# CP-PINNs: Changepoints Detection in PDEs using Physics Informed Neural Networks with Total-Variation Penalty

**Zhikang Dong**[1*] **Paweł Polak**[1,2]
[1]Department of Applied Mathematics and Statistics
[2]Institute for Advanced Computational Science
Stony Brook University
Stony Brook, NY 11794
{zhikang.dong.1, pawel.polak}@stonybrook.edu

## Abstract

The paper shows that Physics-Informed Neural Networks (PINNs) can fail to estimate the correct Partial Differential Equations (PDEs) dynamics in cases of unknown changepoints in the parameters. To address this, we propose a new CP-PINNs model which integrates PINNs with Total-Variation penalty for accurate changepoints detection and PDEs discovery. In order to optimally combine the tasks of model fitting, PDEs discovery, and changepoints detection, we develop a new meta-learning algorithm that exploits batch learning to dynamically refines the optimization objective when moving over the consecutive batches of the data. Empirically, in case of changepoints in the dynamics, our approach demonstrates accurate parameter estimation and model alignment, and in case of no changepoints in the data, it converges numerically to the solution from the original PINNs model.

## 1 Introduction

Physics Informed Neural Networks (PINNs) represent the latest method that utilizes neural networks to approximate nonlinear functions in complex dynamical systems. They combine the adaptability of neural networks with the physics of Partial Differential Equations (PDEs) and data-driven techniques for inverse problems [1]. Unlike conventional numerical methods [2, 3, 4], PINNs incorporate physical laws as inherent prior knowledge. Their applicability is broad, spanning fields such as optics, cardiology, and power systems [5, 6, 7]. However, a significant challenge for PINNs is effectively handling changepoints in dynamic systems. Changepoints are critical moments where abrupt variations or discontinuities occur in the system's behavior, often reflecting underlying shifts in the system's dynamics or external influences [8, 9, 10, 11, 12]. These points are pivotal in inverse problems, where identifying such shifts of parameters can lead to more accurate models and predictions.

This research introduces a new combination of changepoints detection and PINNs, achieved by integrating a total variation penalty [13, 14] into the optimization objective. Our CP-PINNs model, based on established methods [15, 16], has two main functions. (1) It identifies changepoints in PDE parameters, crucial for accurate modeling of systems where parameters exhibit abrupt changes. (2) It refines PDE solutions by adaptively distributing weights in the loss function, a technique derived from meta-learning. This approach not only improves the accuracy of the model in stable conditions but also ensures robustness against instabilities caused by rapid variations in parameters.

*Corresponding Author.

Machine Learning and the Physical Sciences Workshop, NeurIPS 2023.

The introduced CP-PINNs framework offers a broad solution for monitoring dynamical systems governed by specific PDEs, even when parameters are unknown. Its potential applications span across various domains: detecting leakages in pipelines using limited sensor data [17]; traffic flow management by predicting congestion without comprehensive sensor coverage [18]; environmental monitoring for sudden pollutant concentration shifts [19]; ensuring energy grid stability by pinpointing fluctuations or failures [20, 21]; overseeing heat distribution in manufacturing materials [22]; medical imaging to detect tissue property changes [23]; seismic activity monitoring for early earthquake detection [24]; aerospace component stress assessment for safety [25]; agriculture water and nutrient distribution tracking [26, 27]; atmospheric change detection in weather systems [28]; and ensuring product consistency in manufacturing [29]. Section 2 introduces the general problem formulation, Section 3 showcases the effectiveness of CP-PINNs through simulations and Section 4 concludes.

## 2 Model and Estimation

Following [1], in order to approximate the true solution of the PDEs, we use the neural network approximation $u_{NN}(\mathbf{x}, t; \mathbf{\Theta})$ given by $u_{NN}(\mathbf{x}, t; \mathbf{\Theta}) = g \circ T^{(\ell)} \circ T^{(\ell-1)} \circ \cdots \circ T^{(1)}(\mathbf{x})$. For each hidden layer $i = 1, \ldots \ell$, the nonlinear operator $T^{(i)}$ is defined as $T^{(i)}(\mathbf{x}) = \sigma(\mathbf{W}_i \mathbf{x} + \mathbf{b}_i)$ with weights $\mathbf{W}_i \in \mathbb{R}^{\mathcal{M}_i \times \mathcal{M}_{i-1}}$ and biases $\mathbf{b}_i \in \mathbb{R}^{\mathcal{M}_i}$, where $\mathcal{M}_0 = d$ is the input dimension and in the output layer, the operator $g : \mathbb{R}^{M_\ell} \to \mathbb{R}$ is a linear activation function. Next, define a feature function $f(\mathbf{x}, t) = u_t + \mathcal{N}[u; \lambda(t)]$, where $\mathcal{N}[\cdot; \lambda(t)]$ is a nonlinear operator parameterized by $\lambda(t)$, and $u$ is the latent solution. The following is the definition of changepoints in our case.

**Definition 2.1.** *The function $\lambda(t) : [0, T] \to \mathbb{R}$ is piecewise-constant, bounded, with a finite but unknown number of discontinuities $k^*$, located at some of the observations, i.e., $\lambda(t) = \sum_{i=1}^{k^*} \lambda_i \mathbf{1}_{[t_{i-1}, t_i)}(t)$, for $0 = t_0 < t_1 < \ldots < t_{k^*} < T$.*

We write the corresponding boundary and initial residual functions

$$r_b(u_{NN}, \mathbf{x}, t) = u_{NN} - u, \qquad \forall (\mathbf{x}, t) \in \partial\Omega \times (0, T]$$
$$r_0(u_{NN}, \mathbf{x}, t) = u_{NN} - u_0, \quad \forall (\mathbf{x}, t) \in \Omega \times \{t = 0\}.$$

Averaging over observations on their domains, we obtain the fitting loss function for the training of the model

$$L^{\mathfrak{f}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left| r_b\left(\mathbf{x}_b^i, t_b^i\right) \right|^2 + \frac{1}{N_0} \sum_{i=1}^{N_0} \left| r_0\left(\mathbf{x}_0^i\right) \right|^2 \text{ and } L^{\mathfrak{s}} = \frac{1}{N} \sum_{i=1}^{N} \left| f_{NN}(\mathbf{x}^i, t^i) \right|^2, \tag{1}$$

where $L^{\mathfrak{f}}$ denotes the loss function for the observations from the boundary and initial conditions, and $L^{\mathfrak{s}}$ denotes the loss function for the observations inside the domain. By gathering together these two loss functions with equal weights, one obtains the total loss for the training of the original PINNs.

The standard PINNs model assumes that the parameters of PDEs are constant values across the entire time domain. We allow for the changes in the $\lambda(t)$ and introduce additional regularization term in a form of total variation penalty on the first difference in $\lambda(t)$, i.e., $V^\lambda = \sum_{i=1}^{T-1} \delta(t^i) \left| \Delta\lambda(t^i) \right|$, where $\Delta\lambda(t^i) = \lambda(t^{i+1}) - \lambda(t^i)$, and $\delta(t)$ is a U-shape linear function of $t$ which increases the penalty strength closer to the edges of time domain in order to avoid estimation instabilities—in the experiments below we use $\delta(t) = \sqrt{T/(T-t)}$. $V^\lambda$ is a sparsity inducing penalty similar to one used in [13] and [30] for changepoints detection in linear regression, and non-parametric statistics, respectively. We then perform standard PINNs to refine PDEs discovery and solve PDEs within each detected time interval.

In order to balance the three goals during the training process, we define our loss function as a weighted average of the loss terms

$$L(\mathbf{w}) = \left[ L^{\mathfrak{f}}, L^{\mathfrak{s}}, V^\lambda \right] \mathbf{w}, \tag{2}$$

where $\mathbf{w} = [w_1, w_2, w_3]^\top$, $w_i > 0$, for $i = 1, 2, 3$, and $\sum w_i = 1$, is the vector of weights towards corresponding loss terms.

We use the ReLU activation function in expressing $V^\lambda$ due to its extensive optimization and prevalent usage in deep learning frameworks such as PyTorch and TensorFlow. We have

$$V^\lambda = \sum_{i=1}^{T-1} \delta(t^i) \left[ \Delta\lambda^+(t^i) + \Delta\lambda^-(t^i) \right]. \tag{3}$$

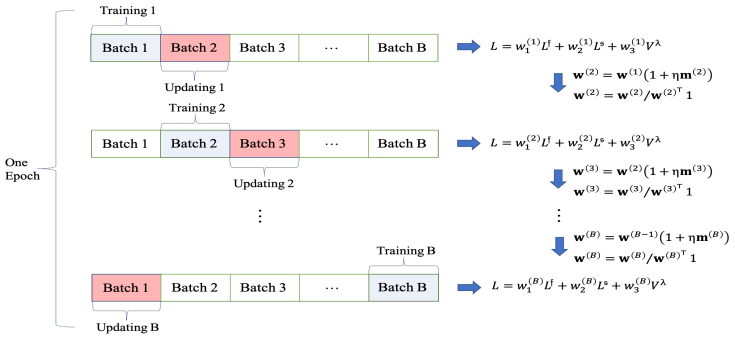


Figure 1: The dataset is partitioned based on spatial information, with each batch encompassing the full temporal information. In the meta-learning approach, the network is trained using the previous distribution of loss weights and updated based on the data from the subsequent batch.

$$\Delta\lambda^+(t^i) = \begin{cases} \Delta\lambda(t^i), & \text{for } \Delta\lambda(t^i) \geq 0, \\ 0, & \text{otherwise.} \end{cases} \qquad \Delta\lambda^-(t^i) = \begin{cases} 0, & \text{for } \Delta\lambda(t^i) \geq 0, \\ -\Delta\lambda(t^i), & \text{otherwise.} \end{cases}$$

In order to optimize the distribution of loss weights $\mathbf{w}$, we propose a cost function at the $k$-th batch as

$$\mathbf{m}^{(k)}\left(\mathbf{w}^{(k-1)}\right) = \left[\frac{L^{\mathfrak{f}}_{(k)}}{L^{\mathfrak{f}}_{(k)} + L^{\mathfrak{s}}_{(k)} + V^{\hat{\lambda}}_{(k-1)}}, \frac{L^{\mathfrak{s}}_{(k)}}{L^{\mathfrak{f}}_{(k)} + L^{\mathfrak{s}}_{(k)} + V^{\hat{\lambda}}_{(k-1)}}, \frac{V^{\hat{\lambda}}_{(k-1)}}{L^{\mathfrak{f}}_{(k)} + L^{\mathfrak{s}}_{(k)} + V^{\hat{\lambda}}_{(k-1)}}\right]^\top,$$

where $L^{\mathfrak{f}}_{(k)} = \frac{1}{B}\sum_{i=1}^{B}\left(u_i - u_{NN}(\mathbf{x}_i^{(k)}, t, \hat{\mathbf{\Theta}}^{(k-1)}(\mathbf{w}^{(k-1)}))\right)^2$, $L^{\mathfrak{s}}_{(k)} = \frac{1}{B}\sum_{i=1}^{B}\left|f_{NN}(\mathbf{x}_i^{(k)}, t)\right|^2$, $V^{\hat{\lambda}}_{(k-1)} = \sum_{i=1}^{T-1}\delta(i)\left[\Delta\hat{\lambda}^+_{(k-1)}(t^i) + \Delta\hat{\lambda}^-_{(k-1)}(t^i)\right]$. $\hat{\mathbf{\Theta}}^{(k-1)}$, $\mathbf{w}^{(k-1)}$ and $\hat{\lambda}_{(k-1)}$ are neural network parameters, loss weights vector and estimated physical parameter at $(k-1)$-th batch, respectively. $B$ is the number of batches.

Figure 1 shows our iterative algorithm for meta learning of the loss weights $\mathbf{w}$. For the $k$-th batch of data, we compute the cost function using data points uniformly sampled across the spatial domain and spanning the entire time domain. This approach adaptively refines the weight vector $\mathbf{w}$ based on out-of-sample data. Specifically, it increases the weight for components with larger losses and decreases it for those with smaller losses. This ensures that the neural network focuses more on regions it currently underfits, leading to a more balanced overall fit.

## 3 Empirical Results

We discuss the advection-diffusion equation, which is widely used in heat transfer [31], mass transfer [32], and fluid dynamics problems [33]. The equation of one-dimensional form is given by

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \lambda(t)\frac{\partial^2 u}{\partial x^2} \quad u(t,-1) = u(t,1) = 0 \quad u(x,0) = -\sin(\pi x),$$

where $\lambda(t) = 0.5$ for $t \in [0, 0.333)$, $0.05$ for $t \in [0.333, 0.667)$ and $1$ for $t \in [0.667, 1]$. $u(x,t) : \Omega \to \mathbb{R}$, and $\Omega = [-1,1] \times [0,1]$. Initially, we set $\mathbf{w}^{(0)} = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]^\top$ and $\eta = 10^{-5}$.

The left panel of Figure 2 presents the detected changepoints over the temporal axis. In comparison, the PINNs[2] method detects these breakpoints at 0.240s and 0.615s and estimates parameters as 0.498, -0.082 and 9.018 respectively, whereas the CP-PINNs method accurately detects them at 0.335s and 0.671s and estimates parameters as 0.5001, 0.048 and 0.999 respectively. The right panel of Figure 2 reveals an upward trend in $w_2$ and a declining trajectory for $w_1$ and $w_3$. This dynamic results from the

[2]Standard PINNs assume a constant coefficient over time, which performs much worse for changepoint scenarios. To ensure a fair comparison, we modify PINNs to allow for a time-dependent coefficient. This highlights the effectiveness of our total variation term combined with meta-learning.

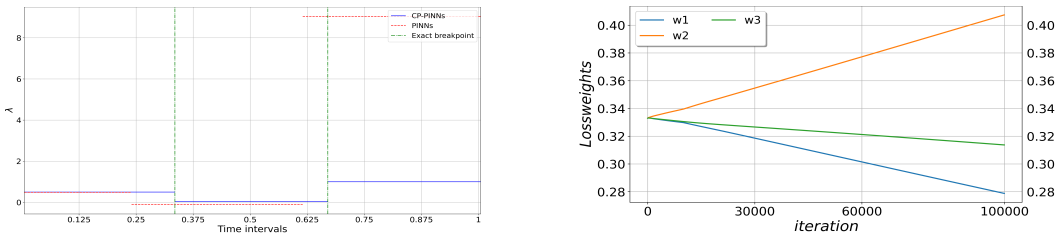


Figure 2: (Left) Parameter estimation of CP-PINNs and PINNs. (Right) Loss weights distribution across time.

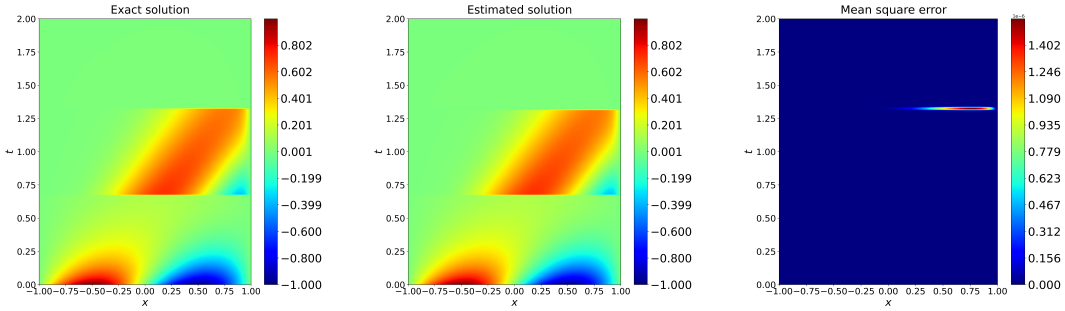


Figure 3: The solution of 1D advection-equation with two breakpoints.

interplay between the total variation term assisting the model in discerning the breakpoints' positions and the meta-learning algorithm gradually directing the model's focus toward efficiently fitting the PDEs. Figure 3 shows the performance of CP-PINNs in solving PDEs, especially in proximity to edges and breakpoints.

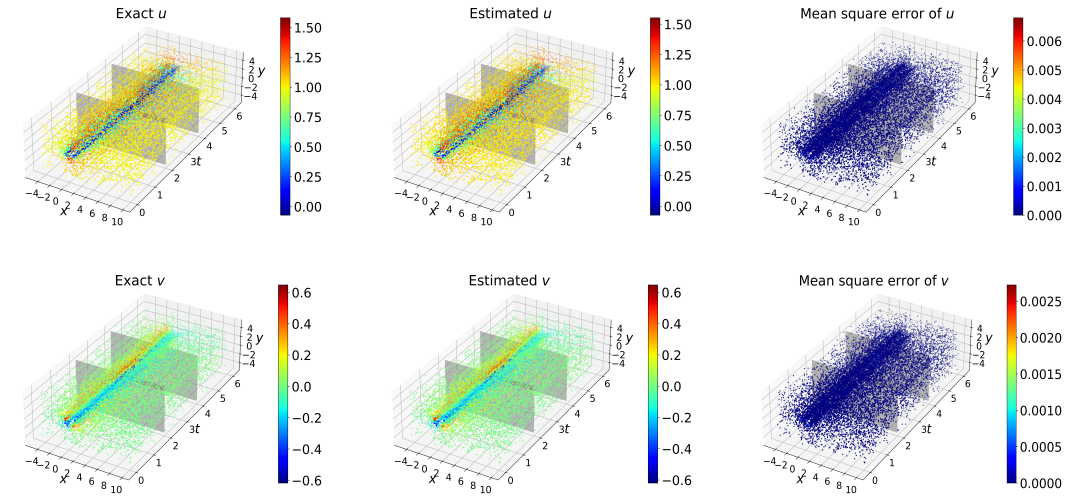


Figure 4: The stream-wise and transverse velocity of 2D Navier-Stokes equation with two breakpoints.

We next explore a scenario involving the Navier–Stokes equation, a standard model for incompressible fluid flow, often applied in contexts such as water flow in pipeline [34], air dynamics over aircraft surfaces [35], and pollutant dispersion [36]. Here, we focus on a specialized 2D case of the equation.

$$u_t + (uu_x + vu_y) = -p_x + \lambda(t)\,(u_{xx} + u_{yy}) \quad v_t + (uv_x + vv_y) = -p_y + \lambda(t)\,(v_{xx} + v_{yy})\,,$$

for $u(x, y, t)$, $v(x, y, t)$ and $p(x, y, t) : \Omega \to \mathbb{R}$, where $\Omega = [-5, 10] \times [-5, 5] \times [0, 6]$. $u(x, y, t)$ and $v(x, y, t)$ denote the stream-wise and transverse velocity components, respectively, and $p(x, y, t)$ is

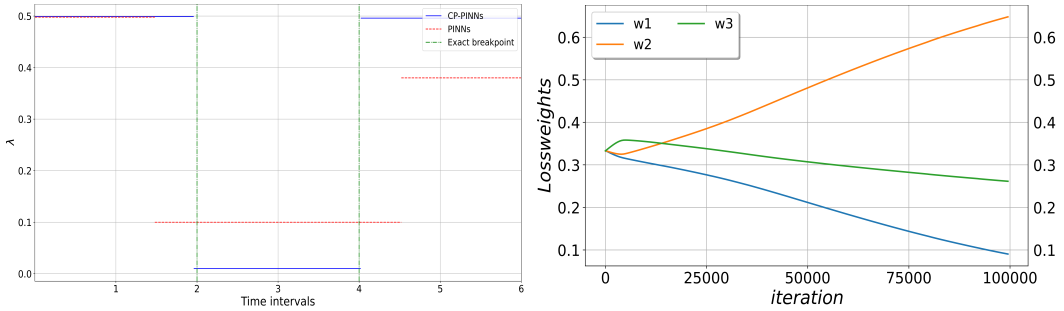


Figure 5: (Left) Parameter estimation of CP-PINNs and PINNs. (Right) Loss weights distribution across time.

the measurement of fluid pressure. Similar to [1], we assume a uniform free stream velocity at the left boundary and a zero pressure outflow condition at the right boundary. The top and bottom boundaries are treated as periodic, ensuring continuous flow in the vertical direction. In our experiment, we focus on incompressible fluid flow passing through the 2D circular cylinder with radius = 0.5, and $\lambda(t)$ is the viscosity coefficient. It's known that fluid viscosity is significantly affected by temperature changes; for instance, heating oil decreases its viscosity, facilitating flow, while cooling increases it. Recognizing this sensitivity, we model $\lambda(t)$ to vary over time: $\lambda(t) = 0.5$ for $t \in [0, 2)$, then $0.01$ for $t \in [2, 4)$, and returns to $0.5$ for $t \in [4, 6]$. This variation simulates the effect of rapid temperature changes on fluid viscosity.

Our method detects changepoints at 1.96s and 4.02s, offering precise parameter estimates for each segment: $\hat{\lambda}(t) = 0.499$ for $t \in [0, 1.96)$, $0.010$ for $t \in [1.96, 4.02)$, and $0.496$ for $t \in [4.02, 6]$. PINNs incorrectly estimate changepoints at 1.48s and 4.52s, yielding the erroneous parameters 0.498, 0.101, and 0.38, respectively. Figure 4 showcases our model's capability in solving both stream-wise and traverse velocities. A comparative analysis of estimation results between PINNs and CP-PINNs, along with the distribution of loss weights over time, can be found in Figure 5.

## 4 Conclusions

We introduce a novel method for identifying changepoints in dynamic systems governed by general PDEs dynamics. Our approach works with piecewise-constant time-changing parameters and leverages total variation regularization on the first-order differences of parameters. We also propose a meta-learning strategy that balances the priorities between changepoint detection, model fitting, and PDEs discovery. For future works, we plan to explore changepoints linked to both temporal and spatial data, like in Quasi-linear PDEs, and situations where multiple parameters undergo changepoints.

## References


[1] 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.

[2] Markus Bär, Rainer Hegger, and Holger Kantz. Fitting partial differential equations to space-time dynamics. *Physical Review E*, 59(1):337, 1999.

[3] Thorsten G Müller and Jens Timmer. Fitting parameters in partial differential equations from partially observed noisy data. *Physica D: Nonlinear Phenomena*, 171(1-2):1–7, 2002.

[4] Xiaolei Xun, Jiguo Cao, Bani Mallick, Arnab Maity, and Raymond J Carroll. Parameter estimation of partial differential equation models. *Journal of the American Statistical Association*, 108(503):1009–1020, 2013.

[5] Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics express*, 28(8):11618–11633, 2020.

[6] Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E Hurtado, and Ellen Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42, 2020.

[7] George S Misyris, Andreas Venzke, and Spyros Chatzivasileiadis. Physics-informed neural networks for power systems. In *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2020.

[8] Richard J Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. Image change detection algorithms: a systematic survey. *IEEE transactions on image processing*, 14(3):294–307, 2005.

[9] Md Foezur Rahman Chowdhury, S-A Selouani, and D O'Shaughnessy. Bayesian on-line spectral change point detection: a soft computing approach for on-line asr. *International Journal of Speech Technology*, 15(1):5–23, 2012.

[10] David Rybach, Christian Gollan, Ralf Schluter, and Hermann Ney. Audio segmentation for speech recognition using segment features. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4197–4200. IEEE, 2009.

[11] M Staudacher, S Telser, A Amann, H Hinterhuber, and M Ritsch-Marte. A new method for change-point detection developed for on-line analysis of the heart beat variability during sleep. *Physica A: Statistical Mechanics and its Applications*, 349(3-4):582–596, 2005.

[12] Ping Yang, Guy Dumont, and John Mark Ansermino. Adaptive change detection in heart rate trend monitoring in anesthetized children. *IEEE transactions on biomedical engineering*, 53(11):2211–2219, 2006.

[13] Zaıd Harchaoui and Céline Lévy-Leduc. Multiple change-point estimation with a total variation penalty. *Journal of the American Statistical Association*, 105(492):1480–1493, 2010.

[14] Zaid Harchaoui, Eric Moulines, and Francis Bach. Kernel change-point analysis. *Advances in neural information processing systems*, 21, 2008.

[15] Kadierdan Kaheman, Eurika Kaiser, Benjamin Strom, J Nathan Kutz, and Steven L Brunton. Learning discrepancy models from experimental data. *arXiv preprint arXiv:1909.08574*, 2019.

[16] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of computing*, 8(1):121–164, 2012.

[17] Ole Morten Aamo. Leak detection, size estimation and localization in pipe flows. *IEEE Transactions on Automatic Control*, 61(1):246–251, 2015.

[18] Jorge A Laval and Ludovic Leclercq. The hamilton–jacobi partial differential equation and the three representations of traffic flow. *Transportation Research Part B: Methodological*, 52:17–30, 2013.

[19] Yufang Wang, Haiyan Wang, Shuhua Chang, and Adrian Avram. Prediction of daily pm 2.5 concentration in china using partial differential equations. *PloS one*, 13(6):e0197666, 2018.

[20] Zekun Yang, Yu Chen, Ning Zhou, Aleksey Polunchenko, and Yilu Liu. Data-driven online distributed disturbance location for large-scale power grids. *IET Smart Grid*, 2(3):381–390, 2019.

[21] Alexander G Tartakovsky. Rapid detection of attacks in computer networks by quickest changepoint detection methods. In *Data analysis for network cyber-security*, pages 33–70. World Scientific, 2014.

[22] Navid Zobeiry and Keith D Humfeld. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence*, 101:104232, 2021.

[23] Travis E Oliphant, Armando Manduca, Richard L Ehman, and James F Greenleaf. Complex-valued stiffness reconstruction for magnetic resonance elastography by algebraic inversion of the differential equation. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 45(2):299–310, 2001.

[24] Ehsan Haghighat, Umair bin Waheed, and George Karniadakis. A novel deeponet model for learning moving-solution operators with applications to earthquake hypocenter localization. *arXiv preprint arXiv:2306.04096*, 2023.

[25] Vishal Singh, Rajesh Kumar, and SN Patel. Non-linear vibration and instability of multi-phase composite plate subjected to non-uniform in-plane parametric excitation: Semi-analytical investigation. *Thin-Walled Structures*, 162:107556, 2021.

[26] Tianshu Bao, Xiaowei Jia, Jacob Zwart, Jeffrey Sadler, Alison Appling, Samantha Oliver, and Taylor T Johnson. Partial differential equation driven dynamic graph networks for predicting stream water temperature. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 11–20. IEEE, 2021.

[27] Liang Song, Shaodong Chen, and Guoxin Wang. Modeling and analysis of environmental vulnerability based on partial differential equation. *Arabian Journal of Geosciences*, 14:1–9, 2021.

[28] Eike H Müller and Robert Scheichl. Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction. *Quarterly Journal of the Royal Meteorological Society*, 140(685):2608–2624, 2014.

[29] Roel van den Berg, Erjen Lefeber, and Koos Rooda. Modeling and control of a manufacturing flow line using partial differential equations. *IEEE Transactions on Control Systems Technology*, 16(1):130–136, 2007.

[30] Ryan J. Tibshirani. Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics*, 42(1):285 – 323, 2014.

[31] Seyed Ali Hosseini, Nasser Darabiha, and Dominique Thévenin. Lattice boltzmann advection-diffusion model for conjugate heat transfer in heterogeneous media. *International Journal of Heat and Mass Transfer*, 132:906–919, 2019.

[32] Ronbanchob Apiratikul. Application of analytical solution of advection-dispersion-reaction model to predict the breakthrough curve and mass transfer zone for the biosorption of heavy metal ion in a fixed bed column. *Process Safety and Environmental Protection*, 137:58–65, 2020.

[33] Ailian Chang, HongGuang Sun, Chunmiao Zheng, Bingqing Lu, Chengpeng Lu, Rui Ma, and Yong Zhang. A time fractional convection–diffusion equation to model gas transport through heterogeneous soil and gas reservoirs. *Physica A: Statistical Mechanics and its Applications*, 502:356–369, 2018.

[34] Jinping Li, Peng Wu, and Jiandong Yang. Cfd numerical simulation of water hammer in pipeline based on the navier-stokes equation. In *Fifth European Conference on Computational Fluid Dynamics*, 2010.

[35] Rémi Bourguet, Marianna Braza, Alain Sévrain, and Abdellatif Bouhadji. Capturing transition features around a wing by reduced-order modeling based on compressible navier–stokes equations. *Physics of Fluids*, 21(9), 2009.

[36] Jiyang Xia and Dennis YC Leung. Pollutant dispersion in urban street canopies. *Atmospheric Environment*, 35(11):2033–2043, 2001.