Multi-asset option pricing using deep learning

By:
Ashutosh Saxena
and
Shantanu Wagh

Under the Supervision of:
Dr. Lokpati Tripathi

Contents

Introduction

In today's financial landscape, multi-asset option pricing has become an essential tool for investors and financial institutions. Multi-asset options, also known as basket options, involve valuing options contracts on more than one underlying asset. In recent years, there has been a growing interest in using physics-informed neural networks (PINNs) for multi-asset option pricing. PINNs are a type of machine learning algorithm that incorporates prior knowledge of the underlying physics into the neural network architecture. In the context of option pricing, this means that PINNs can be trained to capture the underlying dynamics and correlations of multiple underlying assets, leading to more accurate and efficient pricing estimates.

Multi-asset option pricing involves the valuation of options contracts on more than one underlying asset. These options can have complex payoff structures and require the use of advanced mathematical models and computational techniques. Traditional methods for pricing multi-asset options, such as Monte Carlo simulations and analytic pricing models, can be computationally intensive and time-consuming.

PINNs offer a promising alternative for multi-asset option pricing due to their ability to learn the underlying physics of the problem and produce accurate pricing estimates in a fraction of the time required by traditional methods. By incorporating prior knowledge of the underlying physics, PINNs can reduce the number of simulations required to generate accurate pricing estimates, leading to significant cost savings for financial institutions.

In this report, we will explore the principles of multi-asset option pricing using PINNs. We will examine the underlying mathematics and physics of multi-asset option pricing and discuss how PINNs can be used to capture this information in their neural network architecture. We will also examine the advantages and limitations of PINNs for multi-asset option pricing along with highlighting several other empirical ways for option pricing. By the end of this report, we hope to provide a comprehensive overview of multi-asset option pricing with PINNs.

Option Theory

Before moving onto PINN, it is important to understand some basic terminologies. Option theory is a branch of mathematical finance that deals with the pricing and valuation of financial options. Financial options are contracts that give the holder the right, but not the obligation, to buy or sell an underlying asset at a predetermined price and time. The underlying asset can be anything from stocks, bonds, commodities, or currencies.

The principles of option theory are based on mathematical models that describe the behavior of financial markets and the underlying assets that are traded within them. These models involve complex mathematical equations that take into account factors such as market volatility, interest rates, and time decay.

One of the most widely used option pricing models is the Black-Scholes model, developed by Fischer Black and Myron Scholes in 1973. The Black-Scholes model is a mathematical formula that provides a theoretical estimate of the fair value of an option. It is based on the assumption that the price of the underlying asset follows a random walk and that the market is efficient.

Option Pricing Methods

Option pricing methods are mathematical models and techniques used to determine the fair value of financial options. These methods take into account various factors such as the underlying asset's price, volatility, time to expiration, and interest rates. The goal of option pricing is to estimate the theoretical price of an option that would be fair to both the buyer and seller of the contract.

There are several option pricing methods available, each with its own assumptions and limitations. The most commonly used option pricing methods include the Black-Scholes model,the binomial model,Finite difference method and the Monte Carlo simulation.

Monte-Carlo Simulation

Monte Carlo simulation is a widely used statistical technique that involves generating random samples to model complex systems or processes. It is a versatile and powerful tool that has applications in a wide range of fields, including finance, engineering, physics, and medicine.

In finance, Monte Carlo simulation is used to simulate the behavior of financial assets and model the risks associated with various investment strategies. It is particularly useful in valuing financial options, which have complex payoffs that depend on the price of the underlying asset at various points in time.

One of the strengths of Monte Carlo simulation is its ability to model complex systems and processes that cannot be easily modeled using other techniques. It is also a flexible method that can be adapted to various situations and scenarios.

$$dS = rSdt + \sigma SdW$$

where:

dW is a random variable following a normal distribution with mean 0 and standard deviation $\sqrt{dt}$

$$\text{payoff} = \max(S(T) - K, 0)$$

Finite Difference Method

The Finite Difference Method is a popular numerical technique used in option pricing. It involves discretizing the continuous-time option pricing equation into a grid of discrete points in both space and time. The values at these grid points are then solved using the finite difference equations to determine the price of the option.

The Finite Difference Method is a flexible technique that can be used to value a wide range of options, including European, American, and exotic options. It can also be used to model the impact of various factors, such as volatility and interest rates, on option prices.

One of the strengths of the Finite Difference Method is its ability to model complex payoffs and features of options, such as early exercise and barriers. It is also a powerful tool for sensitivity analysis, allowing users to assess the impact of changes in various input variables on the price of the option.

However, the Finite Difference Method can be computationally intensive and requires a significant amount of time and resources to implement. It also requires careful calibration of the model parameters to ensure accurate pricing.

Black-Scholes Model

The Black-Scholes Model is a mathematical model widely used in financial markets to price options contracts. It was first introduced by Fischer Black and Myron Scholes in 1973 and has become a cornerstone of modern finance theory.

The Black-Scholes Model is based on the assumption that the price of the underlying asset follows a geometric Brownian motion, meaning that its price changes randomly over time with a drift and volatility. It also assumes that the market is efficient, meaning that all investors have access to the same information and that the price of the option reflects the true value of the underlying asset.

The model uses a partial differential equation to determine the fair price of a European-style option, which gives the holder the right to buy or sell the underlying asset at a fixed price at the expiration date. The Black-Scholes Model takes into account various factors that impact option prices, including the price of the underlying asset, the strike price, the time to expiration, and the volatility of the underlying asset.

One of the strengths of the Black-Scholes Model is its simplicity and ease of use. It provides a quick and efficient way to calculate the fair price of an option and has become a widely accepted benchmark for option pricing. It is also a useful tool for risk management and hedging strategies.

However, the Black-Scholes Model has its limitations, including its assumptions about the behavior of the underlying asset and the market. It also only applies to European-style options and does not account for the impact of dividends and interest rates.

$$\boxed{\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} + \frac{\partial V}{\partial t} - rV = 0}$$

Single Asset Black Scholes Equation

$$\boxed{\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{n} A_{ij} S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} + r(t)\sum_{i=1}^{n} S_i \frac{\partial V}{\partial S_i} = r(t)V}$$

Multi Asset Black Scholes Equation

The Analytical solution for the one asset case is as follows.

For CALL :-

$$\boxed{C(S_t, t) = N(d_1)S_t - N(d_2)Ke^{-r(T-t)}}$$

For PUT :-

$$\boxed{\begin{aligned} P(S_t, t) &= Ke^{-r(T-t)} - S_t + C(S_t, t) \\ &= N(-d_2)Ke^{-r(T-t)} - N(-d_1)S_t \end{aligned}}$$

Where,

$$\boxed{\begin{aligned} d_1 &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \\ d_2 &= d_1 - \sigma\sqrt{T-t} \end{aligned}}$$

S	:	Current price of the underlying asset
t	:	Current time
r	:	Risk-free interest rate
σ	:	Volatility of the underlying asset
N	:	Cumulative distribution function of the standard normal distribution

Neural Network

Neural networks are a type of artificial intelligence that are modeled after the structure and function of the human brain. They consist of interconnected nodes, called neurons, that work together to process information and make decisions.

Neural networks have become increasingly popular in recent years due to their ability to learn from data and improve their performance over time. They are used in a wide range of applications, including image recognition, natural language processing, and financial modeling.

In financial modeling, neural networks are used to analyze large datasets and make predictions about future trends and events. They are particularly useful in complex financial scenarios, where traditional statistical methods may not be effective.

One of the strengths of neural networks is their ability to identify complex patterns in data that may not be apparent to humans. They are also highly adaptable and can be trained to recognize new patterns and trends as they emerge.

However, neural networks can be computationally intensive and require large amounts of data to achieve accurate results. They also require careful tuning of their parameters to ensure optimal performance.

Physics Informed Neural Network

Physics Informed Neural Networks (PINNs) are a type of neural network that combines the power of deep learning with the physical laws that govern a system. PINNs have been developed to address the limitations of traditional machine learning methods in solving complex physical problems.

In PINNs, the neural network architecture is designed to incorporate prior knowledge of the underlying physical principles governing the system being modeled. This helps to reduce the amount of data required to train the network and also ensures that the resulting solutions are physically consistent.

Physics Informed Neural Networks (PINNs) have recently emerged as a powerful tool for solving complex problems in finance, including option pricing. PINNs combine the power of deep learning with the principles of physics to produce accurate and physically consistent solutions.

In traditional option pricing models, such as the Black-Scholes model, the underlying asset price is assumed to follow a geometric Brownian motion, and the option price is calculated using partial differential equations (PDEs). However, in reality, the asset price may be influenced by other factors such as interest rates, dividends, and volatility. These factors may not be adequately captured by traditional models, leading to inaccuracies in option prices.

PINNs can be used to overcome these limitations by incorporating physical principles and constraints into the neural network architecture. This enables the network to learn the complex relationships between various factors that influence the option price and produce more accurate predictions.

One of the strengths of PINNs for option pricing is their ability to handle complex financial scenarios with multiple underlying assets and stochastic volatility. PINNs can be trained using historical data to predict future option prices and assess the risk associated with different option contracts.

However, PINNs for option pricing require careful calibration of their parameters to ensure optimal performance. They also require a deep understanding of the financial principles governing the system being modeled in order to design an appropriate neural network architecture.

General overview of the algorithm for solving Forward Problems using PINN :-

1. Defining the problem: The governing equations of the problem, initial and boundary conditions or other constraints.

2. Designing the neural network architecture: Designing a neural network architecture that is appropriate for the problem is very important. It involves selecting the number and type of layers, the number of neurons in each layer, and the activation functions.

3. Train the network: Train the network using data, In case of boundary value problems we use the constraints to generate collocation points. This involves minimizing a loss function that represents the difference between the predicted output of the network and the actual output.

4. Incorporate physical constraints: To ensure that the solutions obtained by the network are physically realistic, it is important to incorporate physical constraints into the training process. This can include incorporating the governing equations into the loss function, as well as incorporating any boundary conditions or other constraints.

5. Validating the results: Once the network has been trained, it is important to validate the results to ensure that they are accurate and physically consistent. This can be done by comparing the predicted results to analytical solution if available or other known solutions.

Now let us see some implementations of PINN on some real world examples.

Solving 1-D Heat Equation using PINNs

In traditional numerical methods, the 1D heat equation is typically solved using finite difference or finite element methods, which require explicit discretization of the domain. In contrast, PINNs do not require explicit discretization, making them faster and more flexible. The network architecture can be designed to incorporate prior knowledge of the physical principles that govern the system being modeled, resulting in more accurate and physically consistent solutions.

One of the strengths of PINNs for solving the 1D heat equation is their ability to incorporate boundary conditions and other constraints into the training process. This helps to ensure that the resulting solutions are physically realistic and can be used to make accurate predictions about the behavior of the system being modeled.

We have taken Burgers' equation as an example for 1-D Heat Equation case to solve using Physics Informed Neural Network. The Burgers' equation is a fundamental equation in fluid dynamics, which describes the behavior of a viscous fluid. Solving Burgers' equation is important for understanding fluid mechanics and has applications in many fields, including aerospace engineering, oil and gas exploration, and climate modeling.

In one space dimension, the Burger’s equation along with Dirichlet boundary conditions reads as

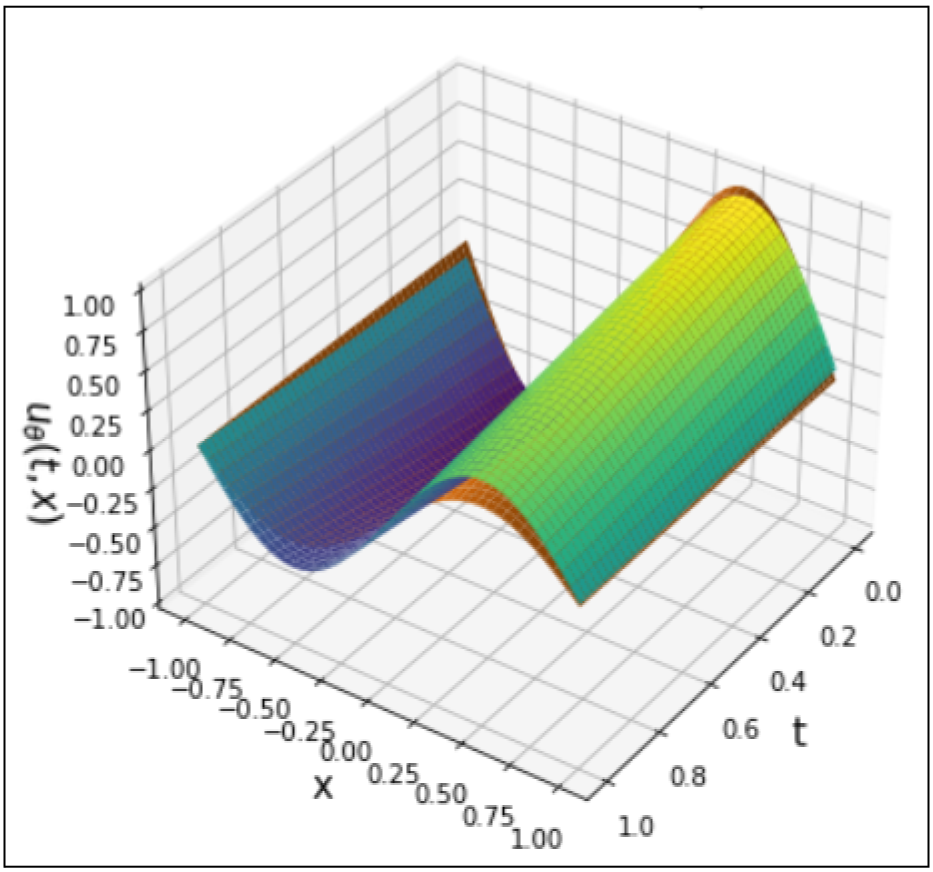
$$\boxed{\begin{aligned} &u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \\ &u(0, x) = -\sin(\pi x), \\ &u(t, -1) = u(t, 1) = 0. \end{aligned}}$$

Let us define f (t, x) to be given by

$$\boxed{f := u_t + u u_x - (0.01/\pi) u_{xx},}$$

The Loss Function for the model will be

$$\boxed{J(f) = \|\partial_t f + \mathcal{L} f\|^2_{2,[0,T]\times\Omega} + \|f - g\|^2_{2,[0,T]\times\partial\Omega} + \|f(0,\cdot) - u_0\|^2_{2,\Omega}\,.}$$



For 1-Dimensional Case this was the predicted solution of PINN Along with the Analytical Solution

Solving Parabolic PDE using PINNs

The parabolic heat equation is a fundamental equation in heat transfer, which describes the distribution of heat in a material over time. Solving the parabolic heat equation is important for understanding heat transfer and has applications in many fields, including engineering, physics, and materials science.

When solving partial differential equations (PDEs) using physics-informed neural networks (PINNs), the Jacobian matrix plays a critical role in enforcing the PDE constraint on the neural network solution. The Jacobian matrix is defined as the matrix of all first-order partial derivatives of a vector-valued function.

To calculate the Jacobian matrix in the context of solving a parabolic PDE using PINNs, we first need to define the loss function for the neural network. The loss function typically consists of two terms: a mean squared error term, which measures the difference between the predicted solution and the true solution at a set of collocation points, and a PDE residual term, which measures the difference between the predicted solution and the PDE constrained at a set of collocation points.

To calculate the PDE residual term, we need to evaluate the Jacobian matrix of the neural network output with respect to the spatial coordinate and the time variable. This involves computing the first-order partial derivatives of the neural network output with respect to each input variable. The resulting Jacobian matrix can then be used to calculate the PDE residual term, which is typically defined as the dot product of the Jacobian matrix with the gradient of the predicted solution with respect to time, minus the Laplacian of the predicted solution with respect to space.

By including the PDE residual term in the loss function and minimizing the loss function with respect to the neural network parameters, we can obtain a solution to the parabolic PDE that satisfies the PDE constrained at a set of collocation points.

The parabolic heat equation can be expressed as:

∂u/∂t = α (∂^2u/∂x^2 + ∂^2u/∂y^2)

Consider a parabolic PDE with d spatial dimensions:

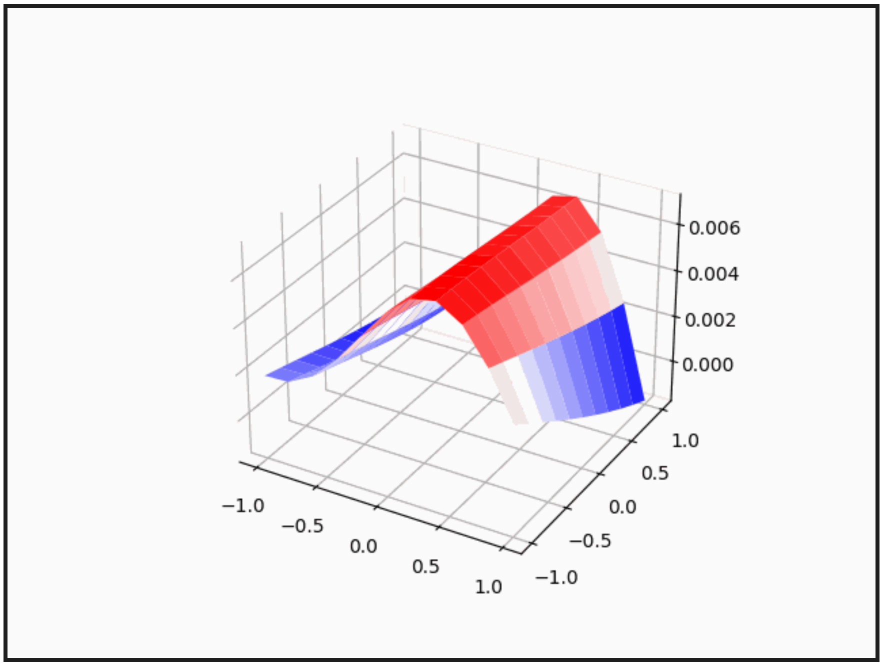
$$
\begin{aligned}
&\frac{\partial u}{\partial t}(t,x) + \mathcal{L}u(t,x) = 0, \qquad (t,x) \in [0,T] \times \Omega, \\
&u(t=0,x) = u_0(x), \\
&u(t,x) = g(t,x), \qquad x \in \partial\Omega,
\end{aligned}
$$

The Jacobian would be:

$$
H\, f\,(x,\, y) \equiv \begin{vmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x\, \partial y} \\ \dfrac{\partial^2 f}{\partial y\, \partial x} & \dfrac{\partial^2 f}{\partial y^2} \end{vmatrix}
$$

The Loss Function for the model will be

$$
J(f) = \|\partial_t f + \mathcal{L} f\|^2_{2,[0,T]\times\Omega} + \|f - g\|^2_{2,[0,T]\times\partial\Omega} + \|f(0,\cdot) - u_0\|^2_{2,\Omega}\,.
$$



For 2-Dimensional Case this was the predicted solution of PINN along the time axis.

Solving Black–Scholes Equation using PINNs

The Black-Scholes equation is a well-known partial differential equation (PDE) used to model the pricing of financial options. It describes the time evolution of the price of a financial asset, such as a stock or a commodity, as a function of its current price, volatility, interest rate, time to maturity, and strike price. Solving the Black-Scholes equation is important for understanding the behavior of financial markets and for pricing financial derivatives.

Physics-Informed Neural Networks (PINNs) have shown promise as a method for solving the Black-Scholes equation. The basic approach is to use a neural network to approximate the solution to the Black-Scholes equation and enforce the PDE constraint by adding a residual term to the loss function. The residual term is typically based on the PDE and the boundary conditions of the problem, which are often known.

1-D case

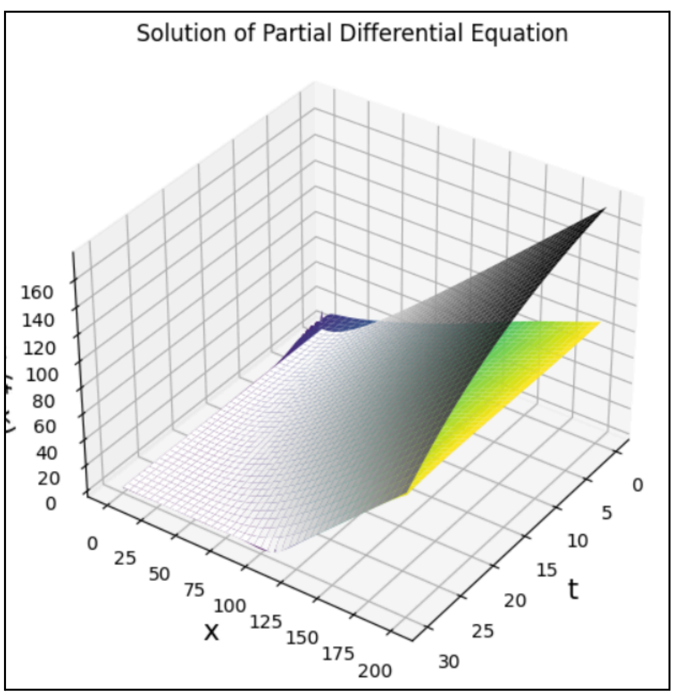
$$\boxed{\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} + \frac{\partial V}{\partial t} - rV = 0}$$

Initial condition:

$$V(S,0) = f(S)$$

Boundary conditions:

As S approaches 0, V(S,t) approaches 0
As S approaches infinity, V(S,t) approaches S



N-D case (here N=2)

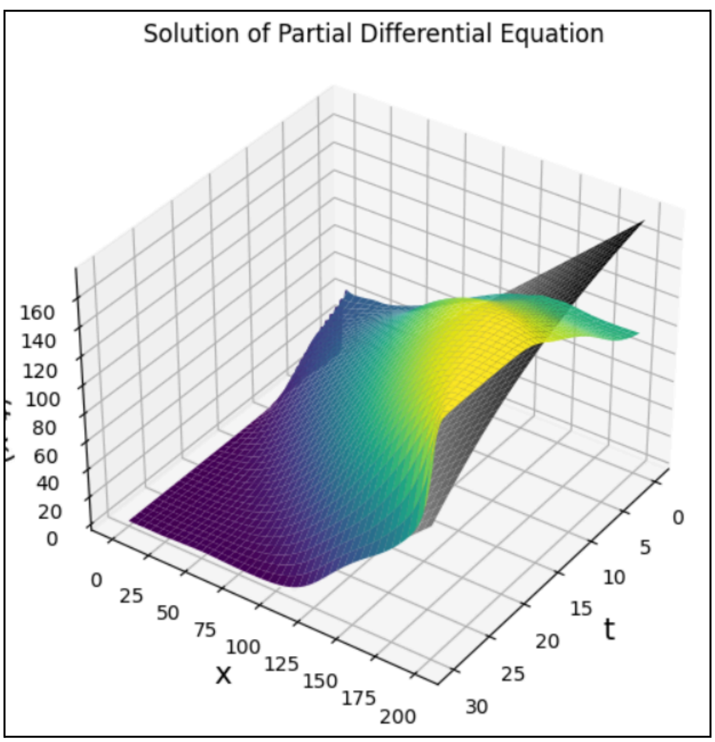
$$\boxed{\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{n} A_{ij} S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} + r(t)\sum_{i=1}^{n} S_i \frac{\partial V}{\partial S_i} = r(t)V}$$

Initial condition:

V(S1, S2, ..., Sn, T) = max(min(S1, S2, ..., Sn) - K, 0)

Boundary conditions:

V(0, S2, ..., Sn, t) = 0
V(S1, 0, ..., Sn, t) = 0
.
.
.
V(S1, S2, ..., 0, t) = 0



Solving Black–Scholes Equation using FDM

The Black-Scholes model is a widely used mathematical model for pricing options, which assumes that the price of an underlying asset follows a geometric Brownian motion. One of the methods commonly used to solve the Black-Scholes equation is the finite difference method.

The finite difference method involves dividing the time and asset price domains into discrete intervals, and approximating the partial differential equation governing the option price using finite difference approximations. The method relies on a set of boundary and initial conditions, and iteratively solves for the option price at each time step.

The accuracy of the finite difference method depends on the number of intervals used, with higher numbers of intervals leading to more accurate results but also increasing computational complexity. Additionally, the method assumes that the underlying asset price follows a continuous path, which may not always hold true in real-world scenarios.

Predicting Volatility of BS Equation using PINNs

Solving final value problems is an important task in various fields of science and engineering, including finance, physics, and biology. A final value problem involves finding the solution of a differential equation that satisfies certain conditions on the final value of the solution.

To estimate the volatility using PINNs, the Black-Scholes equation is reformulated as a partial differential equation (PDE) with the volatility as the unknown function. The PINN is then used to solve the PDE with appropriate boundary conditions. The PINN loss function is designed to include the Black-Scholes equation and the boundary conditions, as well as additional terms to enforce smoothness and stability of the solution.

The use of PINNs to estimate the volatility in the Black-Scholes equation offers several advantages over traditional methods, such as finite difference and Monte Carlo simulations. PINNs can handle complex geometries and boundary conditions without the need for a mesh or discretization, and can learn from limited and noisy data. Furthermore, the use of physics-informed constraints ensures that the solution is physically meaningful and consistent with the underlying model.

Overall, using physics-informed neural networks to estimate the volatility in the Black-Scholes equation is a promising area of research that could improve the accuracy and efficiency of option pricing and risk management in finance.

PINNs vs FDM for option pricing

Both Physics-Informed Neural Networks (PINNs) and Finite Difference Methods (FDM) are numerical methods used for solving partial differential equations (PDEs) that arise in option pricing. However, there are several differences between these two methods.

Finite Difference Methods involve discretizing the solution domain and approximating the derivatives using finite difference approximations. This leads to a set of algebraic equations that can be solved using linear algebra techniques. FDMs have been extensively used in option pricing due to their simplicity and ease of implementation.

On the other hand, PINNs use a neural network to approximate the solution to the PDE and enforce the PDE constraint as a residual term in the loss function. PINNs do not require any grid or mesh generation, and they can handle complex geometries and boundary conditions. PINNs have shown promising results in solving PDEs, especially in problems with high-dimensional input spaces and complex boundary conditions.

When it comes to option pricing, both PINNs and FDMs have their advantages and disadvantages. FDMs are computationally efficient and can handle large-scale problems with ease. However, they may struggle with complex boundary conditions and require careful selection of the discretization parameters to ensure accuracy.

In contrast, PINNs can handle complex boundary conditions and do not require grid generation, but they may be computationally expensive and require careful tuning of the hyperparameters.

In summary, both PINNs and FDMs have their strengths and weaknesses in option pricing. The choice between these two methods depends on the specific problem at hand and the computational resources available.

Summary and Conclusions

In conclusion, the use of Physics Informed Neural Networks (PINNs) for pricing options via the Black-Scholes equation has shown promising results in recent years. By reformulating the Black-Scholes equation as a partial differential equation (PDE) and incorporating it into the PINN loss function, accurate and efficient solutions can be obtained without the need for meshing or discretization.

PINNs offer several advantages over traditional methods such as finite difference and Monte Carlo simulations, including faster computation times, higher accuracy, and the ability to handle more complex models. The use of PINNs for option pricing could potentially lead to better risk management and more informed decision-making in finance.

However, there are also limitations to the current state of research on PINNs for option pricing. One limitation is the lack of interpretability of the models, which could make it difficult for practitioners to understand and justify their decisions. Additionally, more research is needed to explore the potential of PINNs for pricing more complex options, such as exotic and rainbow options.

Overall, the use of PINNs for pricing options via the Black-Scholes equation is a promising area of research that in future may help in accurate option pricing.

References

MA354 Computational Finance

Option Pricing Theory: Definition, History, Models, and Goals

Physics Informed Neural Networks in Computational Finance: High Dimensional Forward & Inverse Option Pricing

Deep Learning-Based Numerical Methods for High-Dimensional Parabolic Partial Differential Equations and Backward Stochastic Differential Equations

Solving high-dimensional eigenvalue problems using deep neural networks: A diffusion Monte Carlo like approach

Numerically Solving Parametric Families of High-Dimensional Kolmogorov Partial Differential Equations via Deep Learning