Preparation Report: Physics-Enabled Learning of Surrogate Models for High-Performance Inkjet Printers

Haoyue Yang
1727095
System and Control
Supervisor: Prof.Amritam Das

Eindhoven, November 1, 2023

Contents

1 Introduction

1.1 The Combination of Physics and Data

The physical model of a system is built from its first principle and is mainly based on proven physical laws or mathematics equations, such as the Navier-stokes equation, Burgers' equation, Laws of thermodynamics, etc. It is widely used in engineering. However, the physical model usually isn't accurate because of some errors caused by the environment and usually has a few unknown parameters that need to be found to determine the value. Moreover, sometimes there's even a need to predict the future state of a system. Therefore, it's necessary to collect data from the real model and combine it with physics laws to build a more accurate and practical model and make state predictions for better control.

There are many data-driven approaches that help to identify parameters in a model and predict the future state based on analyzing current data, such as machine learning. By providing sufficient high-quality training data, the model identified from machine learning can achieve higher accuracy than the physical one. However, sometimes the learned models do not obey the physical laws, and the data can not always with good quality and quantity. In this small data regime, a large majority of machine-learning approaches lack robustness, and can't guarantee convergence. For many physical and biological systems, there exists much prior knowledge that is not utilized in classical machine-learning algorithms. Encoding that prior knowledge can help the algorithms amplify the information of data and find the right solution quickly, even in the case with a few training examples [17].

Therefore, it's a good way to combine the physics prior knowledge implicit in physics with the real data to obtain a model with better accuracy and generalization.

1.2 Physics-Enabled Learning

The physics-enabled learning technique represents a fusion of physics principles and data-driven methodologies, allowing the harnessing of the power of data analysis, machine learning, and computational methods to enhance the understanding of physical systems. This technique can improve the precision of measurements and predictions for model identification. It isn't limited to theoretical research, it improves the generalization of the identified model and has practical applications across industries [16].

The approach incorporates prior knowledge and exploits physical laws, principles, and constraints to inform or guide the data-driven models. The incorporation of prior knowledge improves the quality of prediction and reduces the required amount of data. It also improves data efficiency, since there's no need for large amounts of data, it uses smaller but more targeted datasets by exploiting the physical structure of the system. This is very useful in cases where data is hard, scarce, or expensive to obtain. The physics-enabled learning also has robustness by embedding physics into machine learning models, the predictions become more robust to noisy or incomplete data. The model from this approach is less likely to make predictions that violate fundamental physical principles.

The physics-enabled learning technique can also be applied cross-disciplinarily, it's highly adaptable and can be applied to a wide range of domains, including materials science, structural engineering, climate modeling, fluid dynamics, and many others. Overall, physics-enabled learning is a very powerful method that connects the physical sciences and machine learning, enabling more accurate, and efficient solutions to complex engineering and scientific problems.

Methods used in the approach aim to make the combination of the principles of physics with data-driven techniques enhance our understanding of physical systems and solve related problems. The common methods and techniques used in physics-enabled learning are as follows:

a. Physics-Informed Neural Networks (PINNs) [17]. PINNs use neural networks(NN) to solve related problems to PDEs, and the loss function includes the physical equations, boundary, and initial conditions. PINNs are commonly used for solving problems involving fluid dynamics, heat transfer, and structural mechanics.

b. Gaussian Processes (GPs) [2]. Gaussian process is a useful tool for modeling the uncertainty in physical systems. They can make predictions or do parametric estimations that incorporate both data and physics-based priors.

c. Reinforcement Learning (RL). RL can also be used in physics-enabled learning to optimize control policies for physical systems [11]. The approach can be a useful tool in finding optimal control strategies for robotic systems or autonomous vehicles.

Excluding those common methods introduced before, there are also some approaches that can work in physics-enabled models such as Proper Orthogonal Decomposition (POD), Deep Reinforcement Learning (DRL), and Sparse Bayesian Learning. The choice of method depends on the specific problem at hand and the nature of the physical system being studied. The aim of this research is to find parameters and make predictions, so the PINN and GP are chosen to be the main approaches to solve the problems.

1.3 Surrogate Model

A surrogate model is a simplified and computationally efficient model that approximates the behavior or output of a more complex, often computationally expensive, model or system. It's typically faster to evaluate than the original complex models which makes it useful in situations where the system is time-consuming and resource-intensive [7].

Surrogate models can be used for optimizing tasks, allowing you to explore the design space more efficiently. The model can be updated and evaluated to find the optimal parameters or settings [12]. They can help you understand the sensitivity of the system to different input parameters and perform sensitivity analyses on them to identify critical factors [10]. It can also be used for the estimation of uncertainty in the system's output, which is important in decision-making and risk assessment. Surrogate models can be easier to visualize, which helps understand the underlying system's behavior.

There are several kinds of methods of building surrogate models, such as machine learning and Proper Orthogonal Decomposition (POD). In this report, machine learning is the first choice and mainly uses Neural Networks (NN) and Gaussian Processes(GP) [19].

Surrogate models are used in various fields, including engineering, science, optimization, and more. They are particularly valuable in working with simulation models, finite element analysis, computational fluid dynamics, or other complex numerical simulations. They provide a practical way to make predictions of the system, optimize the performance of the model, and analyze systems while avoiding highly time-consuming simulations every time.

1.4 Application: Fixation Process in Inkjet Printing

The main process of inkjet printing can be considered as a physical integration of a solid medium and liquid material, such as paper and ink. The process has two separate stages, jetting and fixation. The liquid jets into the solid material in the jetting stages, and once a pattern is printed on the medium, the print product gets dried in the fixation stage. The fixation unit is depicted in Figure 3.

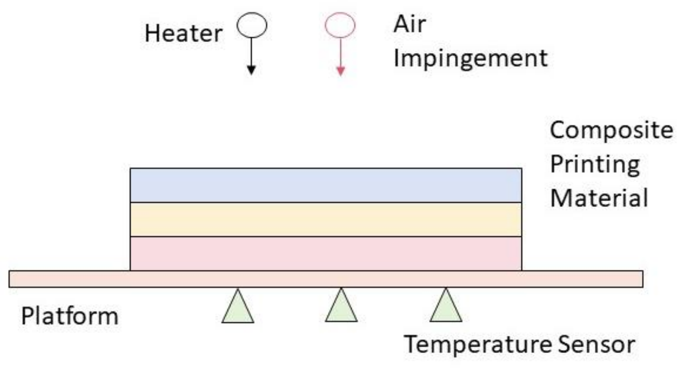


Figure 1: Overview of Fixation Unit

After jetting unit, the printed material is transported to the fixation unit on a solid platform and passes through the heater and air impingement unit to dry the material. The temperature and moisture sensors are distributed on the platform to monitor the material temperature.

An accurate simulation of the fixation process and predicting some states in the future can help identify the most efficient conditions, optimize the print process, ensure print quality, and reduce waste.

1.4.1 Physics Model of Fixation Process

The physics model is a combination of Partial Derivative Equation(PDE) and Ordinary Differential Equation(ODE).

1.4.2 Data from Fixation Process

The schematic graph of the fixation unit is in Figure 3. The unit has several temperature and moisture sensors that can monitor the temperature and moisture of the platform in a fixed position. The data is obtained from those sensors, it's changed through both time and space. To solve PDE, the boundary condition and initial condition are necessary, those conditions can also be obtained from the system in the state when $t = 0$ or $s = 0$.

1.5 Research Objective

Chapter 1.3 introduces the fixation process in a printer. To analyze and predict the dynamic process of the fixation unit, a simulation of the fixation process is needed. There are several parameters still unknown in the ideal model, needs to find a way to get the value of them to complete the model. After, with the simulation model, the temperature and moisture of the composite material can be monitored, and the future state can also be predicted. In this context, completing the model of the fixation process and the prediction of the future, it's a good application for the data-driven physics technique.

This thesis seeks an answer to the following question:

How to use machine learning to combine the physics and data to find the surrogate model of the fixation process?

1.6 Subproblems and Approaches to Solve Them

In order to solve the research objective, sets of subproblems related to it are formulated in this section. Every individual subproblem has a specific theoretical approach and implementation method. Then by combining all the solutions of the subproblems, the final goal of the research can be solved. In the subsequent paragraphs, the subproblems are formulated, and for each problem, the challenge, related theory, and assuming solutions are summarized.

The objective can be divided into two main subproblems:

1) Seek a method to find the value of unknown parameters in the physics model of the printer fixation process based on the collected data from the real setup.
2) Find a way to predict the change of temperature and moisture in the fixation process in a future state.

To solve those two problems step by step, some milestones can be listed as follows:

1) **Find A method to estimate parameters in the model of fixation processes**

   The solution to this problem is machine learning. Gaussian Process and neural network are both branches of machine learning and can both probably solve the problems.

   Raissi proposed a deep learning framework named Physics-informed neural networks(PINN) which can help achieve data-driven discovery tasks and obtain the value of parameters in partial differential equations(PDEs) [17]. The implementation process generally has these steps: formulate the physics-based model, collect data, set up the NN network, define the loss function, and train the network. These steps generate some subproblems:

2) **Build physics model and the datasets of the fixation process**

   The physics model with several unknown parameters is already known, and the data can be collected from the real setup by the sensors.

3) **Build a parametric estimation model based on PINN.**

   To overcome the limitations and improve the performance of PINN, some researchers tried to use a variety of network architectures. Zhiwei Fang presents a Hybrid PINN network for PDEs using a convolutional neural network (CNN) [6], and Lahariya proposed a physics-informed neural network to identify energy buffers that use a recurrent neural network(RNN) [13]. There's also an architecture of NN called ResNet, short for "Residual Network" [8], which has made a significant impact in the field of deep learning. The ResNet will probably also optimize the performance of our model. By comparing the accuracy of results from different networks, the optimal structure will be found.

4) **Optimize the model**

   There are three main directions of optimization: the structure of the network, the loss function, and parameter tuning.

   a. Network architectures

      To overcome the limitations and improve the performance of PINN, some researchers tried to use a variety of network architectures. Zhiwei Fang presents a Hybrid PINN network for PDEs using a convolutional neural network (CNN) [6], and Lahariya proposed a physics-informed neural network to identify energy buffers that use a recurrent neural network(RNN) [13]. There's also an architecture of NN called ResNet, short for "Residual Network" [8], which has made a significant impact in the field of deep learning. The ResNet will probably also optimize the performance of our model. By comparing the accuracy of results from different networks, the optimal structure will be found.

   b. Loss function

      In PINN, the loss function is the summation of several losses. First of all, in the preliminary work, a loss function with parameters is designed in the network and shows different parameters indeed influence the final results, so it's a good question that find proper parameters to optimize the network performance.

      (The form of the loss function is similar to multi-task learning [18]. Therefore, the weighting loss can probably also be used in the PINN network. As for the parameters in the loss function, here are several ways to optimize. Zhao Chen proposed a method named GradNorm to control update weight automatically [4], and Shikun Liu proposed a way to adjust the weight of loss function dynamically [14].)

   c. Hyper-parameter tuning

      Tuning hyper-parameters in Physics-Informed Neural Networks (PINNs) is an important step to optimize the performance of the model and achieve more accurate results. There are several hyper-parameters listed in the following that can be tuned in the algorithms.

      First of all, the learning rate determines the step size during gradient descent. The optimal value of learning rates depends on the specific problem. Experimenting with different learning rates is a good way to find the one that converges efficiently without overshooting [21].

      The architecture of the neural network also influences the performance of the network, including the number of hidden layers and neurons. The optimal architecture can vary greatly depending on the problem.

      The number of training epochs required for convergence also needs to be determined. This can vary based on the complexity of the problem and the effectiveness of other hyper-parameters.

A good way is to monitor training and validation loss to decide when to stop training without overfitting [15].

5) **Prediction of the model**

The method to obtain the prediction of the model is still machine learning, which has the ability to learn patterns and relationships from data. The process to achieve this goal is also not that different. Generate the dataset and use the network to learn the characteristics of the model, once the network finds the best way to describe the system, it's easy to speculate the next status.

To achieve this sub-problem, the most important thing is to obtain an accurate surrogate model that relies on the previous sub-problems. Designing a prediction network is the second task. Then optimize the network by tuning the parameters, update the network structures, and validate the performance of the prediction network.

2 Physics Data-driven Approach

Machine learning is always a good approach to solving data-driven tasks and can also work in the combination of physics and data. It's a powerful tool for data-driven physics, that harnesses the full potential of data, discovers patterns and relationships, and makes predictions and decisions. Neural networks (NN) and Gaussian processes (GP) are both branches of machine learning that might help solve the research problems.

2.1 Phsical Informed Neural Network(PINN)

Physics-Informed Neural Networks, or PINN, is derived from a classic neural network. It's a powerful computational framework that combines neural networks with physical principles to solve complex engineering problems. It's particularly useful for solving partial differential equations (PDEs) and other physics-based problems. The network is often used to solve two main classes of problems: finding the solution and discovering the parameters of PDEs. Since the physics model of the fixation process is in the form of PDE, PINN is a good tool to solve related problems [17].

The core of PINN is integrating Neural Networks (NNs) as versatile function approximators, and capable of capturing intricate relationships within datasets. The most unique character is the ability to enforce the underlying physics directly into the learning process [20]. By incorporating physics constraints as additional loss terms during training, which must be learned simultaneously with the uncertain functions of the differential equation (DE) [1], PINNs ensure that the learned models not only fit the observed data but also obey the physical laws that define the problem. This capability can be exploited to find the solution to complex problems with limited data, thus PINN is exceptionally data-efficient.

The applications of PINNs are diverse and encompass fields such as engineering, computational physics, geosciences, and many others. They can solve problems that involve heat conduction [3], fluid dynamics [25], structural mechanics, parameter estimation [24], data assimilation [9], model prediction [22], and more. The versatility of PINNs is a testament to their adaptability across domains, making them a valuable tool for scientists and engineers seeking accurate and efficient solutions to complex physical systems.

2.2 Gaussian Process with Prior Knowledge from Physical Model

Gaussian Processes (GPs) have become a fundamental branch of machine learning. They offer a flexible framework for classification, regression, optimization, and uncertainty quantification, and it's a useful tools for solving related problems.

Gaussian Processes based on the theory of stochastic processes and statistics. Their fundamental characteristics include the ability to capture complex and nonlinear relationships between variables while providing a natural measure of uncertainty [23]. They offer a Bayesian approach to regression, allowing for the incorporation of prior information and the estimation of predictive distributions, and are widely used for regression and interpolation tasks. GPs are also used for classification tasks, making them versatile tools for supervised learning [23].

3 Preliminary Result

To validate that machine learning can indeed help with solving PDE problems, a simple PDE example is introduced in this section. The example is a heat conduction initial-boundary-value problem. The problem is finding $u(x,t)$ from a PDE in Equation 1

$$\alpha^2 u_{xx} = u_t \tag{1}$$

In Equation 1, has $0 < x < L$ and $t > 0$, boundary conditions $u(0,t) = 0$ and $u(L,t) = 0$ for $t > 0$, and initial condition $u(x,0) = f(x)$ for $0 \leq x \leq L$.

To validate the results of the algorithm, the unknown parameters in the example are substituted by specific values and functions: $\alpha^2 = \frac{2}{3}$, $u(0,t) = 0$, $u(1,t) = 0$, $u(x,0) = 4sin(2\pi x) + 4sin(7\pi x)$ and $L = 1$. Then an accurate solution can be calculated in a mathematical way. By comparing the mathematical solution and the estimated results, it's easy to validate the performance of the network. The solution is Equation 2

$$u(x,t) = 4e^{-\frac{2}{3}(2\pi)^2 t} \sin(2\pi x) + 4e^{-\frac{2}{3}(7\pi)^2 t} \sin(7\pi x) \tag{2}$$

3.1 Data-driven Solutions of PDE

The solution of Equation 1 can also be calculated using machine learning. The step of PINN to solve PDE is introduced in this section.

1) **Design Neural Network Architecture**

   The network helps us estimate the solution of PDE, typically consisting of an input layer, hidden layers, and an output layer. Since the example it's simple, the network doesn't need to be deep. It consists of an input layer, 2 hidden layers, and an output layer.

2) **Loss Function**

   The loss function is the core of PINN, it has two components: data loss and physics loss. The data loss is to measure the difference between the neural network's predictions and the target data. The common choice is a mean squared error (MSE). The physics loss usually uses the boundary and initial conditions as constraints.

3) **Generate Dataset**

   A dataset is generated before training. The dataset includes the spatial and temporal coordinated, sometimes it may also include noisy or sparse measurements in practical issues. In the current algorithms, a dataset is generated that includes 2000 different values of x and t between 0 and 1.

4) **Training**

   Use an optimization algorithm to minimize the combined loss (data loss + physics loss) by adjusting the neural network's weights. Adam is chosen in the current network. In the algorithm, the learning rate is 0.0001, the network iterates 20000 times.

5) **Results**

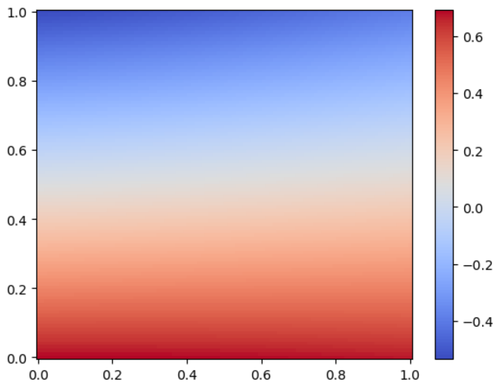
   Figure 2 shows the output of the network in figure form after different. And Figure 2f is generated by the Equation 2. By comparing Figure 2e and Figure 2f, the output of the network is similar to the ground truth which validates the performance of PINN in solution finding of PDEs.

3.2 Data-driven Discovery of PDE

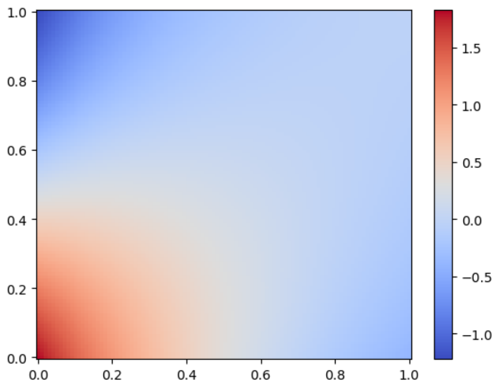
PINN can also solve the parameter estimation problems in PDEs. This section introduces two implementation methods for the data-driven discovery based on PINN.

3.2.1 PINN in "NeuroMANCER"

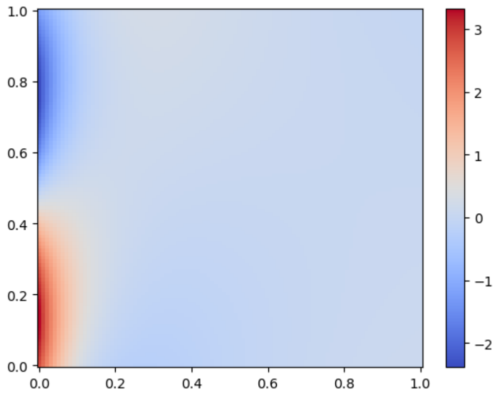
Neural Modules with Adaptive Nonlinear Constraints and Efficient Regularizations (NeuroMANCER) is a powerful open-source library developed for differentiable programming (DP), primarily targeting the solution of PDEs problems, such as physics-informed system identification, parametric constrained optimization problems or parametric model-based optimal control [5]. The library is implemented in PyTorch.



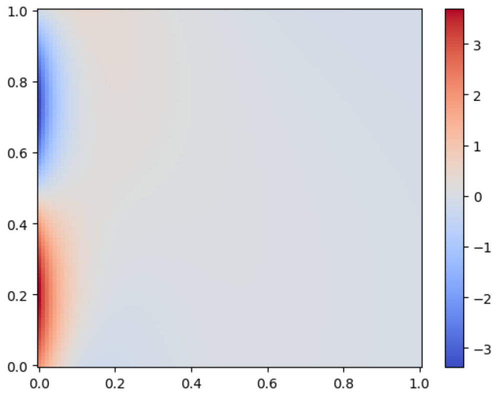
(a) Result of 500 iterations



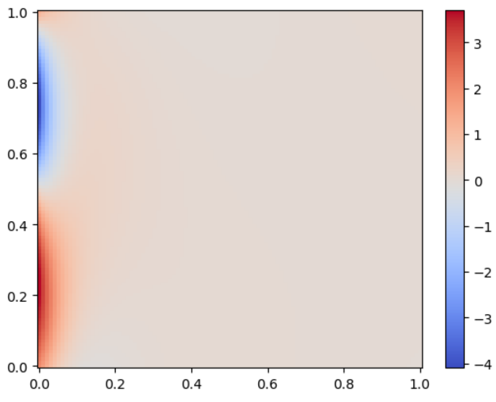
(b) Result of 1000 iterations



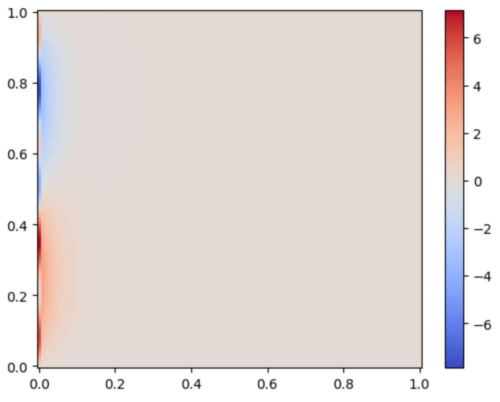
(c) Result of 5000 iterations



(d) Result of 10000 iterations



(e) Result of 20000 iterations



(f) Groundtruth

Figure 2: The comparison with results from the network of different iterations and the grountruth

The implementation of parametric estimation in PDEs based on ”NeuroMANCER” is introduced in this section.

1) **Construct Dataset**

   Generate 100 different values of x and t between 0 and 1, and calculate the corresponding outputs u refer to Equation 2. Then pick N_f collocation points randomly. In this example, $N_f = 1000$. The samples are visualized in

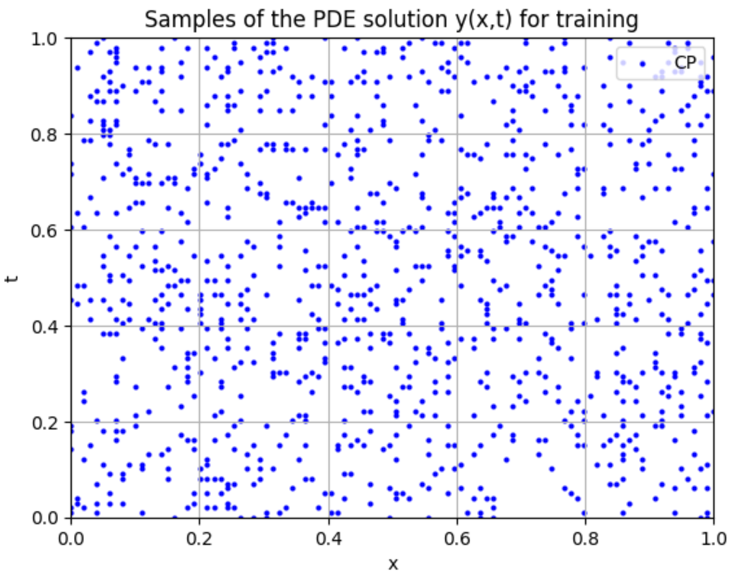


Figure 3: Samples of the PDE solution $u(x, t)$ for training

2) **PINN Architecture in ”NeuroMANCER”**

   In the algorithms, the solution of PDE is approximated as item 3.

$$\hat{u} = NN_\theta(x, t) \tag{3}$$

   This is for the situation when there is u in PDE. Since the example in this section is simple, the physics-informed layers can be directly defined as item 4. item 3.

$$f_{PINN} = \frac{\mathrm{d}u}{\mathrm{d}t} - \lambda \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} \tag{4}$$

   The derivatives can be obtained using Automatic Differentiation(AD). The symbolic variable of the ”NeuroMANCER” library can be exploited to simplify the implementation.

3) **Loss Function**

   In ”NeoroMANCER” algorithm, PINN’s loss function consists of three components: PDE collocation points loss, measurements loss, and bounded loss. The collocation points loss is the evaluation of the PINN over given numbers of collocation points as the following item 5.

$$\ell_1 = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| f_{\text{PINN}} \left(t^i, x^i \right) \right|^2 \tag{5}$$

   PDE measurement points $u(x, t)$ are selected, and supervised learning residual is minimized in the loss function item 6.

$$\ell_2 = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| u \left(t^i, x^i \right) - \hat{u} \left(t^i, x^i \right) \right|^2 \tag{6}$$

The output is expected to be bounded in the PDE solution domain $\hat{u} \in [-500.0, 500.0]$, thus the following inequality constraints via additional penalties is imposed as item 7.

$$\ell_3 = \frac{1}{N_f} \sum_{i=1}^{N_f} \left(\left| \text{RELU}\left(\hat{u}\left(t^i, x^i\right) - u_{\max}\right) \right|^2 + \left| \text{RELU}\left(-\hat{u}\left(t^i, x^i\right) + u_{\min}\right) \right|^2 \right) \tag{7}$$

Then the total loss is the summation of these three functions as item 8.

$$\ell_{PINN} = \ell_1 + \ell_2 + \ell_3 \tag{8}$$

4) **Training**

The "NeuroMANCER" has its own Training function named $Trainer()$ and can be directly used. The learning rate is 0.001 and the iteration can be decided depending on the situation.

5) **Results**

The result of the "NeuroMANCER" network in this example is not good. The loss of the algorithm is very large, and the results are inaccurate. The results of λ under different learning rates od shown in Table 1. The real λ should be 0.666667. So next step is to find a way to make the result more accurate, it will help achieve the objective of the research.

Table 1: The results of λ from PINN

Lr	0.1	0.01	0.001	0.0001
λ	0.017	0.1326	1.022	0.957

To ensure the influence of the learning rate, a different value of the learning rate is chosen and the loss change is described in Figure 4. From Figure 4, 4 different values of learning rate that are 0.1, 0.01, 0.001, and 0.0001 are chosen, the change of loss of the network is totally different. The red line with a 0.01 learning rate has the fastest convergence. The figure proves that the learning rate indeed influences the performance of the network. So next step in the future, more experiments will be taken to find the optimal learning rates.

3.2.2 PINN in Classical Network

There is some limitation in the "NeuroMANCER" algorithm, all the function needed in PINN is encapsulated and hard to modify or optimize. Debugging and analyzing are also very difficult. It's better to build a classical network work without all the encapsulated functions. The main process is similar to PINN in "NeuroMANCER". The only difference is the function and network are built without the "NeuroMANCER" library. It's an assumption and will be achieved in the research step.

1) **Construct Dataset**

This step has no difference with the first step in the last section. Generate 100 different values of x and t between 0 and 1, and calculate the corresponding outputs u refer to Equation 2.

2) **PINN Architecture**

The architecture is built based on PyTorch.

3) **Loss Function**

The loss function is the same as item 5 and item 6.

4) **Training**

The "NeuroMANCER" has its own Training function named $Trainer()$ and can be directly used. The learning rate is 0.001 and the iteration can be decided depending on the situation.

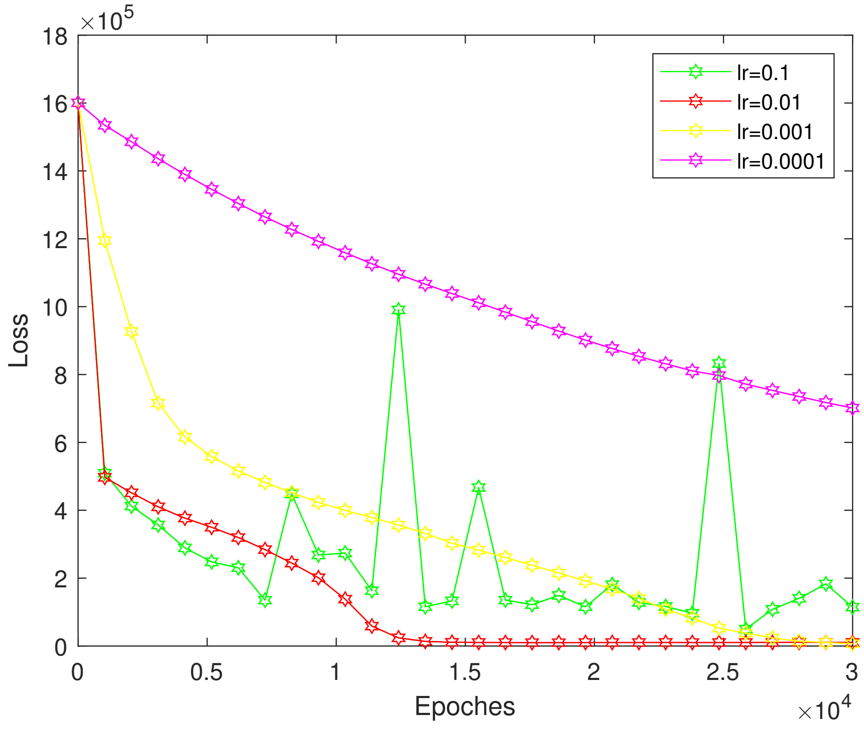


Figure 4: Loss change of different learning rate in PINN

5) **Results and Analysis**

The network hasn't been successfully built yet, it's just an assumption, but the principle and main process are the same as "NeuroMANCER". It will work when substituting the "NeuroMANCER" functions into our own functions.

4 Plan of the Graduation Thesis

4.1 Preparation Phase

August 15th to October 15th.

1) **September: Topic selection and proposal**

   a. Brainstorm potential research areas and topics of interest and Consider the scope and significance of each topic.

   b. Find literature related to the topic, and conduct a comprehensive literature review to research questions

   c. Narrow down topics select a topic for the thesis and confirm the research objective.

2) **October: Research and Preliminary work**

   a. Keep on the related literature searching and reading work to the PINN and GP for PDEs.

   b. Find a similar PDE example to the model of the fixation process but simpler to implement so that can be more familiar with those algorithms.

4.2 Research Phase

1) **November: Understand the physical model and start building the parametric estimation network**

   a. Start to work on the fixation model and consider the method to solve the subproblems mentioned in Chapter 1.
   b. Understand the physical model of the fixation process, and know the principle of it.
   c. Start to build the NN model for the parametric estimation problems

2) **December: Parametric estimation and optimization**
   a. Finish the network of parametric estimation and get a preliminary result.
   b. Validate the result and think about the problems of the network.
   c. Consider whether the network can be more optimal and get a better performance.

3) **January: Optimization and Prediction**
   a. If I have some ideas for optimization, I will modify the model and make the model have a better performance.
   b. If I have no idea how to make the model better currently, I will make the prediction based on the model with parameters already identified.

4) **February: Gaussian, optimization or other innovation**
   a. Try to use Gaussian Process to solve the same problems.
   b. Try to make some innovations in the NN network to make the performance better.
   c. Or do some other innovation work.

5) **March: Organize the code and thesis writing**
   a. Organize the current code of the model, and fix all the issues.
   b. Start writing the thesis.

6) **April: Thesis writing**
   a. Fix all the issues of the code.
   b. Write the thesis and prepare the defense.

References

[1] Ankita, Shalli Rani, Aman Singh, Dalia H Elkamchouchi, and Irene Delgado Noya. Lightweight hybrid deep learning architecture and model for security in iiot. *Applied Sciences*, 12(13):6442, 2022.

[2] Thomas Beckers. An introduction to gaussian process models. *arXiv preprint arXiv:2102.05497*, 2021.

[3] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

[4] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *CoRR*, abs/1711.02257, 2017.

[5] Jan Drgona, Aaron Tuor, James Koch, Madelyn Shapiro, and Draguna Vrabie. NeuroMANCER: Neural Modules with Adaptive Nonlinear Constraints and Efficient Regularizations. 2023.

[6] Zhiwei Fang. A high-efficient hybrid physics-informed neural networks based on convolutional neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5514–5526, 2022.

[7] Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide.* John Wiley & Sons, 2008.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] QiZhi He, David Barajas-Solano, Guzel Tartakovsky, and Alexandre M. Tartakovsky. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *CoRR*, abs/1912.02968, 2019.

[10] Bertrand Iooss and Paul Lemaître. A review on global sensitivity analysis methods. *Uncertainty management in simulation-optimization of complex systems: algorithms and applications*, pages 101–122, 2015.

[11] Andrei Ivanov, I Agapov, A Eichler, Sergey Tomin, et al. Physics-enhanced reinforcement learning for optimal control. In *12th International Particle Accelerator Conference*, number PUBDB-2021-05369. Beschleunigerphysik, 2021.

[12] Slawomir Koziel and Leifur Leifsson. *Surrogate-based modeling and optimization.* Springer, 2013.

[13] Manu Lahariya, Farzaneh Karami, Chris Develder, and Guillaume Crevecoeur. Physics-informed recurrent neural networks for the identification of a generic energy buffer system. In *2021 IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 1044–1049. IEEE, 2021.

[14] Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention. *CoRR*, abs/1803.10704, 2018.

[15] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 2002.

[16] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[17] Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.

[18] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[19] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

[20] Pushan Sharma, Wai Tong Chung, Bassem Akoush, and Matthias Ihme. A review of physics-informed machine learning in fluid mechanics. *Energies*, 16(5):2343, 2023.

[21] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[22] Jianxun Wang, Jinlong Wu, Julia Ling, Gianluca Iaccarino, and Heng Xiao. Physics-informed machine learning for predictive turbulence modeling: Towards a complete framework. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2016.

[23] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[24] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.

[25] Xiaoping Zhang, Yan Zhu, Jing Wang, Lili Ju, Yingzhi Qian, Ming Ye, and Jinzhong Yang. Gw-pinn: A deep learning algorithm for solving groundwater flow equations. *Advances in Water Resources*, 165:104243, 2022.