The next step involves initializing our neural network and determining features that should be included within our model. After much trial and error, we found that a neural network architecture consisting of 3 layers each with 20 neurons returned predictions accurately without compromising too much in terms of speed. A layer which scales each piece of input data to the dimensions of our PDE domain was also incorporated. The swish activation function was found to outperform the hyperbolic tangent, ReLU, and ELU activation functions in our numeric tests, and was therefore chosen to be our activation function of choice.

3.2 The Loss Function

To begin computing the error our neural network is producing, we must define a function that is able to compute the residuals of our PDE. Tensorflow's ability to easily compute derivatives lead to a simple implementation where first and second partial derivatives are obtained and can be easily utilized for determining residuals.

Now that we've implemented a function for computing the residuals of our collocation points, we can set up our loss function. The loss for our PINN is defined exactly as described in [1] as the sum of the residuals of each collocation points, in addition to the sum of the squared differences between the true initial/boundary condition points and the initial/boundary points computed by the PINN. We must also create a function capable of obtaining the gradient of the loss function we've implemented to determine the direction in which to move to minimize errors efficiently. This is again made possible through Tensorflow's automatic differentiation capabilities, and is easily implemented by following the routine established in [1].

3.3 The Training and Discretization Process

Although a large number of learning methodologies exist, we chose to use a simple piece-wise learning technique which decreases the learning rate halfway through the training process. For the majority of our numerical tests, we found that 200 - 400 training epochs was enough to have obtain an appropriate level of accuracy within our output. Following the training process, we needed some way to visualize the results of our predictions, so we chose to discretize our PINN's predictions by creating a meshgrid upon which to map our PINN's solutions onto. We then plotted our solution in 2 dimensions, where the x-axis represents the possible stock price values, the y-axis represents the value of the option, and each graph represents a slice in time. We also plotted in 3 dimensions, incorporating time as the 3rd dimension.

3.4 Applications to the GMIB Variable Annuity

Before jumping into the mathematical formulations of how a PINN for a GMIB Variable Annuity works, we should first define some notation.

$A_t :=$ the value of the investment account which is determined by the underlying stock's value, often called the account value
$G_t :=$ a value which can never be withdrawn but is used to calculate the amount of received in case a policyholder chooses to annuitize their investments, commonly referred to as the benefit base