training iterations on the horizontal axis.

	loss	std dev
RAD sampling	3964.15	3146.88
Random sampling	6533.51	0.04

Table 6: The average loss and standard deviation (std dev) of the loss for RAD and non-adaptive random sampling. Where each instance is trained for 10 runs.

Figure 8 and Table 6 show a substantial improvement in performance with the RAD strategy. The average loss for RAD sampling is 39.33% lower than the non-adaptive random sampling. However, the standard deviation is again higher with RAD sampling.

5 Responsible Research

5.1 Ethics

One of the key ethical aspects of this research is the transparency of the models. To ensure fair option markets, knowledge on pricing options should be as transparent as possible. Specifically, it should be transparent to option writers (the party that creates an option) and the option buyer. In the case where a selective group is able to more accurately determine the value of an option, this information asymmetry can lead to imbalanced option markets. In these markets, there is more potential for parties with more knowledge. To address this, research should be available for all market participants, ensuring that there is balance.

Although research on option pricing models is more publicly available, the financial data required to use these models is expensive. This causes small parties to be held back from competing in the option markets. PINNs, needing considerably less data, could create the possibility for smaller players to compete.

Since the current model does not adopt historical data, there is no observational bias. These observational biases may lead to generalizing performance [16]. Conversely, the model does introduce an inductive bias from the Black-Scholes model and a learning bias from PINN achitecture.

5.2 Scientific Integrity

Throughout this research, Writefull[1] and ChatGPT[2] are used solely for the purposes of checking spelling, grammar and showing synonyms.

5.3 Reproducibility

The methods applied throughout this research, including background on PINNs, residual sampling and option pricing, are thoroughly described in Section 2. Subsequently, the details on the structure of the experiment are explained in Section 3. This also includes the parameters, functions and the order of execution. In addition to the methodology (Section 2) and the experimental setup (Section 3), the code used for the research is available and can be found in Section 7.

[1] https://www.writefull.com/

[2] https://openai.com/index/chatgpt/

6 Discussion

The first part of this section discusses the results from Section 4. Secondly, it considers possible limitations of the research. Finally, future improvements and additions are mentioned and discussed.

6.1 Results

The European option results shown in Section 4.1 are the results from the less complex problem. The experiments show that the combination of $k = 1$, $c = 1$ and $\mathrm{rf} = 50$ is the best performing out of the combinations that were considered. Although there were a few alternatives, including $k = 2$ and $c = 0$ or $\mathrm{rf} = 250$, that performed similar. Conversely, the performance of RAD sampling compared to random sampling shows a greater improvement.

On the other hand, the American put options show a more diverse performance for the different values of k, c and rf. The best recorded performance is with $k = 2$, $c = 0$ and $\mathrm{rf} = 250$. Alternative hyperparameters perform substantially worse, though the better performing parameters show a greater standard deviation. The RAD strategy outperforms the non-adaptive random sampling with a drop in MSE loss of 39.33%.

6.2 Limitations

One of the limitations of the European call option model is the small difference in the loss for the experiments. There is a small difference in the average loss, while the standard deviation is large. This suggesting that retraining could show different results. Combined with small differences in the average loss, it is difficult to draw strong conclusions. A second limitation is that some of the loss curves seem not fully converged. Longer training could improve the performance and increase the difference in loss for different hyperparameters. However, it could also result in the differences becoming smaller.

Inspecting the American put loss curves, the difference in outcome is profound. The cases where the loss is lower compared to competing strategies (e.g. Figure 6), have a standard deviation which is much greater. This suggests that these cases converge only occasionally. Finding a way to make the model more consistent would be a big improvement. The last limitation is the drop and sudden incline in loss which is visualized in Figure 8. This is uncommon behavior in neural networks and makes the result less reliable.

6.3 Future work

Future work could include multi-asset options, similar to the work by Gatta et al. [9]. Multi-dimensional underlying options are a beneficial area of study because they introduce diversification and reduce risk [14].

A second future addition could be an inverse version of the model, which could be used for calculating the Greeks for options [27].

7 Conclusion

Concluding this research, its main objective has been to investigate the effect of RAD sampling on the performance of PINNs for option pricing. Specifically solving Black-Scholes for both European call and American put options. For both type of option the same path is taken to asses the performance of RAD against non-adaptive random sampling. Firstly, various combinations of the RAD hyperparameters k and c were tested for their performance. Secondly, using the preferred values for k and c, the loss of different resampling frequencies rf are assessed. Finally, with these values for k, c and rf, the performance of the RAD sampling method has been matched against non-adaptive random sampling.

For the European call option $k = 1$, $c = 1$ and $\mathrm{rf} = 50$ are the best performing hyperparameters. Here, the RAD strategy shows a drop of 7.57% in loss compared to the non-adaptive methods. On the other hand, the American put options have achieved the best results with $k = 2$, $c = 0$ and $\mathrm{rf} = 250$. The American put option showed a greater improvement with a decrease of 39.33% in loss using RAD sampling compared to non-adaptive sampling. However, the outcomes and performance of the PINNs for American put options are less predictable.

Ultimately, there is a considerable improvement in performance using RAD sampling over non-adaptive random sampling. This increase in performance is already shown with European call options, the improvement is even more prominent in American put options. Expanding the problem to multi-asset options would be the next step to test RAD sampling.

Code

The code from this paper can be found on: https://github.com/hidde8erberg/CSE3000-PINNs

References

[1] A. Andalaft-Chacur, M. Montaz Ali, and J. González Salazar. Real options pricing by the finite element method. *Computers & Mathematics with Applications*, 61(9):2863–2873, 2011.

[2] Yuexing Bai, Temuer Chaolu, and Sudao Bilige. The application of improved physics-informed neural network (IPINN) method in finance. *Nonlinear Dynamics*, 107(4):3655–3667, March 2022.

[3] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[4] Phelim P Boyle. Options: A monte carlo approach. *Journal of financial economics*, 4(3):323–338, 1977.

[5] Michael J Brennan and Eduardo S Schwartz. The valuation of american put options. *The Journal of Finance*, 32(2):449–462, 1977.

[6] Michael J Brennan and Eduardo S Schwartz. Finite difference methods and jump processes arising in the pricing of contingent claims: A synthesis. *Journal of Financial and Quantitative Analysis*, 13(3):461–474, 1978.

[7] Peter Christoffersen and Kris Jacobs. The importance of the loss function in option valuation. *Journal of Financial Economics*, 72(2):291–318, 2004.

[8] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2022.

[9] Federico Gatta, Vincenzo Schiano Di Cola, Fabio Giampaolo, Francesco Piccialli, and Salvatore Cuomo. Meshless methods for american option pricing through physics-informed neural networks. *Engineering Analysis with Boundary Elements*, 151:68 – 82, 2023. Cited by: 6; All Open Access, Hybrid Gold Open Access.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[11] J. Teichmann H. Buehler, L. Gonon and B. Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

[12] John H Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, 1960.

[13] John M Hammersley. Monte carlo methods for solving multivariable problems. *Annals of the New York Academy of Sciences*, 86(3):844–874, 1960.

[14] Cynthia Ikamari, Philip Ngare, and Patrick Weke. Multi-asset option pricing using an information-based model. *Scientific African*, 10:e00564, 2020.

[15] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A*, 476(2239):20200334, 2020.

[16] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, May 2021.

[17] Deniz Kaya. Pricing a multi-asset american option in a parallel environment by a finite element method approach, 2011.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[19] Yanbing Liu, Liping Chen, Jianwan Ding, and Yu Chen. An adaptive sampling method based on expected improvement function and residual gradient in pinns. *IEEE Access*, 12:92130–92141, 2024.

[20] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[21] Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[22] Robert C Merton. Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3(1-2):125–144, 1976.

[23] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36(8):962–977, 2021.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[25] M. Raissi, P. Perdikaris, and G.E. Karniadakis. 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, 2019.

[26] Il'ya Meerovich Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

[27] Ramy Tanios. Physics informed neural networks in computational finance: High dimensional forward & inverse option pricing. Master's thesis, ETH Zurich, 2021.

[28] Haoran Wang. An adaptive residual-based sampling methods in training pinns. Master's thesis, Delft University of Technology, 2024.

[29] Sifan Wang and Paris Perdikaris. Deep learning of free boundary and stefan problems. *Journal of Computational Physics*, 428:109914, 2021.

[30] Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. 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, January 2023.