it. The residual of a PDE equivalent to the difference between the components of a PDE. We can visualize the residuals by examining the heat equation

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} \tag{1}$$

If a PINN were to perfectly solve the heat equation, it would generate a function whose partial derivative with respect to time is equal to that same function's second partial derivative with respect to space, multiplied by some constant. However, a PINN will likely fail to generate a function that exactly fits these requirements, so we can quantify the difference between the components of the PDE, letting u_θ be the approximated solution

$$\frac{\partial u_\theta}{\partial t} - a^2 \frac{\partial^2 u_\theta}{\partial x^2} = residual \tag{2}$$

The goal for any PINN is to have a run time shorter than benchmark methods with residuals that are acceptably small.

Another distinction between the way in which basic neural networks and PINNs work are the way in which they process training data. Typical neural networks use data that is collected, cleaned, and organized in a tabular fashion. This data is then utilized by the neural network to modify its weights and biases until it is able to perform optimally. However, PINNs that are used exclusively to solve PDEs do not make use of collected data. Instead, they use what are known as collocation points. Collocation points are randomly sampled points within a predefined domain which are used to enforce the conditions set by the PDE embedded within the loss function. A PINN will attempt to generate a function which approximates a solution to a PDE by inputting each collocation point into its function approximation and computing the resulting residual. Since we're using mean squared error for our loss function, we square each residual as well before summing them up and finding the mean error.

Since PDEs also have associated boundary and initial/terminal conditions to ensure unique solutions, boundary and initial/terminal training points are also generated to enforce these conditions. The difference between the boundary and initial/terminal points generated by the neural network approximation and the true values are treated in an identical manner as collocation points within the loss function.

3. Approach

The algorithms used to build PINNs for the purpose of solving for the price of American options and GMIB Variable Annuities were repurposed from a program originally used to solve the Burgers and Eikonal equations [1]. The algorithms can be described as follows

3.1 Initialization

We begin by setting the constants to be used in solving for the American option premium, namely the option type (Put or Call), the strike price (K), the time to expiration (T), the volatility of the underlying stock (σ), the risk-free interest rate (r),