Whack-a-Mole Learning: Physics-Informed Deep Calibration for Implied Volatility Surface

Kentaro Hoshisashi, Carolyn E. Phelan and Paolo Barucca

October 8, 2024

Whack-a-mole Learning: Physics-Informed Deep Calibration for Implied Volatility Surface

Kentaro Hoshisashi*
Department of Computer Science
University College London
London, United Kingdom
k.hoshisashi@ucl.ac.uk

Carolyn E. Phelan
Department of Computer Science
University College London
London, United Kingdom
carolyn.phelan.14@ucl.ac.uk

Paolo Barucca
Department of Computer Science
University College London
London, United Kingdom
p.barucca@ucl.ac.uk

***Abstract*—Calibrating the Implied Volatility Surface (IVS) using sparse market data is an essential task for option pricing in quantitative finance. The calibrated values must provide a solution to a specified partial differential equation (PDE) in addition to obeying no-arbitrage conditions modelled by individual differential inequalities. However, this leads to a multi-objective optimization problem, which emerges in Physics-Informed Neural Networks (PINNs) as well as in our generalized framework. In order to address this problem, we propose a novel calibration algorithm called Whack-a-mole Learning (WamL), which integrates self-adaptive and auto-balancing processes for each loss term. The developed algorithm realizes efficient reweighting mechanisms for each objective function, ensuring alignment with constraints of price derivatives to achieve smooth surface fitting while satisfying PDE and no-arbitrage conditions. In our tests, this approach enables the straightforward implementation of a deep calibration method that incorporates no-arbitrage constraints, providing an appropriate fit for uneven and sparse market data. WamL also enhances the representation of risk profiles for option prices, offering a robust and efficient solution for IVS calibration.**

***Index Terms*—Multi-objective Learning, Physics-Informed Neural Networks, Option pricing, Implied Volatility Surface, Partial Differential Equations**

I. Introduction

In the recent deep learning revolution, deep neural networks are increasingly being applied to model and simulate complex systems, which parameterize unknown functions with artificial neural networks (ANNs). Among them, addressing forward and inverse problems involving partial differential equations (PDEs) gave rise to physics-informed neural networks (PINNs, (1)), which incorporate noisy data and physical laws as multi-task learning. The expansion of this approach opens up possibilities for meaningful real-world adoption, including in finance.

One typical PDE-related problem that could benefit from PINNs is the calibration of implied volatility surfaces (IVS). Previous research on finding IVS equivalent to option premiums using ANNs has primarily employed adapted global optimization (2; 3) and advanced network models (4; 5; 6). Several studies have addressed the calibration problems by penalizing the loss constraints (3; 6; 7; 8) or mapping pricing from model parameters (9; 10), and applied PINNs for the Black-Scholes PDE (11). Almost all approaches are considered a subset of IVS requirements, but not for all requirements. To accurately represent the options' market dynamics, IVS should be identified while satisfying PDE and financial principles, including no-arbitrage conditions as derivative constraints, leading to a multi-objective optimization problem.

The major challenge for such IVS calibration using deep learning is managing the combination difficulty between multi-scale losses and fine-tuning individual losses. To tackle this, we propose a novel calibration algorithm called Whack-a-mole Learning (WamL), which integrates self-adaptive and auto-balancing processes in deep learning and applies multi-objective calibration for the IVS as an expansion of PINNs. The developed algorithm balances efficient combination mechanisms for each objective function, ensuring alignment with PDE conditions and price derivative inequalities by layers. This is achieved by updating the loss function weights using gradients simultaneously with the updating of the neural network weights. In parallel, each objective loss is developed to be balanced appropriately by a self-attention mechanism. WamL aims for high interpretability and smooth surface fitting while satisfying PDE constraints and no-arbitrage conditions.

To summarize, our contributions are as follows:

- We introduce the WamL algorithm, which combines self-adaptive and loss-balancing learning algorithms to successfully apply PINNs to the IVS calibration problem. WamL accurately detects implied volatility characteristics while adhering to no-arbitrage and PDE conditions.
- The effectiveness of WamL is demonstrated in that it improves the interpolation, including smooth and appropriate sensitivity profiles. Additionally, by emphasizing the network's efforts to comply with the self-adaptive configurations in learning, WamL enhances the model's overall effectiveness and robustness.
- WamL contributes to enhancing the interpretation tool for IVS calibration and risk profiles through automatic differentiation, offering a robust and efficient solution by recent efficient computation advancements in quantitative finance applications.

This study tackles the challenges of multi-objective training and provides a novel approach to accurately represent options market dynamics while satisfying essential financial constraints.

II. IVS Calibration

A. Calibration Problem

The implied volatility surface (IVS) models values resulting from European option prices. We work with a complete filtered probability space $\left(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P}\right)$ where $\mathbb{P}$ is an associated risk-neutral measure. The price of a European call option[1] C at time t is defined as

$$C = e^{-r\tau}\mathbb{E}\left[(S_T - K)^+ \mid \mathcal{F}_t\right], \tag{1}$$

where S_t is the underlying price at t, K is the strike price, $\tau = T - t$ is the time to maturity T, and r is the deterministic risk-free rate. In the Black-Scholes model (12), implied volatility σ_{imp} is used to model the price for $K, \tau \in [0, \infty)$:

$$\begin{aligned} C_{\text{BS}}(\sigma_{\text{imp}}) &= S_t N(d_+) - e^{-r\tau} K N(d_-), \\ d_\pm &= \frac{\ln(e^{-r\tau} S_t/K) \pm \left(\sigma_{\text{imp}}^2/2\right)\tau}{\sigma_{\text{imp}}\sqrt{\tau}}, \end{aligned} \tag{2}$$

where $N(\cdot)$ is the cumulative normal distribution function.

To generalize for K and τ, (13) proposed the local volatility (LV) model, where European option prices satisfy the PDE

$$f := rK\frac{\partial C}{\partial K} - \frac{1}{2}\sigma_{\text{LV}}^2(K,\tau)K^2\frac{\partial^2 C}{\partial K^2} + \frac{\partial C}{\partial \tau} = 0, \tag{3}$$

with initial and boundary conditions:

$$C_{\tau=0} = (S_T - K)^+, \quad \lim_{K\to\infty} C = 0, \quad \lim_{K\to 0} C = S_t. \tag{4}$$

Plugging this into (2) yields a conversion function between σ_{imp} and σ_{LV} with respect to K and τ,

$$\sigma_{\text{LV}}^2(K,\tau) = \frac{\sigma_{\text{imp}}^2 + 2\sigma_{\text{imp}}\tau\left(\frac{\partial\sigma_{\text{imp}}}{\partial\tau} + rK\frac{\partial\sigma_{\text{imp}}}{\partial K}\right)}{1 + 2d_+K\sqrt{\tau}\frac{\partial\sigma_{\text{imp}}}{\partial K} + K^2\tau\left(d_+d_-\left(\frac{\partial\sigma_{\text{imp}}}{\partial K}\right)^2 + \sigma_{\text{imp}}\frac{\partial^2\sigma_{\text{imp}}}{\partial K^2}\right)} \tag{5}$$

to be fit with the PDE (3).

In this study, we define IVS calibration as identifying the multivariate function respecting the prices $\Phi(x)$, associated with IVS as a function $\varphi(x) \geq 0$ with inputs $x := (K, \tau)$,

$$\Phi(x) = C_{\text{BS}}(K, \tau, \varphi(x)). \tag{6}$$

The inverse problem of the IVS is that, given limited options prices, we would like to identify the implied volatility function with respect to x, redefined as $\sigma_{\text{imp}}(x)$, to fit with that premium resulted by C_{BS} also satisfy PDE. Based on (2$\sim$5), once φ is determined, we can analytically obtain the option price Φ. Furthermore, Φ is second differentiable whenever φ is second differentiable, allowing the representation of the PDE in (3). The challenge of this problem is the scarcity of option price data. To overcome this challenge, one possibility is to interpolate/extrapolate the price data or add further information relevant to the problem.

[1]We focus on European call options, but our discussion can theoretically apply to European put options and their synthesized products.

B. No-Arbitrage Constraints for European Options

The calibration of option prices is limited by sparse data and, as discussed, should obey the constraints imposed by no-arbitrage conditions. The no-arbitrage principle states that market prices should prevent guaranteed returns above the risk-free rate. We consider the necessary and sufficient conditions for no-arbitrage presented in (14), which allows us to express the call option price as a two-dimensional surface appropriately. These necessary and sufficient conditions for no-arbitrage are represented as non-strict inequalities for several first and second derivatives,

$$-e^{-r\tau} \leq \frac{\partial C}{\partial K} \leq 0, \quad \frac{\partial^2 C}{\partial K^2} \geq 0, \quad \frac{\partial C}{\partial \tau} \geq 0. \tag{7}$$

From the above, no-arbitrage conditions require these derivatives to have a specific sign. Standard architecture does not automatically satisfy these conditions when calibrating with a loss function, which is simply based on the mean squared error (MSE) for the prices.

III. Multi-Objective Optimization

IVS calibration is a complex problem that requires fitting a model to PDE and no-arbitrage conditions, which impose additional constraints on the derivative values of the function surface. With the exponential growth in computing power, deep neural networks are increasingly being applied to model and simulate such PDE-based systems.

A. Physics-Informed Neural Networks (PINNs)

Physics-Informed Neural Networks (PINNs), introduced in (1), incorporate underlying physical laws into neural network architecture through PDEs, forming a new class of data-efficient universal function approximators.

We consider a parameterized PDE system given by:

$$\begin{aligned} &\boldsymbol{f}[\Phi(\mathbf{x})], \mathbf{x} \in \Omega, \\ &\boldsymbol{\mathcal{B}}[\Phi(\mathbf{x})], \mathbf{x} \in \partial\Omega, \end{aligned} \tag{8}$$

where $\boldsymbol{f}$ denotes the set of PDE residuals that include differential operators, $\boldsymbol{\mathcal{B}}$ represents the set of boundary conditions, and Ω and $\partial\Omega$ are the spatial domain and the boundary, respectively.

PINNs solve this PDE system as an optimization problem by minimizing the total loss using an artificial neural network in a deep learning context:

$$\mathcal{L} := \mathcal{L}_0 + \mathcal{L}_b + \mathcal{L}_f, \tag{9}$$

$$\begin{aligned} \mathcal{L}_0 &:= \frac{1}{N_0}\sum_{i=1}^{N_0}\left|y^{(i)} - \Phi(\mathbf{x}^{(i)})\right|^2, \\ \mathcal{L}_b &:= \frac{1}{N_b}\sum_{i=1}^{N_b}\left|\mathcal{B}[\Phi(\mathbf{x}^{(i)})]\right|^2, \\ \mathcal{L}_f &:= \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f[\Phi(\mathbf{x}^{(i)})]\right|^2, \end{aligned} \tag{10}$$

where $\mathcal{L}_0$ is defined as the error between observed and modelled values, $\mathcal{L}_b$ enforces the periodic boundary conditions, and $\mathcal{L}_f$ enforces the structure imposed by PDE residual values at a finite set of collocation points.

B. Derivative-Constrained PINNs (DCPINNs)

We now consider the generalization of the PINNs with partial derivative inequality conditions Derivative-Constrained PINNs (DCPINNs) following (15). We assume

$$\boldsymbol{h}[\Phi(\mathbf{x})], \mathbf{x} \in \Omega \tag{11}$$

where $\boldsymbol{h}[\cdot]$ represents a set of differential operators acting on an inequality equation. To fit with inequality constraints, a loss $\mathcal{L}_h$ to be minimized is defined as,

$$\mathcal{L}_h := \frac{1}{N_h} \sum_{i=1}^{N_h} \gamma \circ \left| h\left(\Phi(\mathbf{x}^{(i)})\right) \right|^2. \tag{12}$$

$$\gamma(x) = \begin{cases} x, & \text{if inequality is not satisfied} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where $\mathcal{L}_h$ enforces the structure imposed by inequality with differential operators at a finite set of collocation points, and γ reflects the loss whether the inequality is satisfied or not.

IV. Whack-a-mole Learning (WamL)

IVS calibration aims to identify the surface $\Phi_\theta(x)$ by finding the optimal parameter set θ with inputs $x := (K, \tau)$. Standard architecture does not automatically satisfy all PDEs and no-arbitrage conditions. We consider optimization problems that include multiple losses, not only for the surface $\Phi_\theta(x)$ but also for its derivatives. The total cost to be minimized is,

$$\mathcal{L} := \lambda_0 \hat{\mathcal{L}}_0 + \lambda_b \hat{\mathcal{L}}_b + \underbrace{\lambda_f \hat{\mathcal{L}}_f}_{\text{PDE residuals}} + \underbrace{\Sigma_\alpha \lambda_{h_\alpha} \hat{\mathcal{L}}_{h_\alpha}}_{\text{No-arbitrage penalties}}, \tag{14}$$

$$\hat{\mathcal{L}}_0 := \frac{1}{N_0} \sum_{i=1}^{N_0} m_0^{(i)} \left| C_0^{(i)} - \Phi_\theta\left(x_0^{(i)}\right) \right|^2, \tag{15}$$

$$\hat{\mathcal{L}}_b := \frac{1}{N_b} \sum_{i=1}^{N_b} m_b^{(i)} \left| C_b^{(i)} - \Phi_\theta\left(x_b^{(i)}\right) \right|^2, \tag{16}$$

$$\hat{\mathcal{L}}_f := \frac{1}{N_f} \sum_{i=1}^{N_f} m_f^{(i)} \left| f\left(\Phi_\theta(x^{(i)})\right) \right|^2, \tag{17}$$

$$\hat{\mathcal{L}}_{h_\alpha} := \frac{1}{N_{h_\alpha}} \sum_{i=1}^{N_{h_\alpha}} m_h^{(i)} \left| \gamma_\alpha \circ h_\alpha\left(\Phi_\theta(x^{(i)})\right) \right|^2. \tag{18}$$

where $C_0^{(i)}$ is the observed premium, $i = 1, \ldots, N_0$ from the observed dataset, and $\hat{\mathcal{L}}_{h_\alpha}$ represents each penalty term corresponding to (7), i.e., $\mathcal{L}_{h_{\mathrm{K}}}$, $\mathcal{L}_{h_{\mathrm{KK}}}$ and $\mathcal{L}_{h_\tau}$ correspond to the terms in (7), respectively. The key modifications from standard PINNs configurations are the multiplier λ for each loss and the weights m for each individual loss in each categorized loss.

The concept of Whack-a-mole Learning (WamL), which is proposed in this study, involves a combination of two "whacks" in the learning process. The first "whack" intensifies the gradient of individual losses within categorized losses to enhance local constraints, especially the objective for inequality derivative constraints. The second "whack" fixes the multi-scale imbalance between categorized losses in (15∼18), particularly to mitigate the impact of changing gradient values from epoch to epoch due to the inequality feature in (18).

In line with the neural network philosophy of self-adaptation, WamL applies a straightforward procedure with fully trainable weights to generate multiplicative soft-weighting and attention mechanisms, inspired by (16; 17). The first "Whack" introduces self-adaptive weighting that updates the loss function weights via gradient ascent concurrently with the network weights. We minimize the total cost with respect to θ but also maximize it with respect to the self-adaptation weight vectors m at the k-th epoch,

$$m_\beta^{(j)}(k+1) = m_\beta^{(j)}(k) + \eta_m \nabla_{m_\beta^{(j)}} \hat{\mathcal{L}}_\beta(k). \tag{19}$$

where β specifies each loss $\beta \in \{0, b, f, h_{rmK}, h_{rmKK}, h_{rm\tau}\}$ and η_m is the learning parameter. In the learning step, the derivatives with respect to self-adaptation weights are increased when the constraints of derivative terms are violated. In parallel, for the second "Whack", loss-balancing employs the following weighting function in the loss function based on Eq (14). Considering updated at the k-th epoch,

$$\lambda_\beta(k+1) = \begin{cases} 1, & \text{if } \overline{\left|\nabla_\theta \hat{\mathcal{L}}_\beta(k)\right|} = 0 \\ \frac{1}{2}\left(\lambda_\beta(k) + \frac{\sum_\beta \overline{\left|\nabla_\theta \hat{\mathcal{L}}_\beta(k)\right|}}{\overline{\left|\nabla_\theta \hat{\mathcal{L}}_\beta(k)\right|}}\right), & \text{otherwise} \end{cases} \tag{20}$$

where ∇_i is the partial derivative vector (gradient) with respect to the i-th input vector (value), and $\overline{|\cdot|}$ indicates the average of the absolute values of the elements in the vector. Note that the main reason for the choice of absolute values, instead of the squared values in (17), is to avoid overlooking outlier gradients from the individual losses, as most elements of $\nabla \mathcal{L}_h$ are zero values when the last stage of the training.

The proposed method automatically adjusts the weights of loss terms based on their relative magnitudes during training at regular intervals specified by the user throughout the training process. This adaptable strategy aims to maintain a balanced contribution of each loss term on the optimization procedure, potentially improving convergence and accuracy.

V. Neural Network Formulation

In the WamL approach, we apply a simple but deep feed-forward neural network architecture, specifically a multilayer perceptron (MLP). Let $L \geq 2$ be an integer representing the depth of the network; we consider a neural network constructed with one input vector, L hidden layers, and one output value. Both input values and an output variable are real numbers, i.e., $x \in \mathbb{R}^n$ and $y \in \mathbb{R}$. We can express the price function Φ in (6) which includes MLP as a multivariate function φ depending on the variables x, i.e., $\varphi : \mathbb{R}^2 \to \mathbb{R}$,

$$\varphi(x) = \varsigma \circ A_L \circ f_{L-1} \circ A_{L-1} \circ \cdots \circ f_1 \circ A_1(x), \tag{21}$$

where for $l = 1, \ldots, L$, $A_l : \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_l}$ are affine functions as $A_l(x_{l-1}) = W_l^\top x_{l-1} + b_l$, and d_l is the number of neurons in the layer l for $x_{l-1} \in \mathbb{R}^{d_{l-1}}$, with $W_l \in \mathbb{R}^{d_{l-1} \times d_l}$ and

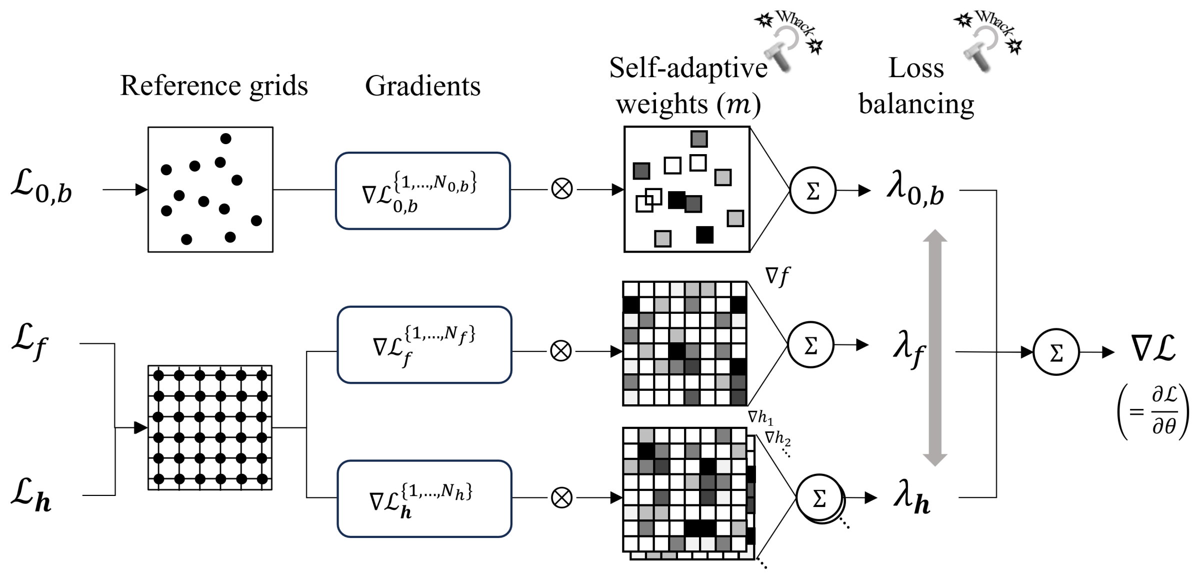
Whack
Whack
Reference grids
Gradients
Self-adaptive weights (m)
Loss balancing
$\mathcal{L}_{0,b}$
$\nabla\mathcal{L}_{0,b}^{\{1,\dots,N_{0,b}\}}$
$\lambda_{0,b}$
$\mathcal{L}_f$
$\nabla\mathcal{L}_f^{\{1,\dots,N_f\}}$
∇f
λ_f
$\nabla\mathcal{L}$
$\left(=\frac{\partial\mathcal{L}}{\partial\theta}\right)$
$\mathcal{L}_h$
$\nabla\mathcal{L}_h^{\{1,\dots,N_h\}}$
∇h_1
∇h_2
λ_h
Σ

Fig. 1. The whole network architecture of Whack-a-mole Learning (WamL).

$b_l \in \mathbb{R}^{d_l}$, $d_0 = n$, $d_L = 1$, and $x_0 = x$. f_l and ς are activation functions which are applied component-wise. To restrict the positivity of implied volatility and boundary conditions (4), we applied the softplus function for the last activation function $\varsigma(x) = \ln(1 + e^x)$. Given a dataset x, which includes a set of pairs $(x^{(i)}, C^{(i)}), i = 1, \dots N$, and a cost function $\mathcal{L}(x, \Phi)$, the network model Φ is found by fitting the parameters θ (i.e., $W$and b) which minimize the loss function.

A challenge lies in that Eq. (14) involves derivatives with respect to x, which are also functions of the parameters. When numerical approximation of derivatives is used, it could result in slow or inaccurate solutions. To address this, this study utilizes an extended backpropagation algorithm from (18) with Automatic Differentiation for the gradient in Eq. (14) through exact derivative formulations. This approach requires activation functions in the network to be second-order differentiable or higher. It is noted that functions like ReLU or ELU need slight additional consideration at non-differentiable singular points. If all activation functions are second-order differentiable, the same is true for the entire network, as shown in (19).

VI. Algorithms

This section introduces the algorithm for WamL for multi-objective problems as an expansion of the work in (1), which controls the inequality loss of the partial derivatives of a neural network function with respect to its input features.

Algorithm 1 exhibits WamL characteristics that set it apart from conventional learning methods. First, the computation points $\mathbf{x}_{\{f,h\}}$ for the derivatives of the MLP do not correspond with the points of the training dataset $\mathbf{x}_0$, due to the fact it is typically sparse and uneven. Instead, the algorithm adjusts the derivatives to fit mesh grids, hence capturing derivative data across a wide array of input features. Secondly, the objective function $\mathcal{L}$ depends not only on the MLP's direct output but also on its derivatives as specified in Eq.(14), all of which depend on identical network parameters. WamL facilitates balancing accurate calibration and categorized losses, which consist of PDE residuals and derivative inequalities, as described in Section IV.

VII. Experimental Design

A. Neural Network Setting and Training Configuration

In selecting appropriate network architectures and adopting suitable learning algorithms, we draw on approaches and configurations from (17; 20) to improve learning efficiency and accuracy. In our experiment, the network architecture φ is a deep setting with four layers ($L = 4$), each containing 64 neurons and using a hyperbolic tangent activation function for smooth activations. We apply non-dimensionalization for input as moneyness, $\hat{K} = K/F$ with forward price F, and use Glorot initialization for θ. Reference grids for PDE residuals (N_f) and no-arbitrage constraints ($N_{h_{\{\mathrm{K},\mathrm{KK},\tau\}}}$) are defined as dense rectangular mesh grids, equally distributed with 101 points in $\hat{K} \in [0, 2.5]$ and in $\tau \in [0, 5]$. We use the following hyperparameters: $\eta_m = 1.0$, $p_m, p_\lambda = 100$, and $k_{\max} = 10000$. For optimization, we employ the Adam optimization (21) with a weight decay setting, starting with a learning rate $\eta = 10^{-3}$ and an exponential decay with a decay rate of 0.9 for every $1,000$ decay steps.

The models compared in this study are

$$\mathcal{L} = \underbrace{\overbrace{\overbrace{\lambda_0\hat{\mathcal{L}}_0 + \lambda_b\hat{\mathcal{L}}_b}^{\text{Vanilla (-WamL)}} + \lambda_f\hat{\mathcal{L}}_f}^{\text{PINNs (-WamL)}} + \lambda_{h_{\mathrm{K}}}\hat{\mathcal{L}}_{h_{\mathrm{K}}} + \lambda_{h_{\mathrm{KK}}}\hat{\mathcal{L}}_{h_{\mathrm{KK}}} + \lambda_{h_\tau}\hat{\mathcal{L}}_{h_\tau}}_{\text{DCPINNs (-WamL)}} . \quad (22)$$

In computing, differentiable solvers have been developed in JAX/Flax (22; 23), which are suitable for inverse problems

Algorithm 1 Whack-a-mole Learning (WamL) Algorithm

```
Require: Dataset $(x_{0,b}, C_{0,b})^{\{1,\dots,N_{0,b}\}}, x_f^{\{1,\dots,N_f\}}, x_h^{\{1,\dots,N_h\}}$
Ensure: Neural network parameters $\theta$
  Consider a deep NN $\varphi_\theta(x)$ with $\theta$, and a loss function
      $$\mathcal{L} := \sum_\beta \lambda_\beta \hat{\mathcal{L}}_\beta\left(m_\beta, \mathbf{x}_\beta\left(, C_\beta\right)\right),$$
  where $\hat{\mathcal{L}}_\beta$ denotes the categorized loss with $\beta \in \{0, b, f, h_{\mathrm{K}}, h_{\mathrm{KK}}, h_\tau\}$, $m_\beta = \mathbf{1}$ are weight vectors for individual loss in $\hat{\mathcal{L}}_\beta$ and $\lambda_\beta = 1$ are dynamic parameters to balance between the different categorized loss.
  With pre-determined hyper-parameter values $\eta$, $\eta_m$, $p_m$, $p_\lambda$, $k_{\max}$, then use a gradient-based optimizer to update $\theta$ as:
  for $k = 1, \dots, k_{\max}$ do
      Compute $\nabla_\theta \hat{\mathcal{L}}_\beta(k)$ by automatic differentiation
      if $k \mod p_m = 0$ then
          Update $m_\beta$ by
              $$m_\beta^{(j)}(k+1) = m_\beta^{(j)}(k) + \eta_m \nabla_{m_\beta^{(j)}} \hat{\mathcal{L}}_\beta(k),$$
          where $m_\beta(k), \hat{\mathcal{L}}_\beta(k)$ shows at $k$-th iteration.
      end if
      if $k \mod p_\lambda = 0$ then
          Update $\lambda_\beta$ by
              $$\lambda_\beta(k+1) = \begin{cases} 1, & \text{if } \overline{\left|\nabla_\theta \hat{\mathcal{L}}_\beta(k)\right|} = 0 \\ \frac{1}{2}\left(\lambda_\beta(k) + \frac{\sum_\beta \overline{\left|\nabla_\theta \hat{\mathcal{L}}_\beta(k)\right|}}{\overline{\left|\nabla_\theta \hat{\mathcal{L}}_\beta(k)\right|}}\right), & \text{otherwise} \end{cases}$$
          where $\overline{|\cdot|}$ is mean of the absolute values of elements.
      end if
      Update the parameters $\theta$ via gradient descent, e.g.,
          $\theta(k+1) = \theta(k) - \eta \nabla_\theta \mathcal{L}(k)$.
  end for
  Return $\theta$
```

when modelling an unknown field. The experiments are conducted using Google Colab[2], which offers GPU computing on the NVIDIA Tesla T4 with 15 GB of video random access memory. The results reported in the figures represent the mean values obtained from 5 experimental runs, with synchronized seed values for random variables across the compared models.

B. Testing with Synthetic Data

The algorithm (i.e. WamL) developed for evaluating IVS was first tested on simulated values in a parameterized two-dimensional case of the surface interpolation problem. We took up the Stochastic Alpha Beta Rho (SABR) model introduced by (24) and prepared a sparse two-dimensional dataset to test our methodology. To fit a more realistic market situation, the following experiment utilized sparse (and uneven) grid data referring to actual historical traded grids from July 10 to July 14, 2023, and generated synthetic option premiums on the grids using Eq. 24 with $\{\alpha, \beta, \rho, \nu\} = \{0.2, 1.0, -0.4, 0.6\}$.

C. Real Data

We also conduct robust backtesting on extensive intraday options data for WamL using S&P 500 options. Our historical dataset comprises intraday traded prices of S&P 500 options for 248 business days from October 1, 2022, to September 30, 2023. We obtained about 500,000 data points on a daily basis via CBOE DataShop[3].

TABLE I
STATISTICS OF INTRADAY PRICES OF S&P 500 OPTIONS FOR 248 BUSINESS DAYS FROM OCTOBER 1, 2022, TO SEPTEMBER 30, 2023.

Elements [per day]	Mean	(min.)	(max.)
All count of intraday traded price	527,646	(183,418)	(734,441)
# of unique grids	6,516	(4,580)	(8,269)
% of call options	40.20 %	(35.16 %)	(47.09 %)
% of short term ($\tau < 1M$)	95.61 %	(91.40 %)	(97.07 %)
% of long term ($\tau > 1Y$)	2.62 %	(1.42 %)	(4.32 %)
% of near ATM ($\hat{K} \in [0.9, 1.1]$)	72.10 %	(64.43 %)	(76.95 %)
% of far OTM ($\hat{K} \notin [0.5, 1.5]$)	2.00 %	(1.17 %)	(3.05 %)

Table I shows that the distribution of input variables for intraday traded options is highly uneven. The vast majority of trades (about 95%) involve options with short expiration dates of less than a month. Moreover, over 70% of these trades are concentrated near the at-the-money strike. Compared to typical scenarios using mesh grid data, this uneven distribution presents a more challenging task for calibrating IVS.

VIII. RESULTS

A. Performance Evaluation on Synthetic Data

To assess the effectiveness of the proposed Whack-a-mole Learning (WamL) algorithm, we initially conducted experiments using synthetic data as described in Section VII-B. A sparse two-dimensional dataset was synthesized to simulate the challenges of limited data availability often encountered in real-world scenarios. The performance of WamL was compared against three baseline models with different loss formulations: Vanilla, PINNs, and DCPINNs. The evaluation metrics included calibration accuracy, adherence to no-arbitrage constraints, PDE residuals, and the quality of the interpolated volatility surface.

Figure 2 presents the IVS generated by each model for the synthetic dataset. WamL demonstrates an ability to accurately capture the intricate patterns of the volatility smile from very sparse data, closely resembling the ground truth surface. In contrast, as shown in Figure 3, the vanilla and PINNs models struggle to reproduce the complex shape of the surface, resulting in a more simplistic and less accurate representation in terms of derivatives of the surface, i.e., no-arbitrage constraints.

[2]Google Colab. http://colab.research.google.com

[3]CBOE DataShop, (2023). https://datashop.cboe.com/option-trades

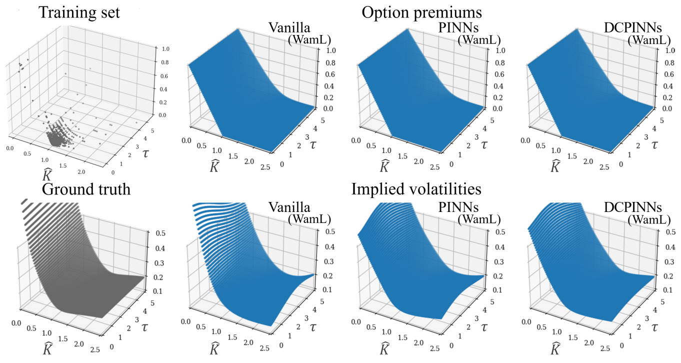

Fig. 2. Synthesized option premiums and inferred prices by calibrated models with WamL, and corresponding calibrated IVS by models.

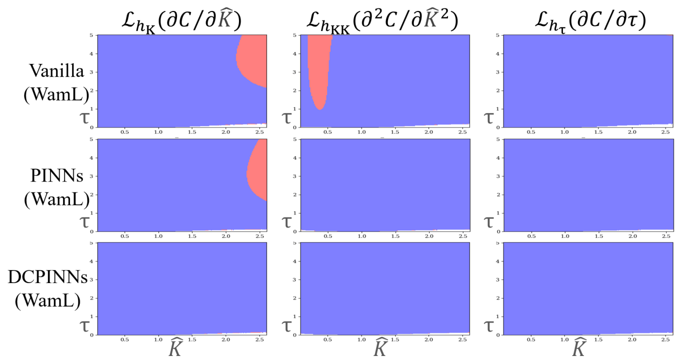

Fig. 3. Heatmaps corresponding to derivative inequalities by option premium surface. The red-colored area indicates a violation of no-arbitrage conditions.

TABLE II

CALIBRATION ERRORS ON A LOGARITHMIC SCALE (MEAN) ON SYNTHETIC DATA. **BOLD VALUES** INDICATE LOWEST (BEST) VALUES.

Models	$\mathcal{L}_0$	$\mathcal{L}_f$	$\mathcal{L}_{h_K}$	$\mathcal{L}_{h_{KK}}$	$\mathcal{L}_{h_\tau}$
Vanilla	**-7.3**	-11.9	-5.9	-2.4	**-inf***
PINNs	-4.3	-12.6	-1.8	-1.4	-12.5
DCPINNs	-5.7	-14.3	**-inf***	-10.1	**-inf***
Vanilla-WamL	**-7.3**	-11.4	-5.5	-2.3	-10.9
PINNs-WamL	-3.7	-14.8	-2.9	**-inf***	-7.0
DCPINNs-WamL	-5.6	**-15.2**	**-inf***	**-inf***	**-inf***

*Negative infinity (-inf) in a logarithm scale indicates zero value.

Quantitative analysis of the calibration errors reveals that WamL outperforms the baseline models. Table II summarizes the categorized losses for each model in relative errors. Note that each loss is not applied to self-adaptive weights or loss-balancing multipliers, i.e., we applied $m_\beta = \mathbf{1}$ and $\lambda_\beta = 1$. This allows us to isolate the effects of each component and understand their contribution to the overall model performance. WamL achieves balanced lower errors, indicating its superior calibration accuracy. The incorporation of physics-informed constraints and the use of automatic differentiation enable WamL to effectively leverage the available data points and produce a highly accurate interpolation of the IVS.

B. Backtesting on Real Market Data

The backtesting procedure involved calibrating the IVS using WamL and the baseline models for each trading day as presented in Section VII-C. The calibrated surfaces were then used to assess the accuracy of option price predictions, PDE residual losses, and adherence to no-arbitrage constraints.

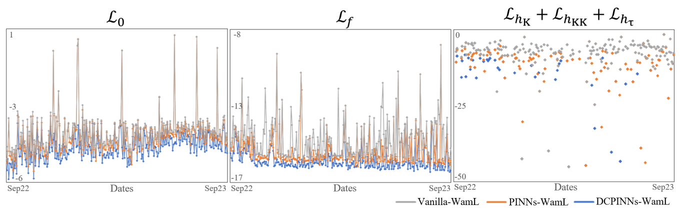

Fig. 4. Backtesting results. Daily calibration errors (logarithmic scale) on S&P 500 options data from October 1, 2022, to September 30, 2023.

Figure 4 illustrates the performance of WamL and the baseline models over the backtesting period. WamL consistently achieves more balanced calibration errors than the other models, which aligns with the results from synthetic data experiments. The superior performance of WamL can be attributed to its robustness in effectively incorporating market dynamics and no-arbitrage constraints into the calibration process.

TABLE III

CALIBRATION ERRORS ON A LOGARITHMIC SCALE (MEAN) ON REAL DATA. **BOLD VALUES** INDICATE LOWEST (BEST) VALUES.

Models	$\mathcal{L}_0$	$\mathcal{L}_f$	$\mathcal{L}_{h_K}$	$\mathcal{L}_{h_{KK}}$	$\mathcal{L}_{h_\tau}$
Vanilla	**-4.6**	-13.6	-6.2	-3.3	-5.5
PINNs	-4.4	-14.2	-4.3	-4.8	-5.7
DCPINNs	-4.5	**-15.1**	-9.5	-9.9	-8.5
Vanilla-WamL	**-4.6**	-13.4	-6.2	-3.3	-5.5
PINNs-WamL	-4.2	-14.2	-5.1	-5.8	-6.6
DCPINNs-WamL	-4.4	-14.9	**-15.5**	**-46.6**	**-9.1**

Table III presents key performance metrics averaged over the entire backtesting period. DCPINNs-WamL achieves significantly lower errors, indicating a strong fit between the calibrated surface and the no-arbitrage constraints, while maintaining competitive performance in other metrics. This demonstrates WamL's ability to improve global fit while effectively balancing multiple objectives.

C. Risk Sensitivity Analysis

An important aspect of IVS calibration is the accurate estimation of risk sensitivities, such as delta, gamma, and theta. Delta measures the sensitivity of option prices to changes in the underlying asset price, gamma quantifies the rate of change of delta, and theta represents the sensitivity of option prices to changes in time to maturity. These risk sensitivities correspond to the inequalities in the no-arbitrage conditions, ensuring that the calibrated IVS adheres to fundamental financial principles.

Figure 5 compares the risk sensitivity profiles generated by WamL and baseline models. WamL reasonably estimates delta, gamma, and theta while satisfying no-arbitrage conditions and producing smooth, continuous risk sensitivity curves.

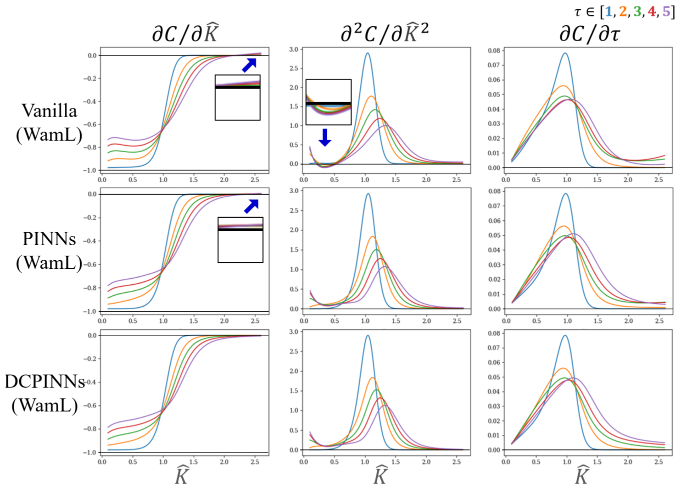

Fig. 5. Risk sensitivity profiles generated by different models. The figure shows the first and second derivatives with respect to moneyness $\mathcal{M}$ (left and centre) and the first derivative with respect to time to maturity τ (right). Each line represents a slice of the surface at different maturities $\tau \in [1, 2, 3, 4, 5]$.

In contrast, the baseline models show discrepancies and irregularities, with the vanilla model struggling the most. The risk sensitivity comparisons highlight WamL's ability to capture the underlying dynamics of the options market, which is crucial for informed risk management and hedging strategies in option portfolio construction and risk exposure management.

D. Loss Scalability and Computational Efficiency

Finally, we demonstrate the functioning of WamL during training and evaluate its computational efficiency. To demonstrate the effectiveness of WamL during training and evaluate its computational efficiency, we analyze the loss weights for each categorized loss across training epochs, as shown in Figure 6. The results reveal that WamL successfully reacts to derivative inequality violations and rectifies them using the gradient descent optimizer throughout the training process. Notably, we observe trade-offs between derivative constraints and premium accuracy, with the balancing weights being alternately adjusted to achieve an optimal balance. This dynamic adjustment of weights, which resembles "Whack-a-mole," highlights WamL's ability to adapt to the changing landscape of the optimization problem, ensuring that the model converges to a solution that satisfies the derivative constraints while maintaining high premium accuracy.

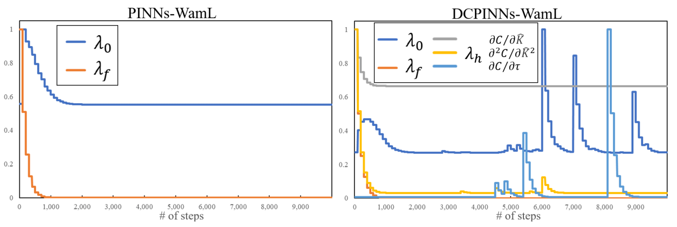

Fig. 6. Evolution of loss weights for each categorized loss during training. Values are normalized relative to the maximum value at each epoch.

TABLE IV
THE COMPUTATION TIMES (MEAN [SECONDS]) FOR CALIBRATING THE IVS ON DATASETS OF CHANGING SIZE

Models	Default(+WamL)	Half data
Vanilla(+WamL)	55.0 (58.1)	53.4 (56.3)
PINNs(+WamL)	115.4 (142.5)	114.1 (140.8)
DCPINNs(+WamL)	119.0 (148.6)	117.9 (147.7)

Table IV presents the computation times required by WamL and the baseline models for calibrating the IVS on datasets of decreasing size. The computational efficiency of WamL is evident from its ability to handle these complex optimization challenges without significant overhead. The efficient use of automatic differentiation and the adaptive loss balancing approach contribute to WamL's fast convergence and reduced computational overhead. WamL also exhibits excellent scalability, as evidenced by the sublinear growth in computation time with respect to the dataset size. This scalability is crucial for handling the ever-increasing volumes of options data in modern financial markets, e.g., in real-time applications.

IX. CONCLUSIONS AND FUTURE WORK

This study introduced Whack-a-mole Learning (WamL), a novel calibration algorithm designed to address the multi-objective problem of derivatives, with a focus on implied volatility surface calibration in finance. WamL demonstrated the ability to capture appropriate patterns from limited sparse data by integrating self-adaptive and auto-balancing processes, enabling reweighting mechanisms for each loss term.

Experiments conducted on both synthetic and real market data highlight WamL's effectiveness in accurately modelling the dynamics of the options market using PDEs while adhering to no-arbitrage constraints and minimizing calibration errors. The adaptive loss balancing approach and the efficient use of automatic differentiation contributed to WamL's fast convergence and reduced computational overhead. Its superior performance, adherence to PDEs and no-arbitrage constraints, and computational efficiency make WamL a promising tool for practitioners in the financial industry.

Future research could focus on incorporating additional constraints and market information to enhance the realism of the model and better capture dependencies in options data. Although the soft constraints approach does not rigorously enforce each constraint in a multi-objective optimization context, WamL has demonstrated its ability to generate suitable surfaces when dealing with noisy real-world data and trade-off structures. While the experiments showcase the efficacy of the WamL algorithm, further research could provide insights into how WamL efficiently finds the minimum loss and explore its potential for application in other domains.

ACKNOWLEDGEMENT

The authors are thankful to the Department of Computer Science, University College London, for providing us with the resources to perform this case study.

APPENDIX

A. The SABR model

The SABR model in (24) is a typical parametric model, which can capture the market volatility smile and skewness and reasonably depict market structure. When F_t is defined as the forward price of an underlying asset at time t, the SABR model is described as

$$\begin{gathered} dF_t = \alpha_t F_t^{\beta} dW_t^1, \quad d\alpha_t = \nu \alpha_t \; dW_t^2, \\ \langle dW_t^1, dW_t^2 \rangle = \rho dt. \end{gathered} \tag{23}$$

Here, W_t^1, W_t^2 are standard Wiener processes, α_t is the model volatility, ρ is the correlation between the two processes, and ν is analogous to vol of vol. The additional parameter β describes the slope of the skewness. Essentially, the IVS in the SABR model is given by a series expansion technique associated with volatility form of (25)

$$\begin{aligned} &\sigma(K,\tau) = \\ &\frac{\alpha\left(1+\left(\frac{(1-\beta)^2}{24}\frac{\alpha^2}{(FK)^{1-\beta}}+\frac{1}{4}\frac{\rho\beta v\alpha}{(FK)^{(1-\beta)/2}}+\frac{2-3\rho^2}{24}v^2\right)\tau\right)}{(FK)^{(1-\beta)/2}\left[1+\frac{(1-\beta)^2}{24}\ln^2\frac{F}{K}+\frac{(1-\beta)^4}{1920}\ln^4\frac{F}{K}\right]}\frac{z}{\chi(z)}, \\ &z = \frac{v}{\alpha}(FK)^{(1-\beta)/2}\ln\frac{F}{K},\ \chi(z) = \ln\left(\frac{\sqrt{1-2\rho z+z^2}+z-\rho}{1-\rho}\right). \end{aligned} \tag{24}$$

REFERENCES

[1] 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*, vol. 378, pp. 686–707, 2019.

[2] S. Liu, A. Borovykh, L. A. Grzelak, and C. W. Oosterlee, "A neural network-based framework for financial model calibration," *Journal of Mathematics in Industry*, vol. 9, pp. 1–28, 2019.

[3] V. Choudhary, S. Jaimungal, and M. Bergeron, "Funvol: A multi-asset implied volatility market simulator using functional principal components and neural sdes," *arXiv preprint arXiv:2303.00859*, 2023.

[4] M. Bergeron, N. Fung, J. Hull, Z. Poulos, and A. Veneris, "Variational autoencoders: A hands-off approach to volatility," *The Journal of Financial Data Science*, vol. 4, no. 2, pp. 125–138, 2022.

[5] Y. Cao, X. Liu, and J. Zhai, "Option valuation under no-arbitrage constraints with neural networks," *European Journal of Operational Research*, vol. 293, no. 1, pp. 361–374, 2021.

[6] R. Cont and M. Vuletić, "Simulation of arbitrage-free implied volatility surfaces," *Available at SSRN*, 2022.

[7] A. Itkin, "Deep learning calibration of option pricing models: some pitfalls and solutions," *arXiv preprint arXiv:1906.03507*, 2019.

[8] D. Ackerer, N. Tagasovska, and T. Vatter, "Deep smoothing of the implied volatility surface," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11552–11563, 2020.

[9] W. McGhee, "An artificial neural network representation of the sabr stochastic volatility model," *Journal of Computational Finance*, vol. 25, no. 2, 2020.

[10] B. Horvath, A. Muguruza, and M. Tomas, "Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models," *Quantitative Finance*, vol. 21, no. 1, pp. 11–27, 2021.

[11] Y. Bai, T. Chaolu, and S. Bilige, "The application of improved physics-informed neural network (ipinn) method in finance," *Nonlinear Dynamics*, vol. 107, no. 4, pp. 3655–3667, 2022.

[12] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of political economy*, vol. 81, no. 3, pp. 637–654, 1973.

[13] B. Dupire *et al.*, "Pricing with a smile," *Risk*, vol. 7, no. 1, pp. 18–20, 1994.

[14] P. Carr and D. B. Madan, "A note on sufficient conditions for no arbitrage," *Finance Research Letters*, vol. 2, no. 3, pp. 125–130, 2005.

[15] K. Hoshisashi, C. E. Phelan, and P. Barucca, "No-arbitrage deep calibration for volatility smile and skewness," *arXiv preprint arXiv:2310.16703*, 2023.

[16] L. McClenny and U. Braga-Neto, "Self-adaptive physics-informed neural networks using a soft attention mechanism," *arXiv preprint arXiv:2009.04544*, 2020.

[17] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, "An expert's guide to training physics-informed neural networks," *arXiv preprint arXiv:2308.08468*, 2023.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[19] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural networks*, vol. 3, no. 5, pp. 551–560, 1990.

[20] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, "Physics-informed neural networks with hard constraints for inverse design," *SIAM Journal on Scientific Computing*, vol. 43, no. 6, pp. B1105–B1132, 2021.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[23] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, "Flax: A neural network library and ecosystem for JAX," 2023.

[24] P. S. Hagan, D. Kumar, A. S. Lesniewski, and D. E. Woodward, "Managing smile risk," *The Best of Wilmott*, vol. 1, pp. 249–296, 2002.

[25] F. Black, "The pricing of commodity contracts," *Journal of financial economics*, vol. 3, no. 1-2, pp. 167–179, 1976.