Chapter 5

PINN-BO: A Black-Box Optimization Algorithm Using Physics-Informed Neural Networks

In scientific and engineering problems, many objective functions are governed by Partial Differential Equations (PDEs), which capture fundamental physical laws that describe how systems change over time and space. For example, the heat equation explains how heat spreads in a given area over time, while the Navier-Stokes equations model the movement of fluids, accounting for factors like viscosity and external forces. PDEs also play a central role in areas such as structural analysis and electromagnetics, highlighting their widespread importance across various fields.

Recently, researchers have started incorporating PDEs knowledge into models of objective functions. For instance, Raissi, Perdikaris, and Karniadakis (2017) proposed a Gaussian Processes Regression (GPR) approach that uses a four-block covariance kernel to combine observations from the objective function and PDEs. Similarly, Jidling et al. (2017) introduced a global constraint for Gaussian Processes (GPs) using differential equations, which eliminates the computational challenges of the four-block kernel while providing stronger constraints. Later, Chen et al. (2021) developed a method to solve non-linear PDEs by treating their solutions as Maximum a Posteriori (MAP) estimators of Gaussian processes. Although these methods are effective, GPs face scalability issues because updating their posterior involves inverting a kernel matrix, which has cubic complexity as the dataset grows.

To overcome these limitations, Physics-Informed Neural Networks (PINNs) have been introduced (Raissi, Perdikaris, and Karniadakis, 2019; Yang, Meng, and Karniadakis, 2021). PINNs use neural networks to capture complex, nonlinear relationships, making them a more flexible and scalable alternative to GPs. Unlike GPs, PINNs can handle a wide range of PDEs, including non-linear equations, and are well-suited for large-scale problems in science and engineering. Recent studies have also explored the theoretical aspects of PINNs, showing how they can effectively incorporate PDE constraints (Schiassi et al., 2021; Wang et al., 2022). Additionally, researchers have investigated connections between PINNs and GPs, demonstrating their ability to model functions that satisfy PDE constraints (Wang, Yu, and Perdikaris, 2022).

PDEs offer valuable information that can significantly improve the modeling of black-box objective functions, enabling more sample-efficient optimization by reducing the need for expensive evaluations. However, there has been limited exploration of how to effectively use PDEs in black-box optimization, where the objective function is often unknown and noisy.

This chapter presents PINN-BO, a black-box optimization framework that employs a PINN to model the unknown function, incorporates physical knowledge expressed by PDEs constraints into the optimization process. Hence, PINN-BO aims to enhance model accuracy and improve constraint handling, particularly in problems governed by established physical laws. We summary **the contributions of this chapter** as follows:

- We introduce a novel black-box optimization problem with physics information, described by Partial Differential Equations (PDEs), which is used to govern the objective function.
- We propose PINN-BO, a black-box optimization algorithm employing Physics-Informed Neural Network with Partial Differential Equations (PDEs) induced by natural laws to perform efficient optimization, bringing several benefits: improved sample-efficiency of optimization, scalable computation that only grows linearly with the number of function evaluations, and the ability to incorporate a broad class of PDEs.
- We provide a theoretical analysis of our proposed PINN-BO algorithm to illustrate that incorporating linear PDEs can lead to $\mathcal{O}\left(\sqrt{T\gamma_T}\sqrt{\gamma_T - I(f;\mathbf{Y}_T;\mathbf{U}_r)}\right)$ regret, where T is the number of black-box function evaluations and $I(f;\mathbf{Y}_T;\mathbf{U}_r)$ is interaction information between the black-box function f, its observations $\mathbf{Y}_T$ and the PDE data $\mathbf{U}_r$ (see Section 5.3).
- We perform experiments with a variety of tasks showing that our algorithm outperforms current state-of-the-art black-box optimization methods.

This chapter begins by introducing the problem setting and give an example to illustrate our proposed setting, followed by a detailed description of the PINN-BO framework and its theoretical analysis.

5.1 Problem Setting

In this chapter, we consider a global optimization problem setting where the objective function $f\colon \mathcal{D} \to \mathbb{R}$ is associated with a PDE:

$$\min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) \text{ s.t. } \mathcal{N}[f](\mathbf{x}) = g(\mathbf{x}),$$

where $\mathcal{D} \subset \mathbb{R}^d$ is a d-dimensional bounded domain and $\mathcal{N}[f]$ denotes a differential operator of the function f with respect to the input $\mathbf{x}$. The function f is an expensive, black-box function, and its evaluations are obtainable only through noisy measurements in the form of $y = f(\mathbf{x}) + \epsilon$, where ϵ represents sub-Gaussian noise. Additionally, the function $g(\mathbf{x})$ is a cheap-to-evaluate function, which may also involve noise, with respect to the PDE-constraint.

Remark 5.1.1. Considering the damped harmonic oscillator from classical physics as a real-world example where $f(x)$ is the displacement of the oscillator as a black-box function of time x, and measuring the displacement of a damped harmonic oscillator accurately can be expensive due to several factors, e.g., instrumentation cost, environmental factors, and calibration requirements. This displacement is governed by the PDE: $m\frac{d^2f}{dx^2} + c\frac{df}{dx} + kf = 0$, where $\mathcal{N}[f](x) = m\frac{d^2f}{dx^2} + c\frac{df}{dx} + kf$, $g(x) = 0$, m, c, k are mass, damping and spring values, respectively. Furthermore, it is assumed that the boundary conditions of the PDEs are either unknown or inaccessible.

These assumptions widely hold in many problem settings. As an example, Cai et al. (2020) examines a two-dimensional heat transfer problem with forced heat convection around a circular cylinder. The heat measurement entails high costs due to the material, size, and shape of the system. The problem has known incompressible Navier-Stokes and heat transfer equations. However, the thermal boundary conditions are difficult to ascertain precisely because of the complex and large instruments. The unavailability of these boundary conditions prevents the straightforward solution of the underlying function f using traditional numerical techniques.

5.2 Proposed PINN-BO Method

In this section, we present our proposed Physics-Informed Neural Network based Black-box Optimization (PINN-BO). PINN-BO algorithm combines optimization and machine learning techniques to efficiently optimize an unknown black-box function over a given input space while leveraging physics-informed constraints described by a PDE. Following the fundamental principles of Bayesian optimization, our algorithm consists of two primary steps: (1) constructing a model of the black-box objective function, and (2) employing this model to select the next function evaluation point during each iteration. In the first step, our approach diverges from traditional Bayesian Optimization algorithms that typically utilize Gaussian Processes (GPs) to model the objective function. Instead, we employ a fully connected neural network denoted as $h(\mathbf{x};\boldsymbol{\theta})$ to learn the function f as follows:

$$h(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{\sqrt{m}}\mathbf{W}_L\psi(\mathbf{W}_{L-1}\psi(\cdots\psi(\mathbf{W}_1\mathbf{x})),$$

where $\psi\colon \mathbb{R}\to\mathbb{R}$ is a coordinate-wise smooth activation function (e.g., ReLU, Tanh), $\mathbf{W}_1\in\mathbb{R}^{m\times d}, \mathbf{W}_i\in\mathbb{R}^{m\times m}, 2\le i\le L-1, \mathbf{W}_L\in\mathbb{R}^{1\times m}$, and $\boldsymbol{\theta}\in\mathbb{R}^p$ is the collection of parameters of the neural network, $p = md + m^2(L-2) + m$ and d is the dimension of inputs, i.e., $\mathbf{x}\in\mathcal{D}\subset\mathbb{R}^d$. We initialize all the weights to be independent and identically distributed as standard normal distribution $\mathcal{N}(0,1)$ random variables. To leverage the information embedded within the PDE governing the objective function f, our algorithm generates a set of N_r PDE data points denoted as $\mathcal{R} = \{\mathbf{z}_j, u_j\}_{j=1}^{N_r}$. Here, u_j represents the noisy evaluations of the function g at the corresponding point $\mathbf{z}_j$, where $u_j = g(\mathbf{z}_j) + \eta_j$. Besides, we denote $\mathcal{D}_t = \{\mathbf{x}_i, y_i\}_{i=1}^{t}$ as the set of noisy observations of the unknown function f after t optimization iterations, where $y_t = f(\mathbf{x}_t) + \epsilon_t$. We further define some other notations:

$$\phi(\cdot) = \nabla_{\boldsymbol{\theta}} h(\cdot;\boldsymbol{\theta}_0); \quad \omega(\cdot) = \nabla_{\boldsymbol{\theta}}\mathcal{N}[h](\cdot;\boldsymbol{\theta}_0)$$

where $\phi(\cdot)$ is the gradient of $h(\cdot;\boldsymbol{\theta}_0)$ with respect to the parameter $\boldsymbol{\theta}$, evaluated at initialization $\boldsymbol{\theta}_0$. Similarly, $\omega(\cdot)$ represents the gradient of $\mathcal{N}[h](\cdot;\boldsymbol{\theta}_0)$ with respect to model parameters $\boldsymbol{\theta}$ and evaluated at initialization $\boldsymbol{\theta}_0$, where $\mathcal{N}[h](\cdot;\boldsymbol{\theta}_0)$ is the result of applying differential operator $\mathcal{N}$ (with respect to the input) to $h(\cdot;\boldsymbol{\theta}_0)$. Both $\mathcal{D}_t$ and $\mathcal{R}$ play an important role in the subsequent stages of the algorithm, specifically in the minimization of the loss function associated with learning the network $h(\mathbf{x},\boldsymbol{\theta}_t)$ at optimization iteration t:

$$\mathcal{L}(t) = \textstyle\sum_{i=1}^{t-1}[y_i - \nu_t h(\mathbf{x}_i;\boldsymbol{\theta}_{t-1})]^2 + \sum_{j=1}^{N_r}[u_j - \nu_t\mathcal{N}[h](\mathbf{z}_j;\boldsymbol{\theta}_{t-1})]^2, \quad (5.1)$$

where ν_t is a scale parameter that controls the exploration-exploitation trade-off.

For the second step, we employ a greedy strategy to pick the next sample point $\mathbf{x}_t$. At each iteration t, the algorithm updates the neural network by optimizing the loss function described in Eqn 5.1 by gradient descent, with scaled function value predictions $\nu_t h(\cdot;\boldsymbol{\theta}_{t-1})$ and scaled predictions with respect to the governed PDE $\nu_t \mathcal{N}[h](\cdot;\boldsymbol{\theta}_{t-1})$. In the proof of Section 5.3, we show this action is equivalent to placing the GP prior over function values $f_{1:t}$ and PDE values $g_{1:N_r}$ (Corollary C.2.1.1 in the Appendix C). Then the posterior distribution of the function prediction $\widetilde{f}_t(\mathbf{x}) = h(\mathbf{x},\boldsymbol{\theta}_{t-1})$ at a new data point $\mathbf{x}$ can be viewed as being sampled from a GP with specific posterior mean and variance function (Lemma 5.3.4). This allows us to directly use the network prediction as an acquisition function following the principle of Thompson Sampling. Then, the next evaluation point $\mathbf{x}_t$ is selected by minimizing this acquisition function $\widetilde{f}_t(\mathbf{x}) = h(\mathbf{x},\boldsymbol{\theta}_{t-1})$. Then, the black-box function is queried at point $\mathbf{x}_t$, resulting in a (noisy) observation y_t, which is subsequently used to update the dataset $\mathcal{D}_t$. The PDE observations set $\mathcal{R}$, in combination with the observations in $\mathcal{D}_t$, is integrated into the training process of the neural network by minimizing the squared loss, as described in Eqn 5.1. We provide a concise step-by-step summary of our approach in Algorithm 4.

To enhance the exploration step, it is important to bring the additional information to improve our model of objective function, especially in the regions where optima lies. In our case, this task of exploration is easier as we have access to PDE which provides knowledge about the objective function and reduces the amount of information that is needed to model the function. In Section 5.3 (Theoretical Analysis), we derive a scaling factor $\nu_t = B + \widetilde{R}\sqrt{2\gamma_t - 2I(f;\mathbf{Y}_t;\mathbf{U}_r) + \log\left(\frac{1}{\delta}\right)}$, which directly reflects this intuition. It reduces the maximum information gain (which can be thought of as the complexity of the function modeling) by the interaction information $I(f;\mathbf{Y}_t;\mathbf{U}_r)$, which is a generalization of the mutual information for three variables: unknown function f, its observations $\mathbf{Y}_t$, and the PDE data $\mathbf{U}_r$. This information can be calculated as $I(f;\mathbf{Y}_t;\mathbf{U}_r) = \frac{1}{2}\log\left(\frac{\det\left(\frac{\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t}{\lambda_1}+\mathbf{I}\right)\det\left(\frac{\boldsymbol{\Omega}_r^\top\boldsymbol{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}{\det\left(\frac{\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t}{\lambda_1}+\frac{\boldsymbol{\Omega}_r^\top\boldsymbol{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}\right)$, where $\boldsymbol{\Phi}_t = [\phi(\mathbf{x}_1)^\top,\ldots,\phi(\mathbf{x}_t)^\top]^\top$ and $\boldsymbol{\Omega}_r = [\omega(\mathbf{z}_1)^\top,\ldots,\omega(\mathbf{z}_{N_r})^\top]^\top$. The value of ν_t signifies how our algorithm continues the exploration in regions of search space where the function f has no implicit knowledge through PDE observations. These are the regions indicated by $\gamma_t - I(f;\mathbf{Y}_t;\mathbf{U}_r)$, which is the amount of information about the unknown function f remains after our algorithm interacts with PDE data $\mathbf{U}_r$.

Algorithm 4 Physics-informed Neural Network based Black-box optimization (PINN-BO)

Input: The input space $\mathcal{D}$, the optimization budget T, PDE training set size N_r, $\delta \in (0,1)$, parameters $B, R_1, R_2, \lambda_1, \lambda_2$ (see Assumption 5.3.2 and 5.3.3 in Section 5.3).

1: Initialize $\mathcal{D}_0 = \emptyset$ and $\boldsymbol{\theta}_0 \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
2: Generate set $\mathcal{R} = \{\mathbf{z}_j, u_j\}_{j=1}^{N_r}$ from the PDE.
3: **for** $t = 1$ to T **do**
4: Set $\nu_t = B + \widetilde{R}\sqrt{2\gamma_t - 2I(f;\mathbf{Y}_t;\mathbf{U}_r) + \log\left(\frac{1}{\delta}\right)}$, where $\widetilde{R} = \sqrt{\left(\frac{R_1}{\lambda_1}\right)^2 + \left(\frac{R_2}{\lambda_2}\right)^2}$.
5: $\widetilde{f}_t(\mathbf{x}) = h(\mathbf{x};\boldsymbol{\theta}_{t-1})$
6: Choose $\mathbf{x}_t = \operatorname{argmin}_{\mathbf{x}\in\mathcal{D}} \widetilde{f}_t(\mathbf{x})$ and receive observation $y_t = f(\mathbf{x}_t) + \epsilon_t$
7: Update $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\mathbf{x}_t, y_t\}$
8: Update $\boldsymbol{\theta}_t = \operatorname{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ using Eqn. 5.1 by gradient descent with $\nu = \nu_t$.
9: **end for**

5.3 Theoretical Analysis

In this section, we provide a regret bound for the proposed PINN-BO algorithm. As presented in Section 2.3.6, to quantify the algorithm's regret, we employ the cumulative regret, defined as $R_T = \sum_{t=1}^{T} r_t$ after T iterations. Here, $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$ represents the optimal point of the unknown function f, and $r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t)$ denotes the instantaneous regret incurred at time t. Our regret analysis is built upon the recent NTK-based theoretical work of Wang, Yu, and Perdikaris (2022) and proof techniques of GP-TS Chowdhury and Gopalan (2017). Before proceeding with the theoretical analysis, we now introduce a set of definitions and assumptions. They clarify our proof and set up the basis and conditions for our analysis. **A detailed proof can be found in Section 5.3 of the Appendix.**

Definition 5.3.1. We define matrix $\mathbf{K}_{\text{NTK}-\text{PINN}}$ as the *neural tangent kernel of a Physics-Informed Neural Network (NTK of PINNs)*:

$$\mathbf{K}_{\text{NTK}-\text{PINN}} = \begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{ur} \\ \mathbf{K}_{ru} & \mathbf{K}_{rr} \end{bmatrix}, \tag{5.2}$$

where $(\mathbf{K}_{uu})_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, $(\mathbf{K}_{ur})_{ij} = \langle \phi(\mathbf{x}_i), \omega(\mathbf{z}_j) \rangle$, $(\mathbf{K}_{rr})_{ij} = \langle \omega(\mathbf{z}_i), \omega(\mathbf{z}_j) \rangle$ and $\mathbf{K}_{ru} = \mathbf{K}_{ur}^{\top}$, defined using $\mathcal{D}_t$ and $\mathcal{R}$.

Assumption 5.3.2. We assume the noises $\{\epsilon_i\}_{i=1}^{T}$ where $\epsilon_i = y_i - f(\mathbf{x}_i)$ and $\{\eta_j\}_{j=1}^{N_r}$ where $\eta_j = u_j - g(\mathbf{z}_j)$ are conditionally sub-Gaussian with parameter $R_1 > 0$ and $R_2 > 0$, where $\{\epsilon_i\}_{i=1}^{T}$ and $\{\eta_j\}_{j=1}^{N_r}$ is assumed to capture the noises induced by querying the black-box, expensive function $f(\cdot)$ and cheap-to-evaluate PDE-related function $g(\cdot)$ respectively.

$$\forall i \geq 0, \forall \lambda_1 \in \mathbb{R},\ \mathbb{E}[e^{\lambda_1 \epsilon_i} | \mathcal{F}_{t-1}] \leq e^{\frac{\lambda_1^2 R_1^2}{2}}$$
$$\forall j \geq 0, \forall \lambda_2 \in \mathbb{R},\ \mathbb{E}[e^{\lambda_2 \eta_j} | \mathcal{F}'_{N_r-1}] \leq e^{\frac{\lambda_2^2 R_2^2}{2}}$$

where $\mathcal{F}_{t-1}, \mathcal{F}'_{N_r-1}$ are the σ-algebra generated by the random variables $\{\mathbf{x}_i, \epsilon_i\}_{i=1}^{t-1} \cup \{\mathbf{x}_t\}$ and $\{\mathbf{z}_j, \eta_j\}_{j=1}^{N_r-1} \cup \{\mathbf{z}_{N_r}\}$, respectively.

Assumption 5.3.3. We assume f to be an element of the Reproducing Kernel Hilbert Space (RKHS) associated with real-valued functions defined on the set $\mathcal{D}$ (For more details about RKHS, see Section 2.1.4). This specific RKHS corresponds to the Neural Tangent Kernel (NTK) of a physics-informed neural network (NTK-PINN) and possesses a bounded norm denoted as $\|f\|_{\mathcal{H}_{k_{\text{NTK-PINN}}}} \leq B$. Formally, this RKHS is denoted as $\mathcal{H}_{k_{\text{NTK-PINN}}}(\mathcal{D})$, and is uniquely characterized by its kernel function $k_{\text{NTK-PINN}}(\cdot,\cdot)$. The RKHS induces an inner product $\langle \cdot, \cdot \rangle$ that obeys the reproducing property: $f(\mathbf{x}) = \langle f, k_{\text{NTK-PINN}}(\cdot, \mathbf{x}) \rangle$ for all $f \in \mathcal{H}_{k_{\text{NTK-PINN}}}(\mathcal{D})$. The norm induced within this RKHS, $\|f\|_{\mathcal{H}_{k_{\text{NTK-PINN}}}} = \sqrt{\langle f, f \rangle_{\mathcal{H}_{k_{\text{NTK-PINN}}}}}$, quantifies the smoothness of f concerning the kernel function $k_{\text{NTK-PINN}}$, and satisfies: $f \in \mathcal{H}_{k_{\text{NTK-PINN}}}(\mathcal{D})$ if and only if $\|f\|_{k_{\mathcal{H}_{\text{NTK-PINN}}}} < \infty$.

Assumptions 5.3.2 and 5.3.3 represent commonly employed and well-established assumptions in GP-based Bandits and Bayesian Optimization (Chowdhury and Gopalan, 2017; Vakili et al., 2021). We are now prepared to establish an upper bound on the regret incurred by our proposed PINN-BO algorithm.

We begin by presenting key lemmas for establishing the regret bound in Theorem 5.3.11 of the proposed algorithms. The following lemma demonstrates that, given the assumption of an infinitely wide network, the output of the trained physics-informed neural network used to model the unknown function f governed by a linear PDE, after running t optimization iterations in Algorithm 4, can be regarded as sampling from a GP with specific mean and covariance functions.

Lemma 5.3.4. *Conditioned on* $\mathcal{D}_t = \{\mathbf{x}_i, y_i\}_{i=1}^t, \mathcal{R} = \{\mathbf{z}_j, u_j\}_{j=1}^{N_r}$, *the acquisition function* $\widetilde{f}_t(\mathbf{x}) = h(\mathbf{x}; \boldsymbol{\theta}_{t-1})$ *can be viewed as a random draw from a* GP $\left(\mu_t^f(\mathbf{x}), \nu_t^2 \left(\sigma_t^f\right)^2(\mathbf{x})\right)$ *with the following mean and covariance functions:*

$$\mu_t^f(\mathbf{x}) = \phi(\mathbf{x})^\top \boldsymbol{\xi}_t^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{U}_r \end{bmatrix}$$
$$\left(\sigma_t^f\right)^2(\mathbf{x}) = \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle - \phi(\mathbf{x})^\top \boldsymbol{\xi}_t^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \boldsymbol{\xi}_t \phi(\mathbf{x}),$$

where

$$\widehat{\mathbf{K}}_{\text{PINN}}^{-1} = \begin{bmatrix} \mathbf{K}_{uu} + \lambda_1 \mathbf{I} & \mathbf{K}_{ur} \\ \mathbf{K}_{ru} & \mathbf{K}_{rr} + \lambda_2 \mathbf{I} \end{bmatrix}^{-1} = \begin{bmatrix} \widetilde{\mathbf{A}} & \widetilde{\mathbf{B}} \\ \widetilde{\mathbf{C}} & \widetilde{\mathbf{D}} \end{bmatrix}$$
$$\boldsymbol{\Phi}_t = [\phi(\mathbf{x}_1)^\top, \ldots, \phi(\mathbf{x}_t)^\top]^\top$$
$$\boldsymbol{\Omega}_r = [\omega(\mathbf{z}_1)^\top, \ldots, \omega(\mathbf{z}_{N_r})^\top]^\top, \ \boldsymbol{\xi}_t = \begin{bmatrix} \boldsymbol{\Phi}_t^\top & \boldsymbol{\Omega}_r^\top \end{bmatrix}^\top$$
$$\mathbf{K}_{uu} = \boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top, \ \mathbf{K}_{ur} = \boldsymbol{\Phi}_t \boldsymbol{\Omega}_r^\top, \mathbf{K}_{ru} = \mathbf{K}_{ur}^\top, \mathbf{K}_{rr} = \boldsymbol{\Omega}_r \boldsymbol{\Omega}_r^\top$$
$$\mathbf{Y}_t = [y_1, y_2, \ldots, y_t]^\top, \mathbf{U}_r = [u_1, u_2, \ldots, u_{N_r}]^\top$$

Generic BO methods (without the extra information from PDE) utilized the *maximum information gain* over search space $\mathcal{D}$ at time t: $\gamma_t := \max_{\mathcal{A} \subset \mathcal{D}: |\mathcal{A}| = t} I(\mathbf{Y}_\mathcal{A}, f_\mathcal{A})$, where $I(\mathbf{Y}_\mathcal{A}, f_\mathcal{A})$ denotes the mutual information between $f_\mathcal{A} = [f(\mathbf{x})]_{\mathbf{x} \in \mathcal{A}}$ and noisy observations $\mathbf{Y}_\mathcal{A}$, which quantifies the reduction in uncertainty about the objective function f after observing y_A. The maximum information gain is the fundamental component when analyzing regret bound for their algorithm (Srinivas et al., 2009; Vakili et al., 2021). However, in our work, the PDE evaluations $\{u_j\}_{j=1}^{N_r}$ of the function g is considered as the second source of information that contributes to reduce the uncertainty of f. Therefore, we introduce the *interaction information* as the generalization of *mutual information* for three random variables:

Definition 5.3.5. The **interaction information** between f, its observations $\mathbf{Y}_\mathcal{A}$, (where $\mathcal{A} \subset \mathcal{D}$), and the PDE data $\mathbf{U}_r$ can be defined as:

$$I(f; \mathbf{Y}_\mathcal{A}; \mathbf{U}_r) = I(f; \mathbf{Y}_\mathcal{A}) - I(f; \mathbf{Y}_\mathcal{A} | \mathbf{U}_r),$$

where $I(f; \mathbf{Y}_\mathcal{A})$ quantifies the reduction in uncertainty in f due to observing $\mathbf{Y}_\mathcal{A}$, while $I(f; \mathbf{Y}_\mathcal{A} | \mathbf{U}_r)$ represents the additional information contributed by $\mathbf{U}_r$ to enhance the mutual information between f and $\mathbf{Y}_\mathcal{A}$.

The next lemma provides the closed-form expression of the interaction information.

Lemma 5.3.6. *The interaction information between f and observation $\mathbf{Y}_t$ and PDE data $\mathbf{U}_r$, for the points chosen from Algorithm 4 can be calculated as:*

$$I(f;\mathbf{Y}_t;\mathbf{U}_r) = \frac{1}{2}\log\left(\frac{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)\det\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}\right)$$

Remark 5.3.7. Following Remark 3.3 in Wang, Yu, and Perdikaris (2022), both matrices $\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}$ and $\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}$ are positive semi-definite. It can be clearly seen that the interaction information given in Lemma 5.3.6 is non-negative:

$$\begin{aligned} I(f;\mathbf{Y}_t;\mathbf{U}_r) &= \frac{1}{2}\log\left(\frac{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)\det\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}\right) \\ &= \frac{1}{2}\log\left(\frac{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}+\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_1\lambda_2}\right)}{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}\right) \geq 0 \end{aligned}$$

The inequality uses the identity $\det(\mathbf{A}+\mathbf{B}) \geq \det(\mathbf{A})$, where $\mathbf{A},\mathbf{B}$ are two positive semi-definite matrices.

Our next result shows how the prediction of the neural network model is concentrated around the unknown reward function f, which is the key to a tighter regret bound.

Lemma 5.3.8. *Assume that $\|\omega(\cdot)\|_2 \leq L$, where $\omega(\cdot) = \nabla_{\boldsymbol{\theta}}\mathcal{N}[h](\cdot;\boldsymbol{\theta}_0)$ and $\rho_{min}(\mathbf{K}_{uu})$ the smallest eigenvalue of kernel matrix $\mathbf{K}_{uu}$ defined in lemma 1. Set $N_r = c_r\left(1+\frac{\rho_{min}(\mathbf{K}_{uu})}{\lambda_1}\right)/L^2$ for a positive constant c_r. Under the same hypotheses as stated in Assumption 5.3.2 and Assumption 5.3.3, and denote $\widetilde{R} = \sqrt{\left(\frac{R_1}{\lambda_1}\right)^2+\left(\frac{R_2}{\lambda_2}\right)^2}$. Let $\delta \in (0,1)$. Then, with probability at least $1-\delta$, the following confidence bound holds for all $\mathbf{x} \in \mathcal{D}$ and $t \geq 1$:*

$$\begin{aligned} &|f(\mathbf{x}) - \mu_t^f(\mathbf{x})| \\ &\leq \sigma_t^f(\mathbf{x})\left(B + \widetilde{R}\sqrt{2I(f;\mathbf{Y}_t) - 2I(f;\mathbf{Y}_t;\mathbf{U}_r) + \mathcal{O}(1) + \log(1/\delta)}\right) \\ &\leq \sigma_t^f(\mathbf{x})\left(B + \widetilde{R}\sqrt{2\gamma_t - 2I(f;\mathbf{Y}_t;\mathbf{U}_r) + \mathcal{O}(1) + \log(1/\delta)}\right) \end{aligned}$$

Proof sketch for Lemma 5.3.8 We split the problem into two terms: The prediction error of an element f in the RKHS as assumed in Assumption 5.3.3 with noise-free observations and the noise effect. Our proof differs from most GP-based Bayesian Optimization methods, which use single-block kernel matrices. In contrast, our predictive mean and covariance function involve the inversion of a block matrix $\widehat{\mathbf{K}}_{\text{PINN}}^{-1}$, as stated in Lemma 5.3.4. We employ the block matrix inversion formula (see Appendix A, (Rasmussen, Williams, et al., 2006)) to express $\widehat{\mathbf{K}}_{\text{PINN}}^{-1}$ as four distinct matrices. Subsequently, using equivalent transformations, intermediate matrix identities, and utilizing the expression *Interaction information* provided in Lemma 5.3.6, we derive the final bound.

Remark 5.3.9. The upper bound of the confidence interval presented in Lemma 5.3.8 shares a similar form with the existing confidence interval of GP-TS as outlined in

Chowdhury and Gopalan (2017). It is worth emphasizing, however, that our bound offers valuable insights into the significance of integrating PDEs to attain a tighter confidence bound. This insight can be summarized as follows: The expression $I(f;\mathbf{Y}_t) - I(f;\mathbf{Y}_t;\mathbf{U}_r)$ equals to $I(f;\mathbf{Y}_t|\mathbf{U}_r)$, which represents the expected mutual information between the function f and the observations $\mathbf{Y}_t$, given $\mathbf{U}_r$. Lemma 5.3.6 quantifies $I(f;\mathbf{Y}_t;\mathbf{U}_r)$ in terms of the kernel Gram matrices induced by black-box function and the PDE observations. As mentioned in Remark 5.3.7, the condition $I(f;\mathbf{Y}_t;\mathbf{U}_r) \geq 0$ implies that $I(f;\mathbf{Y}_t) \geq I(f;\mathbf{Y}_t|\mathbf{U}_r)$. This inequality signifies that knowing the values of the PDE component $\mathbf{U}_r$ reduces the statistical information between the observations $\mathbf{Y}_t$ and the unknown function f. In other words, knowing the values of PDE component $\mathbf{U}_r$ can diminish the number of observations $\mathbf{Y}_t$ required to estimate the unknown function f.

Remark 5.3.10. The value of $I(f;\mathbf{Y}_t;\mathbf{U}r)$ depends on the specific problem. For instance, if we assume that f is a function in RKHS with a linear kernel and $\mathcal{N}[f] = \sum_{i=1}^{n} \frac{\partial f}{\partial \mathbf{x}_i}$, the lower bound for the interaction information is:

$$I(f;\mathbf{Y}_t;\mathbf{U}_r) = \Theta\left(\frac{dN_r}{dN_r+1}(1-1/T)\right) = \Theta(1),$$

which is a constant. This is because the linear kernel differential feature map sends all PDE points to the same vector in the RKHS. This example aims to show how to bound the interaction information for a known PDE.

We are now ready to present the main theoretical result of the paper:

Theorem 5.3.11. *Let* $\mathbf{K}_{uu}, \mathbf{K}_{ur}, \mathbf{K}_{ru}$, *and* $\mathbf{K}_{rr}$ *be four matrices as defined in Lemma 5.3.4. Let* $\delta \in (0,1)$. *Assume that* $\|\omega(\cdot)\|_2 \leq L$ *and* $\rho_{min}(\mathbf{K}_{uu})$ *be the smallest eigenvalue of kernel matrix* $\mathbf{K}_{uu}$. *Set* $N_r = c_r\left(1 + \frac{\rho_{min}(\mathbf{K}_{uu})}{\lambda_1}\right)/L^2$ *for a constant* $c_r > 0$. *Additionally, let* $\widetilde{R} = \sqrt{\left(\frac{R_1}{\lambda_1}\right)^2 + \left(\frac{R_2}{\lambda_2}\right)^2}$ *and* $I_0 = \frac{1}{2}\log\frac{\det(\mathbf{K}_{rr}+\lambda_2\mathbf{I})}{\det\left(\mathbf{K}_{rr}+\lambda_2\mathbf{I}-\mathbf{K}_{ru}\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}\right)}$. *Then with probability at least* $1-\delta$, *the regret of PINN-BO running for an unknown function* f *governed by a linear PDE, lying in the* $\mathcal{H}_{k_{\text{NTK-PINN}}}$, $\|f\|_{H_{k_{\text{NTK-PINN}}}} \leq B$ *as stated in Assumption 5.3.3, after* T *iterations satisfies:*

$$R_T = \mathcal{O}\Bigg(\sqrt{Td\log BdT}\Bigg[B\sqrt{\gamma_T - I_0 + \log(2/\delta)} + \widetilde{R}\sqrt{\gamma_T}\sqrt{\gamma_T - I(f;\mathbf{Y}_T;\mathbf{U}_r) - I_0 + \log(2/\delta)}\Bigg]\Bigg)$$

5.4 Experiments

In this section, we demonstrate the effectiveness of our proposed PINN-BO algorithm through its application of synthetic benchmark optimization functions as well as real-world optimization problems. Our implementations of both problems are available at: https://github.com/phantrdat/pinn-bo.

5.4.1 Experimental Setup

For all experiments, we compared our algorithm with common classes of surrogate models used in black-box optimization, including Gaussian Processes (GPs)

and Deep Neural Networks (DNNs). For GPs, we employ the most popular strategy GP-EI (Mockus, Tiesis, and Zilinskas, 1978) and GP-UCB (Srinivas et al., 2009) with the Matérn Kernel. Our implementations for GP-based Bayesian Optimization baselines utilize public library GPyTorch `https://gpytorch.ai/` and BOTorch `https://botorch.org/`. We also include two recent DNNs-based works for black-box optimization: Neural Greedy (Paria et al., 2022) and Neural-BO (Phan-Trong, Tran-The, and Gupta, 2023) described below:

- Neural Greedy in Paria et al. (2022) fits a neural network to the current set of observations, where the function values are randomly perturbed before learning the neural network. The learned neural network is then used as the acquisition function to determine the next query point. Since the NeuralGreedy code is not publicly available, we use our own implementation following the setting described in Paria et al. (2022) (see Appendix F.2 therein).
- Neural-BO in Phan-Trong, Tran-The, and Gupta (2023) utilizes the Thompson Sampling strategy for selecting the next evaluation point. In this approach, the mean function is estimated using the output of a fully connected deep neural network. To implement this baseline, we adhere to the configuration outlined in Section 7 in Phan-Trong, Tran-The, and Gupta (2023).

For our proposed PINN-BO algorithm, we employ a fully connected deep neural network (DNN) as the surrogate model. The network's weights are initialized with independent samples drawn from a normal distribution $\mathcal{N}(0, 1)$. The model's hyper-parameters, which include depth, width, and learning rate, are selected as follows: For each function, we perform a grid search for tuning, where each hyper-parameter tuple is trained with 50 initial points. The width is explored within the set $\{100, 200, 500\}$, while the depth and the learning rate are searched across the values $\{2, 3, 4\}$ and $\{0.001, 0.005, 0.01, 0.02, 0.05, 0.1\}$, respectively. Subsequently, we select the tuple of (depth, width, learning rate) associated with the lowest mean-square error during evaluation. To train the surrogate neural network models, we utilize the (stochastic) gradient descent optimizer along with an Exponential Learning Rate scheduler with a factor of $\gamma = 0.95$. To accelerate the training process, we update the parameters $\boldsymbol{\theta}_t$ of surrogate models in Algorithm 4 after every 10 optimization iterations with 100 epochs.

5.4.2 Synthetic Benchmark Functions

We conducted optimization experiments on five synthetic functions: DropWave (2), Styblinski-Tang (10), Rastrigin (20), Michalewics (30), and Cosine Mixture (50), where the numbers in parentheses indicate the input dimensions of each function. We selected them to ensure a diverse range of difficulty levels, as suggested by the difficulty rankings available at `https://infinity77.net/global_optimization/test_functions.html`. To enhance the optimization process, we incorporated the PDEs associated with these objective functions. The detailed expressions of these functions and their corresponding PDEs can be found in Section C.1.1 of the Appendix C. Additionally, the noise in function evaluations follows a normal distribution with zero mean, and the variance is set to 1% of the function range. Results for Rastrigin, Michalewics, and Cosine Mixture functions are presented in Figure 5.1 and Figure . All experiments reported here are averaged over 10 runs, each with random initialization. All methods begin with the same initial points. The

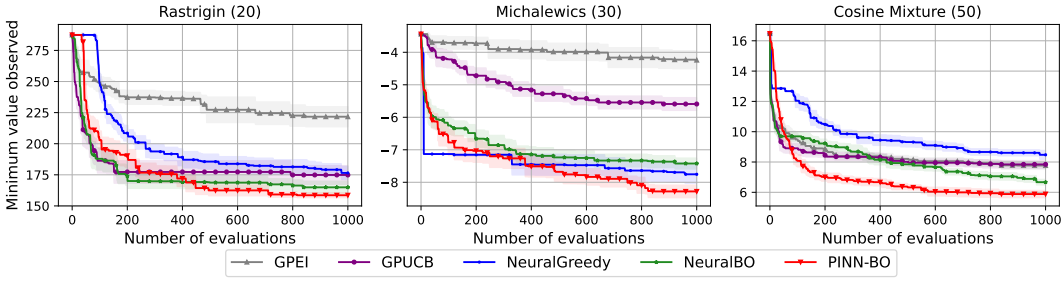


FIGURE 5.1: The optimization results for Rastrigin, Michalewics and Cosine Mixture functions comparing the proposed PINN-BO with the baselines. The standard errors are shown by color shading.

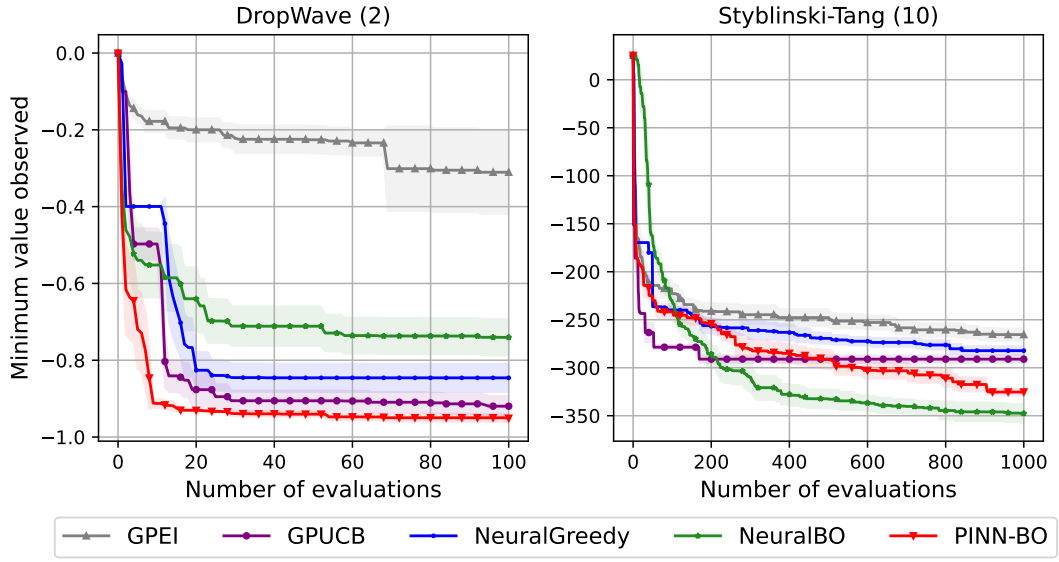


FIGURE 5.2: The optimization results for DropWave and Styblinski-Tang functions comparing the proposed PINN-BO with the baselines. The standard errors are shown by color shading.

results demonstrate that our PINN-BO is better than all other baseline methods, including GP-based BO algorithms (GP-EI, GP-UCB), and NN-based BO algorithms (NeuralBO, NeuralGreedy).

5.4.3 Real-world Applications

In this section, we explore two real-world applications where the objective functions are constrained by specific PDEs. We consider two tasks: (1) optimizing the Steady-State temperature distribution, satisfying the Laplace equation, and (2) optimizing the displacement of a beam element, adhering to the non-uniform Euler-Bernoulli equation. We continue to compare our proposed method with the baselines mentioned in Section 5.4.1.

5.4.3.1 Optimizing Steady-State Temperature

In this study, we showcase the benchmark optimization outcomes achieved by our proposed PINN-BO algorithm, comparing them with baseline methods for the

steady-state temperature optimization task. The steady-state heat equation represents a special case of the heat equation when the temperature distribution no longer changes over time. It describes the equilibrium state of a system where the temperature is constant, and no heat is being added or removed. The governing PDE for the temperature distribution is expressed as: $\nabla^2 T(x,y) = 0$, where x, y are spatial variables that represent the positions within a two-dimensional space. We explore the heat equation in a domain where x and y lie within the defined range of $[0, 2\pi]$. To thoroughly investigate the problem, we consider three different heat equations, where the solution of each problem is associated with one (unknown) boundary condition, each contributing to a deeper understanding of the system:

Heat Equation with Boundary Conditions 1

$$\begin{aligned} T(x,0) &= 5\sin(y) + \sqrt{1+y} \\ T(x,2\pi) &= y\sin\left(3\cos(y) + 2\exp(y)\sin(y)\right) \\ T(0,y) &= 10\cos(x) + x\exp\left(\sqrt{x^2+\sin(x)}\right) \\ T(2\pi,y) &= 3\sqrt{\exp(x\exp(-x))}\sin(x) + \cos(3x)\cos(3x) \end{aligned}$$

Heat Equation with Boundary Conditions 2

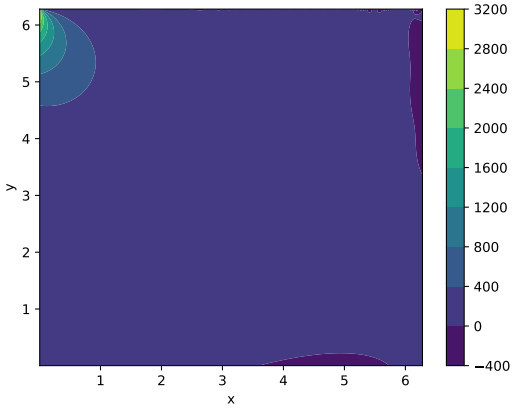
$$\begin{aligned} T(x,0) &= \sin(x)\cos(2x) + x^2\sqrt{3x} + e^{\sin(x)} \\ T(x,2\pi) &= e^{\sin(x)}\sqrt{3x} + x^2\cos(x)\sin^2(x) + e^{\cos(x)} \\ T(0,y) &= \sqrt{2y}\sin(y) + y^3\cos(2y) + e^{\cos(y)} \\ T(2\pi,y) &= \sin(y)\cos(2y) + y^3\sqrt{2y} + e^{\sin(y)} \end{aligned}$$

Heat Equation with Boundary Conditions 3

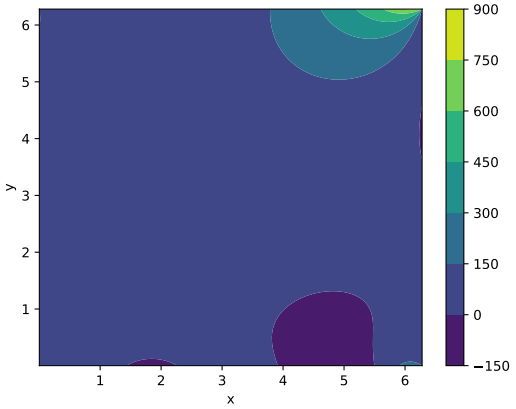
$$\begin{aligned} T(x,0) &= \left(\sin(x) + \cos(2x)\right)\sqrt{3x} + x^2 + e^{\sin(x)} \\ T(x,2\pi) &= \left(e^{\sin(x)} + \sqrt{3x}\right)\cos(x) + \left(\sin^2(x) + x^2\right)e^{\cos(x)} \\ T(0,y) &= \left(\sqrt{2y} + \sin(y)\right)\left(\cos(2y) + y^3\right) + e^{\cos(y)} \\ T(2\pi,y) &= \left(\sin(y) + \cos(2y)\right)\left(\sqrt{2y} + y^3\right) + e^{\sin(y)} \end{aligned}$$

We utilized `py-pde`, a Python package designed for solving partial differential equations (PDEs), available at the following GitHub repository: `https://github.com/zwicker-group/py-pde`. This tool enabled us to obtain solutions to heat equations at various input points, with specific boundary conditions serving as the input data. In Figure 5.3, the temperature distribution within the defined domain $[0, 2\pi]$ is visualized. It is important to emphasize that the boundary conditions used for benchmarking purposes are unknown to the methods employed. Adhering to the framework of black-box optimization, we assume that solving the PDEs incurs a substantial computational cost. The figure illustrates the spatial distribution of temperature values $T(x,y)$ across domain $[0, 2\pi] \times [0, 2\pi]$, with each subfigure corresponding to one of the mentioned boundary conditions.

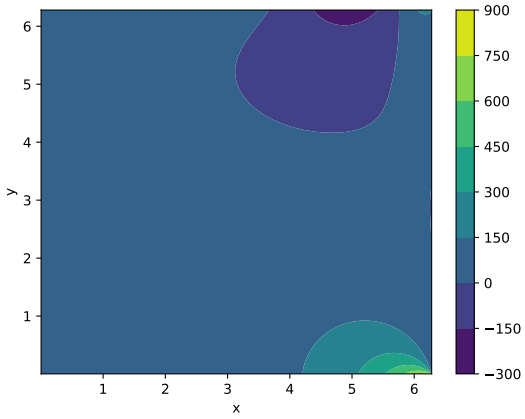
We conducted temperature optimization by identifying the locations (x,y) where the temperature reaches its maximum. As illustrated in Figure 5.3, the area with high



(A) Solution of Temperature Equation with boundary conditions 1



(B) Solution of Temperature Equation with boundary conditions 2



(C) Solution of Temperature Equation with boundary conditions 3

FIGURE 5.3: The figures depict the solutions for temperature distributions governed by the heat equation, with each figure corresponding to a specific tuple of boundary conditions described in Section 5.4.3.1. It is evident that the region with the highest temperature is relatively small in comparison to the entire domain.

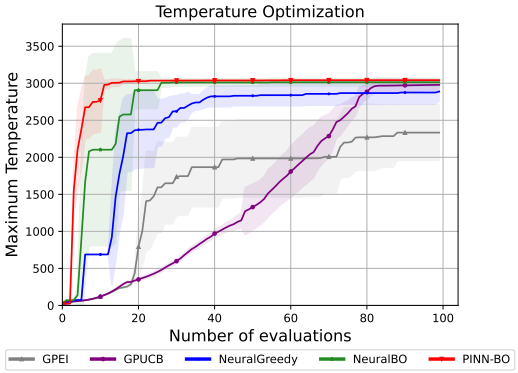
temperatures is relatively small in comparison to the regions with medium or low temperatures. For each baseline, we performed the optimization process 10 times, computing the average results. The comparative outcomes are presented in Figure 5.4.

5.4.3.2 Optimizing Beam Displacement

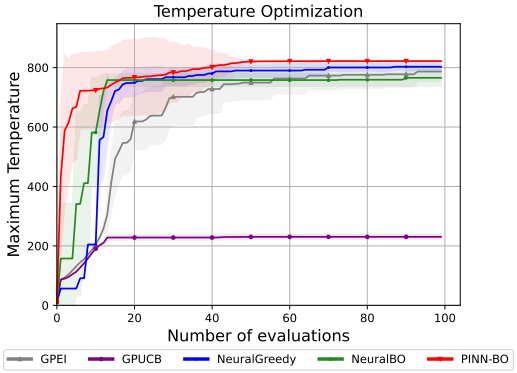
We present the benchmark optimization outcomes obtained through our proposed method, PINN-BO, and the baseline approaches, addressing the task of minimizing the deflection of a non-uniform Euler-Bernoulli beam. The governing differential equation describing the behavior of a non-uniform Euler-Bernoulli beam is provided below:

$$\frac{d^2}{dx^2}\left(EI(x)\frac{d^2w(x)}{dx^2}\right) = q(x),$$

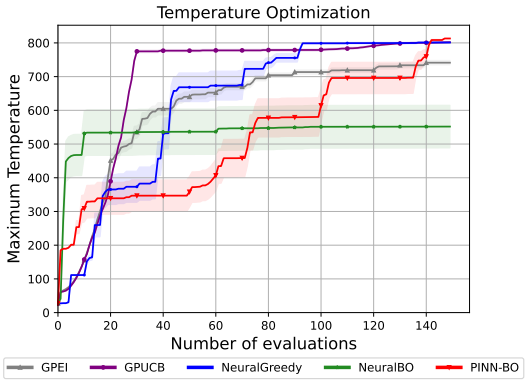
where $EI(x)$ represents the flexural rigidity of the beam, which can vary with position x, and $w(x)$ represents the vertical displacement of the beam at position x and



(A) Temperature Optimization in case of boundary conditions 1



(B) Temperature Optimization in case of boundary conditions 2



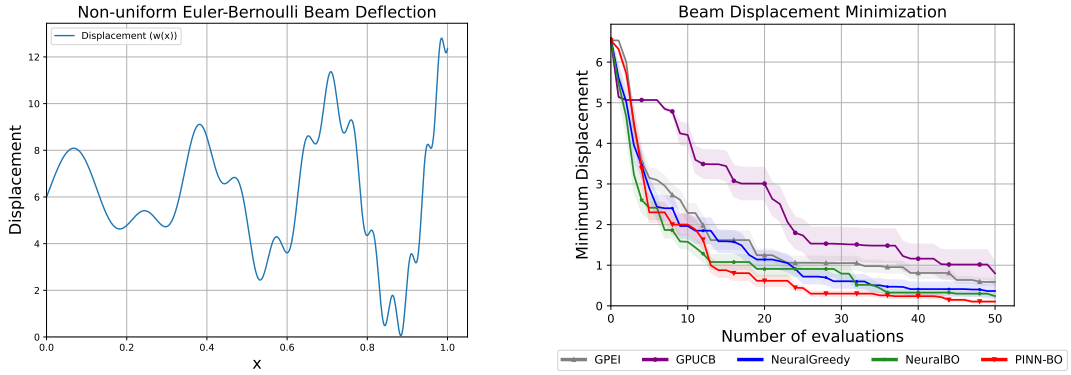
(C) Temperature Optimization in case of boundary conditions 3

FIGURE 5.4: The figure shows the temperature optimization results of our PINN-BO and other baselines. For all three cases with different positions of maximum temperature, our PINN-BO performs better than all other baselines.

$q(x)$ represents the distributed or concentrated load applied to the beam. In our implementation, we consider the detailed expression of $\mathrm{EI}(x)$ and $q(x)$ as follows:

$$\begin{aligned}\mathrm{EI}(x) &= \frac{e^x}{\rho(x)},\\ \rho(x) &= 2.4x - 64\pi^2 e^{4x}\sin\left(4\pi e^{2x}\right) - 396e^{2x}\sin\left(20x\right)\\ &\quad + 80e^{2x}\cos\left(20x\right) + 16\pi e^{2x}\cos\left(4\pi e^{2x}\right) + 0.4\end{aligned}$$

We employed the Finite Difference Method (FDM) to solve the non-uniform Euler-Bernoulli beam equation. It's crucial to note that this step is solely for generating observations at each input point. Despite obtaining this solution, our methods and all baseline techniques continue to treat this solution as a black-box function, accessing observations solely through querying. In Figure 5.5a, the displacement values $w(x)$ for $x \in (0,1)$ are illustrated. The optimization results for both our PINN-BO and the other baseline methods are presented in Figure 5.5b.



(A) Displacement of non-uniform Euler beam

(B) Displacement Optimization of non-uniform Euler beam

FIGURE 5.5: Displacement of a non-uniform Euler Beam and minimum displacement found by our PINN-BO and the other baselines. The left panel illustrates the natural displacement profile of the non-uniform Euler beam under given loads $q(x)$, flexural rigidity $EI(x)$, and boundary conditions. The right panel depicts the optimized position on the beam where the displacement is minimized, highlighting the location where the structural response is at its lowest.