Physics-Informed Neural Networks for Non-Newtonian Fluid Thermo-Mechanical Problems: An Application to Rubber Calendering Process

A Preprint

Thi Nguyen Khoa Nguyen¹²³, Thibault Dairay², Raphaël Meunier², Mathilde Mougeot¹⁴

¹Université Paris-Saclay, ENS Paris-Saclay, CNRS, Centre Borelli, Gif-sur-Yvette, 91190 France
²Michelin, Centre de Recherche de Ladoux, Cébazat, 63118 France
³CEA, DAM, DIF, F-91297 Arpajon, France
⁴ENSIIE, Évry-Courcouronnes, 91000 France

June 28, 2022

Abstract

Physics-Informed Neural Networks (PINNs) have gained much attention in various fields of engineering thanks to their capability of incorporating physical laws into the models. However, the assessment of PINNs in industrial applications involving coupling between mechanical and thermal fields is still an active research topic. In this work, we present an application of PINNs to a non-Newtonian fluid thermo-mechanical problem which is often considered in the rubber calendering process. We demonstrate the effectiveness of PINNs when dealing with inverse and ill-posed problems, which are impractical to be solved by classical numerical discretization methods. We study the impact of the placement of the sensors and the distribution of unsupervised points on the performance of PINNs in a problem of inferring hidden physical fields from some partial data. We also investigate the capability of PINNs to identify unknown physical parameters from the measurements captured by sensors. The effect of noisy measurements is also considered throughout this work. The results of this paper demonstrate that in the problem of identification, PINNs can successfully estimate the unknown parameters using only the measurements on the sensors. In ill-posed problems where boundary conditions are not completely defined, even though the placement of the sensors and the distribution of unsupervised points have a great impact on PINNs performance, we show that the algorithm is able to infer the hidden physics from local measurements.

Keywords Physics-informed neural networks, non-Newtonian fluid, thermo-mechanical problem.

1 Introduction

In the last few years, advances in deep learning have gained much attention in the field of physical modeling and engineering. There has been an increasing amount of research using deep learning methods to study the physics of systems by enforcing physical constraints, which are usually expressed by partial differential equations (PDEs), into the models [Raissi et al., 2019, Sirignano and Spiliopoulos, 2018, Rackauckas et al., 2020, Li et al., 2020, Gao et al., 2021]. Among these studies, the series of work of Karniadakis’ group about Physics-Informed Neural Networks (PINNs) [Raissi et al., 2019] rise as an attractive and remarkable scheme of solving forward and inverse PDEs problems using only a moderate amount of supervised data [Zhu et al., 2019, Sun and Wang, 2020, Sun et al., 2020]. PINNs take advantage of both data-driven modeling and physics-based modeling to estimate the unknown physics of interest, which may be physical fields or physical parameters of a system. PINNs use neural networks as approximators and integrate the physical constraints of the systems into the cost function. The training process of PINNs tries to penalize (1) the loss on the initial/boundary condition if the problem is well-defined, (2) the loss on supervised data, which are observed solutions or measurements captured by sensors if they are available, (3) the loss on collocation data to
embed the physical constraints expressed by the governing PDEs. Thanks to their simplicity and powerful capability when dealing with forward or inverse problems, PINNs are gaining more and more attention from researchers and they are gradually being improved and extended [Lu et al., 2021; Wang et al., 2020a; Jagtap et al., 2020b; McClenny and Braga-Neto, 2020; McClenny et al., 2021; Jagtap et al., 2020a; 2022b]. Besides that, there are continuing efforts to overcome the limitation of PINNs in terms of high computational costs [Jagtap et al., 2020c; Jagtap and Karniadakis, 2020; Shukla et al., 2021b]. Thus far, the applicability of PINNs has been demonstrated in various fields of research. For example, Kissas et al. [2020] employed PINNs for the modeling of cardiovascular flows, Tartakovskiy et al. [2020] and He et al. [2020] applied PINNs to subsurface flow and transport problems, Shukla et al. [2021a] used PINNs to identify polycrystalline nickel material coefficients. Recently, software editors (e.g. NVIDIA) are also showing great interest for this kind of methods [Hennigh et al., 2021; Cai et al., 2021c].

As far as computational fluid dynamics (CFD) is concerned, there has been a growing number of studies that focus on the capability of PINNs to infer the solution or estimate unknown physical parameters in problems of fluid mechanics [Raissi et al., 2020; Rao et al., 2020; Reyes et al., 2021; Zobeiry and Humfeld, 2021]. Raissi et al. [2020] employed a method that used PINNs to infer hidden physics states including the velocity and pressure fields from partial knowledge of a relevant variable (the concentration field in their work) by using the governing physical laws. This work demonstrated the effectiveness of PINNs to solve ill-posed problems and inspired many applications [Cai et al., 2021a,b,c; Jin et al., 2021; Mao et al., 2020; Rao et al., 2020]. Rao et al. [2020] proposed a new scheme for PINNs to model incompressible laminar fluid flows without using any supervised measurements. Reyes et al. [2021] used PINNs to estimate the viscosity of non-Newtonian fluid flow by using only velocity measurements. Recently, Mahmoudabadbozchelou et al. [2022] proposed the non-Newtonian PINNs for complex fluid flows with different non-Newtonian constitutive models. Through these previous studies, PINNs have shown a promising power to deal with ill-posed inverse problems with multi-physics features.

In the industrial context, we often dispose of data that are captured from sensors. However, the impact of the location of these sensors on the performance of PINNs has not been paid much attention to. This is a crucial question in industrial use cases as the supervised measurements, which mostly depend on the placement of the sensors, are not randomly or uniformly distributed but only located at specific positions. Recently, increasing efforts have been made to study the optimal locations of the sensors [Jagtap et al., 2022a; Cai et al., 2021c]. Besides that, the distribution of unsupervised points (or collocation points) also plays an important role in the training process as the PDE residual is evaluated at these points. In most existing applications of PINNs, these collocation points are often chosen randomly inside the domain or generated using a Latin Hypercube Sampling strategy, which may cause large errors on the prediction at the points of discontinuity or at high gradient locations. This opens the opportunity to improve the accuracy of PINNs by taking the collocation points from some mesh that provides an a priori knowledge on the location of discontinuity or high gradient location (e.g. finite element mesh).

In this paper, we introduce a novel application of PINNs to infer the solution and identify unknown physical parameters in a non-Newtonian fluid thermo-mechanical problem. We consider an incompressible non-Newtonian fluid flow that is governed by a generalized Stokes equation coupled with a heat transfer equation. This set of PDEs is commonly used to model rubber manufacturing processes in the tire industry. First, we assess the performance of PINNs using virtual sensors coming from an industrial calendering problem simulation. To this end, we study the influence of the location of sensors and the distribution of unsupervised data on the accuracy of PINNs prediction in an ill-posed problem, where boundary conditions are not completely defined. More precisely, we suppose that only the measurements for the temperature are captured by the sensors, we aim to infer the velocity, the pressure, and the temperature at all points in the domain. Then, we examine the capability of PINNs to estimate unknown physical parameters from noisy measurements in an inverse problem. All our results are compared to a reference solution obtained by an in-house finite element solver.

The following of this paper is organized as follows. In section 2, we briefly review the framework of PINNs, their scope of use and methods using adaptive activation functions and deep Kronecker networks to improve the accuracy of PINNs. We then study the influence of the position of supervised and unsupervised data on the accuracy of PINNs to solve a thermo-mechanical rubber calendering problem in section 3. The effect of noisy measurements is also investigated in this section. Finally, we summarize the conclusions in section 4.

2 Methodology

In this section, the framework of Physics-Informed Neural Networks (PINNs) [Raissi et al., 2019] is briefly presented. Later in this section, the approaches using locally adaptive activation functions [Jagtap et al., 2020a] and deep Kronecker networks with adaptive activation functions [Jagtap et al., 2022b] to improve the performance of PINNs are discussed.
2.1 Physics-Informed Neural Networks

PINNs aim at solving three main classes of problems: forward problems, inverse problems, and ill-posed problems for PDEs. For the forward problems the governing equations are known, i.e. the physics laws (i.e. PDEs with boundary/initial conditions), and the physical parameters are well-defined. In this case, the goal is to infer the solution of the PDEs. For the inverse problems, some measurements for the PDE solutions are available and some physical parameters are unknown. The goal is to identify the values of unknown parameters. For ill-posed problems, partial data associated with the PDEs solutions are available, but the initial and/or boundary conditions are not well-defined, and the purpose is to infer all the physical fields of interest.

To illustrate the methodology of PINNs, let us consider the following parameterized PDE defined on the domain \(\Omega \subset \mathbb{R}^d\) with the boundary \(\partial \Omega\):

\[
\begin{align*}
  &u_t + \mathcal{N}_x(u, \lambda) = 0, \text{ for } x \in \Omega, t \in [0,T] \\
  &B(u, x, t) = 0, \text{ for } x \in \partial \Omega \\
  &u(x, 0) = g(x), \text{ for } x \in \Omega
\end{align*}
\]

where \(x \in \mathbb{R}^d\) and \(t\) are the spatial and temporal coordinates, \(\mathcal{N}_x\) is a differential operator, \(\lambda\) is the PDE parameter, \(u\) is the solution of the PDE with initial condition \(g(x)\) and boundary condition \(B\), which could be Dirichlet, Neumann, Robin, or periodic boundary conditions. The subscripts denote the partial differentiation in time or space.

In the conventional framework of PINNs [Raissi et al., 2019], the solution \(u\) of the PDE is approximated by a fully-connected feedforward neural network \(\mathcal{N}\), which takes the spatial and temporal coordinates \((x, t)\) as inputs and the solution \(u\) as output, given a set of supervised data. This can be represented as follows:

\[
u \approx \hat{u} = \mathcal{N}_N^D(x, t, \theta)
\]

where \(\hat{u}\) denotes the prediction value for the solution and \(\theta\) denotes the trainable parameters of the neural network. The neural network \(\mathcal{N}\) is composed of \(D\) hidden layers. Let the hidden variable of the \(l^{th}\) hidden layer be denoted by \(\mathcal{N}_N^l\) then the neural network can be expressed as:

\[
\begin{align*}
  \mathcal{N}_N^0 &= (x, t) \\
  \mathcal{N}_N^l &= \sigma(W^l\mathcal{N}_N^{l-1} + b^l) \text{ for } 1 \leq l \leq D - 1 \\
  \mathcal{N}_N^D &= W^D\mathcal{N}_N^{D-1} + b^D
\end{align*}
\]

For the problems of inferring the PDE solutions, the trainable parameters \(\theta\) are defined by the weights \(\{W^l\}_{l=1}^D\) and the biases \(\{b^l\}_{l=1}^D\) of the neural networks. For the problems of identification, these parameters include, in addition, the PDEs parameters that need to be defined. The parameters of the neural network are trained by minimizing the cost function \(L\):

\[
L = L_{pde} + L_{ic} + L_{bc} + L_{data}
\]

where the terms \(L_{pde}, L_{ic}, L_{bc}\) and \(L_{data}\) penalize the loss in the residual of the PDE, the initial condition, the boundary condition, and the supervised data (measurements), respectively, which can be represented as follow:

\[
\begin{align*}
  L_{pde} &= \frac{1}{N_{pde}} \sum_{i=1}^{N_{pde}} |\hat{u}_{ti} + \mathcal{N}_x(\hat{u}_i, \lambda)|^2 \\
  L_{ic} &= \frac{u_{ic}}{N_{ic}} \sum_{i=1}^{N_{ic}} |\hat{u}(x_i, 0) - g(x_i)|^2 \\
  L_{bc} &= \frac{u_{bc}}{N_{bc}} \sum_{i=1}^{N_{bc}} |B(\hat{u}_i, x_i, t_i)|^2 \\
  L_{data} &= \frac{u_{data}}{N_{data}} \sum_{i=1}^{N_{data}} |\hat{u}(x_i, t_i) - u(x_i, t_i)|^2
\end{align*}
\]

where \(N_{ic}, N_{bc}, N_{data}\) denote the numbers of learning points for the initial condition, boundary condition, and measurements, respectively, and \(N_{pde}\) denotes the number of residual points (or collocation points or unsupervised...
points) of the PDE. $w_{ic}, w_{bc}$ and $w_{data}$ are the weight coefficients for different loss terms, which are used to adjust the contribution of each term to the cost function. These coefficients can be either pre-specified before the training or tuned during the training. Recently, many studies aim at improving the efficiency of PINNs by using adaptive weights in the cost function [Wang et al., 2020a,b; McCleneny and Braga-Neto, 2020]. We note that the number $N_{data}$ depends on the supervised data while the numbers $N_{ic}, N_{bc}$ (if the initial and boundary conditions are well-defined) and $N_{pde}$ are user-defined. To construct the residual loss $L_{pde}$, automatic differentiation (AD) [Baydin et al., 2018] is used to compute the partial derivatives of the output $\hat{u}$ of the network with respect to the inputs ($x, t$). AD builds a computation graph to calculate the gradients numerically. The benefit of using AD to compute the derivatives is that we do not need to discretize the domain, which is essential in classical numerical methods.

The definition of the cost function (1) depends on the problems. For example, for forward and well-posed problems where the initial and boundary conditions of the PDE are well-defined, the loss term for the measurements $L_{ic}, L_{bc}$ dissolve, then the term $L_{data}$ becomes mandatory for the neural network to approximate the solutions. We also note that PINNs may provide different performances with different network initializations. In this work, when comparing results of the same configuration of PINNs (i.e. the same networks architecture, activation function and optimizer), we train PINNs from random initialization five times and then choose the model with the smallest training loss as the final solution for both visualization and numerical comparison. When comparing results of different configuration of PINNs, we train PINNs from random initialization five times and then choose the model with the smallest training loss as the final solution for the visualization, and choose the mean and standard deviation values of the five models for numerical comparison.

### 2.2 Locally adaptive activation functions

Jagtap et al. [2020a] proposed two approaches using layer-wise (L-LAAFs) and neuron-wise (N-LAAFs) locally adaptive activation functions to improve the performance of deep and physics-informed neural networks. These methods have been proven to be a very efficient way to train the network and have been employed successfully in many applications [Markidis, 2021; Gnanasambandam et al., 2022; Bai et al., 2022]. Here, we briefly present the formulation of L-LAAFs and N-LAAFs which closely follows the framework introduced by Jagtap et al. [2020a]. These methods introduce a scalable parameter for each layer (layer-wise) or for every neuron (neuron-wise) separately. A slope recovery term is also added in the cost function to accelerate the training process.

- **Layer-wise locally adaptive activation functions (L-LAAFs):** in the $l^{th}$ layer of the neural network, we add an adaptive parameter $a^l$ scaled by a pre-specified factor $n$ as:

$$\mathcal{N}_l = \sigma(na^l(W^l \mathcal{N}^{l-1} + b^l)) \quad \text{for} \quad l = 1, \ldots, D - 1$$

- **Neuron-wise locally adaptive activation functions (N-LAAFs):** in the $l^{th}$ layer with $N_l$ neurons, we add adaptive parameters $a_{i}^l$ scaled by a pre-specified factor $n$ for each neuron as:

$$\mathcal{N}_l = \sigma(na_{i}^l(W^l \mathcal{N}^{l-1} + b^l)_{i}) \quad \text{for} \quad l = 1, \ldots, D - 1; \quad i = 1, \ldots, N_l$$

The trainable parameters of the neural networks $\theta$ is constituted of $\{W^l, b^l\}_{l=1}^D$ and $\{a^l\}_{l=1}^{D-1}$ for L-LAAFs or $\{a_{i}^l\}_{i=1}^{D-1}$ for N-LAAFs. The scaling factor $n$ and the adaptive parameters $a^l$ (or $a_{i}^l$) are often initialized as $na^1 = 1$ (or $na_{i}^1 = 1$), so that the initialization of LAAFs is the same as the standard fixed activation function. A slope recovery term $S_a$ is added to the cost function in order to force the neural network to increase the value of activation slope quickly and thus increase the training speed:

$$L = L_{pde} + L_{ic} + L_{bc} + L_{data} + S_a$$

where $S_a$ is defined as:

$$S_a = \begin{cases} 
\frac{1}{D-1} \sum_{l=1}^{D} \exp(a^l) & \text{for L-LAAFs} \\
\frac{1}{D-1} \sum_{i=1}^{N_l} \exp \left(\frac{\sum_{l=1}^{D} a_{i}^l}{N_l} \right) & \text{for N-LAAFs}
\end{cases}$$

We note that introducing additional trainable parameters for each layer or each neuron will lead to higher computational cost compared to vanilla PINNs or the method using the global adaptive activation function introduced in Jagtap et al. [2020b]. However, the computational cost remains comparable between these methods as the depth and width of the networks increase. Furthermore, adding additional parameters in L-LAAFs and N-LAAFs gives the neural networks more flexibility (as higher degrees of freedom) and thus increases the capability of training.
2.3 Deep Kronecker neural networks with adaptive activation functions

Kronecker neural networks (KNNs) \cite{Jagtap2022} is a general framework for neural networks with adaptive activation functions. KNNs employ the Kronecker product to construct the weight matrices and bias vectors. A detailed description of the method can be found in the cited reference. Here we present the simplest formulation of KNNs, which can be viewed as a generalization of the standard feedforward networks. More precisely, in the \( l \)th layer of KNNs, we introduce trainable parameters \( \{\alpha_k^l, \omega_k^l\}_{k=1}^{K} \) as:

\[
    \mathcal{N}^l = \sum_{k=1}^{K} \alpha_k^l \sigma_k^l (\omega_k^l (W^l \mathcal{N}^{l-1} + b^l)) \quad \text{for} \quad l = 1, \ldots, D - 1
\]

where \( \{\sigma_k^l\}_{k=1}^{K} \) are fixed activation functions. We note that the feedforward network is a special case of KNNs when \( K = 1 \) and \( \alpha_k^l, \omega_k^l = 1 \forall l, k \). When \( K = 1 \) and \( \alpha_k^l = 1 \forall l, k \), KNNs can be seen as a feedforward network with layer-wise locally adaptive activation functions (L-LAAFs). When \( K \) increases, the number of trainable parameters also increases, which helps to improve the training capability but however leads to high computation cost in KNNs.

In the original work of \cite{Jagtap2022}, the authors proposed a particular network of KNNs named as Rowdy neural network (Rowdy Net), which uses sinusoidal functions as activation functions to capture high-frequency components in the target function:

\[
    \sigma_k(x) = n \sin((k-1)nx) \quad \text{for} \quad k = 2, \ldots, K
\]

where \( n \) is a scaling factor and \( \sigma_1 \) can be any standard activation function. In this work, we choose \( \sigma_1(x) = \tanh(x) \). We will compare Rowdy Net with the KNNs using only \( \tanh \) activation functions \( \sigma_k(x) = \tanh(x) \) for \( k = 1, \ldots, K \). We refer to this network as KNNs-tanh in the following.

3 Application to rubber calendering modeling

3.1 Problem configuration

In the industry of tire manufacturing, calendering is a mechanical process used to create and assemble thin tissues of rubber. As sketched in Figure 1 during calendering, the rubber is smoothed out through several rotating cylindrical rolls. The modeling of the entire calendering process is a challenging task involving different length and time scales, different materials, and potentially 3D effects. However, for the sake of simplicity, the goal of the present study is only to model the 2D temperature, velocity, and pressure fields inside rubber materials going through two contra-rotating rolls of the calender as depicted in Figure 2.

![Figure 1: Sketch of a calender used to create thin tissues of rubber for tire manufacturing. The rubber material is colored in black for illustration.](image)

The physical domain of interest together with the geometrical configuration of the study are presented in Figure 2 the 2D domain is simply bounded by an inlet, two rotating boundaries corresponding to the edges of the rolls, and an outlet. From an industrial point of view, it is of prime interest to monitor the rubber temperature inside the domain.
In particular, because rubber materials are subjected to viscous heating, an important temperature elevation can be observed in highly-sheared regions (specifically at the inlet where the material can be accumulated in a rubber bead and in the very thin gap between the rolls). Such high temperatures can lead to unexpected rubber curing that could deteriorate the final material quality. In the next section, the physical equations used to model the rubber mechanical and thermal fields will be introduced.

Figure 2: Sketch of the 2D configuration and geometrical setup considered in the present study. The physical domain of interest is colored in green.

3.2 Problem formulation

From the physical point of view, the rubber is assimilated as an incompressible non-Newtonian fluid flow in the present study (in particular, the elastic part of the material is not considered here). Moreover, only the steady-state regime is considered. Consequently, the equations describing the behaviour of the rubber are a generalized Stokes equation for momentum conservation, a divergence-free velocity equation for mass conservation, and a convection-diffusion equation with a viscous heating term for energy conservation. The constitutive law considered in all the presented cases is a power-law combined with an Arrhenius law to take into account the dependence of viscosity on temperature. The governing system of PDEs can be written as follows:

\[ \nabla \cdot (2\eta(\vec{u}, T) \varepsilon(\vec{u})) - \nabla p = \vec{0} \]  
\[ \nabla \cdot \vec{u} = 0 \]  
\[ \vec{u} \cdot \nabla T = \frac{\lambda}{\rho C_p} \nabla^2 T + \frac{1}{\rho C_p} \eta(\vec{u}, T) |\gamma(\vec{u})|^2 \]

where \( \vec{u} = (u_x, u_y)^T \) is the velocity vector, \( p \) is the pressure, \( T \) is the temperature, \( \lambda = 20 \text{ W.m}^{-1} \text{K}^{-1} \) is the thermal conductivity, \( \rho = 1000 \text{ kg.m}^{-3} \) is the rubber density, \( C_p = 1000 \text{ J.K}^{-1} \) is the heat capacity at 25°C. The strain-rate tensor \( \varepsilon \) is defined as:

\[ \varepsilon(\vec{u}) = \frac{1}{2} \left( \begin{array}{cc} 2 \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \\ \frac{\partial u_y}{\partial x} & 2 \frac{\partial u_y}{\partial y} \end{array} \right) \]

The dynamic viscosity \( \eta \) is defined as:

\[ \eta(\vec{u}, T) = K |\gamma(\vec{u})|^{n-1} \exp\left( \frac{E_{\alpha}}{R} \left( \frac{1}{T} - \frac{1}{T_{\alpha}} \right) \right) \]

where \( K = 2 \times 10^5 \text{ Pa.s}^n \) is the consistency, \( \tau = 1 \text{ s} \) is the characteristic time, \( n = 0.2 \) is the power law exponent, \( E_{\alpha} = 7000 \text{ J.mol}^{-1} \) is the activation energy of the rubber, \( R \) is the perfect gas constant, and \( \gamma(\vec{u}) = \sqrt{2 \sum_{i,j} \xi^2_{i,j}} \) is the generalized strain rate.

We consider the following quantities of reference: a length \( L_0 = 0.01 \text{ m} \), a velocity \( u_0 = 0.5 \text{ m.s}^{-1} \), two temperatures \( T_1 = 393.15 \text{ K}, T_2 = 476.75 \text{ K} \), and for the pressure, we choose \( p_0 = K \frac{u_0}{L_0} = 10^7 \text{ Pa} \). The corresponding
dimensionless variables are: \( \tilde{u}_x = \frac{u_x}{u_0}, \tilde{u}_y = \frac{u_y}{u_0}, \tilde{T} = \frac{T - T_1}{T_2 - T_1}, \tilde{x} = \frac{x}{L_0}, \tilde{y} = \frac{y}{L_0}. \) Since the dimension of the consistency \( K \) is a function of \( n \), the quantity \( \tilde{p} = \frac{p}{p_0} \) is still a dimensional number.

The system of PDEs is rewritten as:

\[
\tilde{\nabla} \cdot \left( 2\tilde{\eta}(\tilde{\mathbf{u}}, \tilde{T}) \tilde{\epsilon}(\tilde{\mathbf{u}}) \right) - \tilde{\nabla} \tilde{p} = 0 \\
\tilde{\nabla} \cdot \tilde{\mathbf{u}} = 0 \\
\tilde{\mathbf{u}} \tilde{\nabla} \tilde{T} = \frac{1}{Pe} \tilde{\nabla}^2 \tilde{T} + \frac{Br}{Pe} \tilde{\eta}(\tilde{\mathbf{u}}, \tilde{T}) |\tilde{\gamma}(\tilde{\mathbf{u}})|^2
\]

where the Péclet number \( Pe = \frac{\rho C_p u_0 L_0}{\lambda} \) and the Brinkman number \( Br = \frac{K u_0^2}{\lambda (T_2 - T_1)} \) are dimensionless. The dimensionless dynamic viscosity \( \tilde{\eta} \) is defined as follows:

\[
\tilde{\eta}(\tilde{\mathbf{u}}, \tilde{T}) = |\tilde{\tau}(\tilde{\mathbf{u}})|^{n-1} \exp \left( \frac{E_\alpha}{R} \left( \frac{1}{T_2 - T_1 T + T_1} - \frac{1}{T_\alpha} \right) \right)
\]

For the rest of this paper, we shall only consider the dimensionless PDE system in the PINNs training process.

### 3.3 Reference solutions and validation framework

To generate reference High-Fidelity (HF) solutions of velocity, pressure, and temperature fields, the equations (5), (6), (7) are discretized and solved using an in-house generic and multi-purpose finite element solver named MEF++ and co-developed by Laval University and Michelin [Laval and GIREF 2021, Guénette et al. 2004]. The velocity and temperature boundary conditions prescribed in the finite element simulation are given in Figure 3. As the first aim of this study is to assess the impact of the location of supervised points on the performance of PINNs, data captured from real sensors would be very valuable. However, obtaining different scenarios of the placement of the sensors in a real calender is an expensive and impractical process for a research study. Therefore, in our work, we use the HF solutions provided by the finite element solver as virtual measurements obtained by sensors. To evaluate the performance of PINNs in predicting physical fields, we calculate the relative \( L^2 \) error defined as follows:

\[
\epsilon_v = \frac{||v - \hat{v}||_2}{v_{\text{max}} - v_{\text{min}}}
\]

where \( v \) denotes the reference simulated field of interest and \( \hat{v} \) is the corresponding PINNs prediction. The choice is made to divide the absolute error by the reference field amplitude instead of the reference field \( L^2 \) norm in order to avoid any artificial high values of the error for fields very close to zero (see for example \( u_y \) in Figure 7a). In inverse problems where the goal is to identify unknown scalar parameters, we calculate the relative error as follows:

\[
\epsilon_\mu = \frac{(\mu - \hat{\mu})^2}{\mu^2}
\]

where \( \mu \) denotes the reference value of the unknown parameter and \( \hat{\mu} \) is the corresponding PINNs prediction.
Physics-informed neural networks for non-Newtonian fluid thermo-mechanical problems: an application to rubber calendering process

(a) Computational domain and boundary conditions

(b) Mesh close to the inlet

(c) Mesh close to the outlet

Figure 3: Sketch of the computational domain (the mesh is colored in black) with the velocity and temperature boundary conditions used in the reference finite element simulation: $\mathbf{n}$ and $\mathbf{\tau}$ denote respectively the normal and tangential vectors with respect to the corresponding boundary.

In our work, the errors are evaluated on the finite element mesh, which provides us an \textit{a priori} knowledge on the high gradient location. The number of points on the mesh is $N = 163,596$. The in-house solver gives us the high-fidelity solution on the finite element mesh. Figure 4 shows the visualization of 1,000 random points on this mesh compared to 1,000 randomly uniformly distributed points inside the domain. Figure 5 illustrates the density of points on the line $y = 0$ for each case. We point out that when the points are taken from the finite element mesh, the density of points is higher at the output than at the input of the calender. This is because the finite element mesh is refined to take into account the very thin gap between the rolls close to the domain outlet.

Figure 4: Visualization of 1,000 random points on the meshes
Physics-informed neural networks for non-Newtonian fluid thermo-mechanical problems: an application to rubber calendering process

3.4 Inferring physical fields of interest from sensor measurements

In this section, we suppose that the boundary conditions of the problem are not completely defined, and we dispose of some measurements of the temperature from virtual sensors (in our case, coming from our finite element simulation). The goal is to infer the pressure, velocity, and temperature fields at all points in the domain. We note that here, no information on the pressure field is given, but only its gradient in the PDE residual is. Thus the pressure is only identifiable up to a constant.

3.4.1 Impact of sensor location

We aim at studying the impact of the location of the sensors and the number of supervised points on the accuracy of PINNs prediction. For this purpose, we use PINNs with the spatial coordinates x, y as inputs and T, p, u_x, u_y as outputs and minimize the following cost function:

\[ L = L_{\text{data}} + L_{\text{pde}} = \frac{1}{N_T} \sum_{i=1}^{N_T} \omega_T (\hat{T} - T^*)^2 + \frac{1}{N_f} \sum_{i=1}^{N_f} (\omega_1 \epsilon_1^2 + \omega_2 \epsilon_2^2 + \omega_3 \epsilon_3^2 + \omega_4 \epsilon_4^2) \]

where \( \hat{T} \) are the predictions obtained by the neural network and \( T^* \) are the supervised data (measurements captured from the sensors) for the temperature. We note that there is no data for the velocity and the pressure in the cost function except their relation with the temperature given by the system of PDEs. The quantities \( \epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4 \) correspond to the dimensionless PDEs residuals defined in the equations \( 5, 6, 7 \). More precisely:

\[ \begin{align*}
\epsilon_1 &= \frac{\partial}{\partial x} \left( 2\eta(\hat{u}, \hat{T}) \frac{\partial \hat{u}_x}{\partial x} \right) + \frac{\partial}{\partial y} \left( \eta(\hat{u}, \hat{T}) \left( \frac{\partial \hat{u}_x}{\partial y} + \frac{\partial \hat{u}_y}{\partial x} \right) \right) - \frac{\partial \hat{p}}{\partial x} \\
\epsilon_2 &= \frac{\partial}{\partial x} \left( \eta(\hat{u}, \hat{T}) \left( \frac{\partial \hat{u}_x}{\partial y} + \frac{\partial \hat{u}_y}{\partial x} \right) \right) + \frac{\partial}{\partial y} \left( 2\eta(\hat{u}, \hat{T}) \frac{\partial \hat{u}_y}{\partial y} \right) - \frac{\partial \hat{p}}{\partial y} \\
\epsilon_3 &= \frac{\partial \hat{u}_x}{\partial x} + \frac{\partial \hat{u}_y}{\partial y} \\
\epsilon_4 &= \hat{u}_x \frac{\partial \hat{T}}{\partial x} + \hat{u}_y \frac{\partial \hat{T}}{\partial y} - \frac{1}{Pe} \left( \frac{\partial^2 \hat{T}}{\partial x^2} + \frac{\partial^2 \hat{T}}{\partial y^2} \right) - \frac{Br}{Pe} \eta(\hat{u}, \hat{T}) |\gamma(\hat{u})|^2
\end{align*} \]

The weight coefficients \( \omega_i \) for \( i \in \{T, 1, 2, 3, 4\} \) are pre-specified in our case (see below for their specific values). We suppose that there are sensors in the domain so that we can measure the temperature of the flow at these points. The placement of these sensors may be important as it may affect the inferred results. We study three cases of the sensors’ placements as shown in Figure 6. In the first case (Figure 6a), the sensors are put randomly inside the calender. This is often the test case used in academic applications, but not always realistic in an industrial context. In the second case (Figure 6b), the sensors are put on two vertical lines located at the input and the output of the calender. In the third case (Figure 6c), the sensors are put randomly in the input zone of the calender. The last two cases are tested as they are often the case in industrial tasks.
Physics-informed neural networks for non-Newtonian fluid thermo-mechanical problems: an application to rubber calendering process

To study the impact of the sensor’s location on PINNs performance, we fix the number of measurements (supervised points) for the temperature $N_T = 500$. For the training of PINNs, the number of collocation points (unsupervised points) is $N_f = 10,000$ and these points are generated randomly on the finite element mesh, which provides an \textit{a priori} knowledge on high gradient location (see Figure 4b). In all cases, we use a feedforward neural network with 5 hidden layers with 100 neurons per layer. This architecture is inspired by the work of [Raissi et al., 2019]. Our preliminary results suggest that this choice gives a good balance between the network representation capacity and the computational costs. First, we use PINNs with $tanh$ activation function without employing locally adaptive activate functions or deep Kronecker networks. To minimize the cost function, we adopt Adam optimizer with learning rate decay strategy, which is proven to be very efficient in training deep learning models [You et al., 2019]. The results are obtained with 50,000 epochs with the learning rate $lr = 10^{-3}$, 200,000 epochs with the learning rate $lr = 10^{-4}$ and 150,000 epochs with $lr = 10^{-5}$. For the weight coefficients, we fix $\omega_1 = \omega_2 = \omega_3 = \omega_4 = 1$ and $\omega_T = \frac{N_T}{\sum_{i=1}^{N_T} (T_i)^2}$. By putting these weights, we aim at minimizing the relative errors between the PINNs prediction and solution for the supervised points in the cost function, instead of minimizing the absolute errors as in the conventional PINNs [Raissi et al., 2019].

Figure 6: \textit{Geometry of the rubber calender and the location of sensors}: the black points represent the location of sensors (or supervised points for the temperature).

Figure 7 illustrates the visual comparison between the reference solution and PINNs prediction in different scenarios of the location of sensors. We observe that in all three cases, PINNs are only able to approximate accurately the temperature in the zones where there are sensors and fail to predict the velocity and pressure. We note that since there is no supervised measurement for the velocity and pressure, these fields can be accurately approximated only after the temperature is. Figure 8 shows the detail of each loss term in the cost function. We see that after the training process, the loss for the data (temperature measurements) and the loss for PDEs residuals are both minimized to very small values. This performance demonstrates that even when the temperature is accurately approximated at all points inside the domain (as case 1) and the PDEs constraints are well respected, PINNs still fail to infer the velocity and the pressure, \textit{i.e.} the information of the temperature is not sufficient to guarantee a unique solution for the velocity and the pressure fields. The same conclusion is obtained when using PINNs with locally adaptive activation functions or deep Kronecker networks (which are not shown here for conciseness).
To make sure that the problem has a unique solution, we take in addition the knowledge of the velocity boundary conditions. Moreover, since the supervised points for the temperature are only available at some specific location in case 2 (Figure 6b) and case 3 (Figure 6c), PINNs, as other deep learning methods, have difficulties to extrapolating the values of the temperature at points far from the sensors. Thus we also provide PINNs with the information of the temperature on the roll boundaries only. We note that when the sensors are distributed as in case 1 (Figure 6a), the learning points for the temperature on the roll boundaries are not necessary. The cost function in PINNs now becomes:

\[
L = L_{\text{data}} + L_{\text{pde}} + L_{\text{bc}}
\]

\[
= \frac{1}{N_T} \sum_{i=1}^{N_T} \omega_T (\hat{T}^i - T^i)^2 + \frac{1}{N_f} \sum_{i=1}^{N_f} (\omega_1 e_1^2 + \omega_2 e_2^2 + \omega_3 e_3^2 + \omega_4 e_4^2) + \frac{1}{N_{\text{rolls}}} \sum_{i=1}^{N_{\text{rolls}}} (\omega_{\text{rolls}} (\hat{T}^\text{rolls,}i - T^\text{rolls,}i)^2) + \frac{1}{N_{\text{bc}}} \sum_{i=1}^{N_{\text{bc}}} \left( \omega_{\text{bcx}} (\hat{u}^{\text{bcx},i} - u^{\text{bcx},i})^2 + \omega_{\text{bcy}} (\hat{u}^{\text{bcy},i} - u^{\text{bcy},i})^2 \right)
\]

where \(T^\text{rolls,}i\) denotes the values of the temperature on the rolls and \(u^{\text{bcx},i}, u^{\text{bcy},i}\) denotes the components of the velocity on the four boundaries of the calender. For the weight coefficients, we fix \(\omega_1 = \omega_2 = \omega_3 = \omega_4 = 1, \omega_T = \frac{N_T}{\sum_{i=1}^{N_T} (T^i)^2}, \omega_{\text{rolls}} = \frac{N_{\text{rolls}}}{\sum_{i=1}^{N_{\text{rolls}}} (T^\text{rolls,}i)^2}, \omega_{\text{bcx}} = \frac{100N_{\text{bcx}}}{\sum_{i=1}^{N_{\text{bcx}}} (u^{\text{bcx},i})^2} \) and \(\omega_{\text{bcy}} = \frac{10N_{\text{bcy}}}{\sum_{i=1}^{N_{\text{bcy}}} (u^{\text{bcy},i})^2} \). Here we still minimize the relative errors between the prediction and the solution at the learning points, but we put more important weights on the boundary condition for the velocity.

![Figure 8: Cost function during training process of PINNs.](image)

![Figure 9: Cost function during training process of PINNs after adding the knowledge for the velocity boundary conditions.](image)
Next, we compare vanilla PINNs with PINNs using locally adaptive activation function (L-LAAFs and N-LAAFs) and PINNs with deep Kronecker networks (Rowdy Net and KNNs-tanh) to verify whether these methods help to improve the performance of PINNs. For L-LAAFs and N-LAAFs, we fix the scaling factor $n = 1$ and initialize the adaptive parameters $a^k = 1$ or $a^k_i = 1$. For Rowdy Net and KNNs-tanh, we fix $K = 2$ and $a^k_i = \omega^k = 1$ and the scaling factor $n = 1$ (for Rowdy Net). Figure 11 shows the cost function during the training process of these methods in different scenarios of sensor location. Table 1 illustrates the performance of each model in case 2 of sensor location in terms of relative $L^2$ errors with the reference solution. We see that when using locally adaptive activation functions or deep Kronecker networks, PINNs provide better results compared to vanilla PINNs. This performance is as expected since these models offer higher degrees of freedom in the optimization problem than vanilla PINNs. Thus the neural networks have additional flexibility in training, which leads to better performance than vanilla PINNs. When comparing LAAFs and deep Kronecker networks, we see that the model with KNNs-tanh gives the best accuracy for all physical fields, and then come N-LAAFs, Rowdy Net and L-LAAFs. However, during the training process, we observe that the models with deep Kronecker networks, especially KNNs-tanh, are very sensitive during the training process with high learning rate. These models only become more stable when the learning rate decreases. We note that we can further increase the accuracy of Rowdy Net and KNNs-tanh by increasing the number of terms $K$, which however leads to a high computational cost. In this work, we only consider $K = 2$, which already yields satisfying performance. Besides that, in Table 1 we see that the error for $T$ is larger than the other fields. We can reduce this error by increasing the number of training epochs (see Table 3 for more details).

![Figure 10: Reference solution and PINNs prediction when using in additional the boundary condition for the velocity and the temperature as learning data.](image)

![Figure 11: Cost function during training process of PINNs with different approaches.](image)

| Method         | $\epsilon_T$   | $\epsilon_{u_x}$ | $\epsilon_{u_y}$ | $\epsilon_P$ |
|----------------|-----------------|-------------------|-------------------|--------------|
| Vanilla        | 11.1 ± 0.14     | 7.57 ± 0.12       | 5.73 ± 0.04       | 1.79 ± 0.00  |
| L-LAAFs        | 10.8 ± 0.16     | 7.50 ± 0.13       | 5.69 ± 0.05       | 1.61 ± 0.00  |
| N-LAAFs        | 10.2 ± 0.14     | 7.00 ± 0.09       | 5.19 ± 0.02       | 1.10 ± 0.00  |
| Rowdy Net      | 10.3 ± 0.13     | 7.24 ± 0.09       | 5.33 ± 0.03       | 1.32 ± 0.00  |
| KNNs-tanh      | 10.1 ± 0.16     | 6.89 ± 0.10       | 5.05 ± 0.04       | 0.96 ± 0.00  |

Table 1: Relative $L^2$ errors between reference solution and PINNs prediction with different methods in case 2.
Next, we study two different cases for the distribution of collocation points: in the first case, these points are generated randomly inside the domain while in the second case, they are generated randomly on the finite element mesh. An example of the visualization of these collocation points can be seen in Figure 4. Table 2 summarizes the performance of PINNs using KNNs-tanh approach in terms of relative $L^2$ errors compared with the reference solution. These errors are evaluated on the finite element mesh. It can be observed that, using the collocation points randomly taken from the finite element mesh allows PINNs to give a much better precision than using the points randomly taken inside the domain. We also notice that, since the errors are evaluated on the finite element mesh which is much finer at the output, the error of the prediction at the output gives a more important weight than the one at the input in the final accuracy. This remark can be seen clearly in case 3 of the placement of the sensors where we obtain a very high relative error for $u_x$ and $p$, even though, PINNs only fail to predict accurately the solution at the output of the calender (which is reasonable as we only have supervised points at the input).

| Case | $\epsilon_T$ (1) | $\epsilon_{u_x}$ (1) | $\epsilon_{u_y}$ (1) | $\epsilon_p$ (1) |
|------|-----------------|-----------------|-----------------|-----------------|
| Case 1 | 14.3 | 17.8 | 5.59 | 18.9 |
| Case 2 | 13.9 | 49.9 | 10.3 | 0.95 |
| Case 3 | 14.8 | 125 | 5.76 | 55.1 |

Table 2: Relative $L^2$ errors between the reference solution and the predictions by PINNs in different scenarios of sensors’ placement. The errors are evaluated on the finite element mesh (FE). We compare two cases of the collocation points’ distribution: (1) these points are taken randomly on the random mesh. (2) these points are taken randomly on the finite element mesh. The results highlighted in bold are the ones that provide satisfying predictions visually and numerically.

### 3.4.2 Impact of noisy measurements:

In realistic circumstances, the measurements captured from the sensors are always corrupted with some noise. We investigate here the impact of noisy measurements on the quality of PINNs prediction. To this end, we only consider the case where there are two lines of sensors that are put at the input and output of the calender (case 2). We generate the noise as uncorrelated Gaussian noise for the supervised data. We vary the level of noise to verify the effectiveness of PINNs when dealing with noisy measurements. PINNs configuration using KNNs-tanh is the same as the cases before except that now we train with more epochs. More precisely, we use Adam optimizer with 50,000 epochs with the learning rate $lr = 10^{-3}$, 200,000 epochs with the learning rate $lr = 10^{-4}$ and 300,000 epochs with $lr = 10^{-5}$. Table 3 illustrates in more detail the quality of PINNs prediction in terms of relative $L^2$ errors with the reference solution. These errors are evaluated on the finite element mesh. It is clear that the accuracy of PINNs prediction remains robust even when the supervised data are corrupted with 10% noise. Figure 12 shows the visualization of the reference solution and PINNs prediction at the horizontal line $y = 0$ and at the vertical line $x = -6.5$. We can see that at the line $y = 0$, the visual prediction for $u_y$ does not fit perfectly the reference solution. However, when we visualize the results at the line $x = -6.5$, which locates very close to the input of the calender, the predictions for $u_y$ fit accurately the reference solution with comparable errors to other fields. On this line, we see that the visual predictions for $T$ and $p$ are not accurate. Even though these errors are very small compared to the scale of $T$ and $p$, this behavior is somehow expected since we do not have many collocation points at the input of the calender. As a result, the predictions for all fields in this area are not as accurate as the ones at the output.

| Noise Level | $\epsilon_T$ | $\epsilon_{u_x}$ | $\epsilon_{u_y}$ | $\epsilon_p$ |
|-------------|--------------|-----------------|-----------------|--------------|
| 0% noise    | 3.24 | 4.63 | 4.87 | 0.83 |
| 1% noise    | 4.16 | 5.73 | 6.16 | 1.07 |
| 5% noise    | 4.55 | 6.94 | 6.57 | 1.64 |
| 10% noise   | 5.60 | 7.21 | 7.16 | 1.69 |

Table 3: Relative $L^2$ errors between the reference solution and the predictions by PINNs with different level of noise.
We first employ and compare vanilla PINNs with PINNs using locally adaptive activation function and deep Kronecker
with Rowdy Net offer significantly faster convergence and better training loss than other models. In addition, we note
where \( \text{Eq. (4)} \) but we dispose of some measurements for the temperature and the velocity (which may be noisy) from the
to this end, we suppose that we do not know the value of the thermal conductivity \( \lambda \) during the training process, except L-LAAFs, other models are very sensitive with high learning rates, especially
3.5 Identifying unknown physical parameters
In the industrial context, it is often the case that some physical parameters are not defined but we observe some
measurements of the solution. These problems are referred to as inverse problems, which are difficult to be solved in
practice using traditional methods. In this section, we aim to demonstrate the capability of PINNs of identifying an
unknown physical parameter (namely the thermal conductivity of the rubber \( \lambda \)) from some observed measurements.
To this end, we suppose that we do not know the value of the Péclet number and Brinkman number in the
dimensionless PDE. Hence, we use PINNs to define these two unknown parameters in the dimensionless PDE and then
deduce the value of the conductivity. We note that, in this case, the only considered PDE is the equation of conservation
of energy (Eq. (7)) as the conductivity only appears in this equation. We study the impact of the sensors’ location and
also the impact of noisy measurements on PINNs identification capability.

In identification problems, the unknown parameters (Péclet number and Brinkman number in our case) are considered
as network parameters that are learned during the training process. As for the case before, we use PINNs with the
spatial coordinates \( x, y \) as inputs and \( T, p, u_x, u_y \) as outputs and minimize the following cost function:

\[
L = L_{\text{data}} + L_{\text{pde}}
\]

\[
= \frac{\omega_T}{N_T} \sum_{i=1}^{N_T} (\hat{T}^i - T^i)^2 + \frac{1}{N_U} \sum_{i=1}^{N_U} \left( \omega_{u_x} (\hat{u}_x^i - u_x^i)^2 + \omega_{u_y} (\hat{u}_y^i - u_y^i)^2 \right) + \frac{1}{N_f} \sum_{i=1}^{N_f} \omega_4 e_4^2
\]

where \( e_4 \) is the dimensionless PDE residual corresponding to Eq. (7). The weight coefficients are fixed as follows:

\[
\omega_T = \frac{N_T}{\sum_{i=1}^{N_T} (T^i)^2}, \quad \omega_{u_x} = \frac{N_U}{\sum_{i=1}^{N_U} (u_x^i)^2}, \quad \omega_{u_y} = \frac{N_U}{\sum_{i=1}^{N_U} (u_y^i)^2}
\]

and \( \omega_4 = 1 \). We note that here, no information on the boundary is given.

We first employ and compare vanilla PINNs with PINNs using locally adaptive activation function and deep Kronecker
networks. For the training of PINNs, the number of measurements captured from the sensors for the temperature and
the velocity is \( N_T = N_U = 500 \). The number of collocation points is \( N_f = 10,000 \) and these points are randomly
taken from the finite element mesh. To minimize the cost function, we use Adam optimizer with 50,000 epochs with the
learning rate \( lr = 10^{-3} \), 100,000 epochs with the learning rate \( lr = 10^{-4} \) and 150,000 epochs with \( lr = 10^{-5} \). For
L-LAAFs and N-LAAFs, we fix the scaling factor \( n = 1 \) and initialize the adaptive parameters \( a^h = 1 \) or \( a^k = 1 \). For
Rowdy Net and KNNS-tanh, we fix \( K = 2 \) and \( \alpha^h = \omega^h = 1 \) and the scaling factor \( n = 1 \) (for Rowdy Net). Figure 13
shows the cost function during the training process for different cases of sensor location. In all cases, we see that PINNs
with Rowdy Net offer significantly faster convergence and better training loss than other models. In addition, we note
that during the training process, except L-LAAFs, other models are very sensitive with high learning rates, especially

Figure 12: Reference solution and PINNs prediction for the physical fields (a-d) at the line \( y = 0 \) and (e-h) at the line
\( x = -6.5 \).
the models with Rowdy Net and KNNS-tanh, and then become more stable with smaller learning rate. Table 4 illustrates in more detail the performance of each approach in terms of relative errors in case 1 of sensor location. Again, we see clearly that the models with KNNS-tanh and N-LAAFs perform slightly better than the vanilla PINNs while the model with Rowdy Net outperforms the others. We can increase the performance of Rowdy Net and KNNS-tanh by increasing the number of terms $K$. However, we observe that with $K = 2$, these models already offer satisfying results.

Next, we investigate the impact of the location of sensors on the performance of PINNs. To this end, we use the same configuration as before and PINNs with Rowdy Net, which is proven to be more efficient than other models as shown in Figure 13, to identify the unknown parameters in all cases.

![Figure 13: Cost function during training process with different approaches in inverse problem.](image)

![Table 4: Relative error (in %) between reference values and PINNs prediction with different approaches in case 1.](image)

| Approach      | $\epsilon_{1}/P_c$  | $\epsilon_{Br}/P_c$ | $\epsilon_{\lambda}$ |
|---------------|----------------------|----------------------|-----------------------|
| Vanilla       | 22.5 ± 0.51          | 11.2 ± 0.41          | 0.85 ± 0.02           |
| L-LAAFs       | 41.6 ± 0.30          | 23.3 ± 0.33          | 1.49 ± 0.01           |
| N-LAAFs       | 16.0 ± 0.42          | 7.45 ± 0.04          | 0.61 ± 0.03           |
| Rowdy Net     | 2.18 ± 0.04          | 0.09 ± 0.02          | 0.04 ± 0.00           |
| KNNS-tanh     | 19.3 ± 0.32          | 8.16 ± 0.05          | 0.74 ± 0.03           |

Table 4: Relative error (in %) between reference values and PINNs prediction with different approaches in case 1.

![Figure 14: Prediction for unknown parameters during the training process while considering different cases of the location of sensors. The small plots show the results at the last 100,000 epochs.](image)

![Table 5: Prediction for unknown parameters at the end of PINNs training process.](image)

| Case     | $1/P_c$     | $Br/P_c$    | $\lambda$    |
|----------|-------------|-------------|--------------|
| High-fidelity model | 4.00e-03   | 1.20e-01   | 2.00e+01    |
| Case 1   | 3.94e-03   | 1.20e-01   | 1.99e+01    |
| Case 2   | 3.81e-03   | 1.16e-01   | 1.96e+01    |
| Case 3   | 3.89e-03   | 1.19e-01   | 1.98e+01    |

Table 5: Prediction for unknown parameters at the end of PINNs training process.
Figure 14 shows the prediction values for $1/Pe, Br/Pe$ and $\lambda$ during the process. We note that the value of $\lambda$ is inferred from the values of $1/Pe$ and $Br/Pe$. In all cases, the predictions of PINNs for PDEs parameters converge to the true values. Table 5 presents the final prediction for these parameters at the end of the training process. We see that the case 1 (Figure 6a) and case 3 of the location of sensors (Figure 6c) provide a slightly better performance than the case 2 (Figure 6b). However, it is remarkable that in case 2, with only two lines of sensors at the input and output of the calender, PINNs are still capable of approximating the unknown parameters with high accuracy.

Next, we investigate the impact of noisy measurements on the performance of PINNs. We generate the noise as uncorrelated Gaussian noise. We vary the number of supervised points and the level of noise to verify the effectiveness of PINNs when dealing with noisy measurements. We use the same configuration as the case before for PINNs with Rowdy Net, however, we increase the number of epochs for the training process. More precisely, we use Adam optimizer with 50,000 epochs with the learning rate $lr = 10^{-3}$, 100,000 epochs with the learning rate $lr = 10^{-4}$ and 200,000 epochs with $lr = 10^{-5}$.

Figure 15 illustrates the cost function during the training process for different level of noisy measurements. We see that the more the level of noise increases, the more the model has difficulties in minimizing the cost function. This is somehow expected since with the effect of noise, the PDEs constraints are less respected by the supervised measurements. Table 6 summarizes the performance of PINNs for case 3 in terms of prediction values compared to reference values. We see that, even though the accuracy decreases when the level of noise increases, PINNs are still able to approximate the unknown parameters with high accuracy even when the training data are corrupted with 10% noise.

|            | $1/Pe$ | $Br/Pe$ | $\lambda$ |
|------------|--------|---------|-----------|
| High-fidelity model | 4.00e-03 | 1.20e-01 | 2.00e+01 |
| 0% noise    | 3.94e-03 | 1.19e-01 | 1.98e+01 |
| 1% noise    | 3.89e-03 | 1.18e-01 | 1.97e+01 |
| 5% noise    | 3.81e-03 | 1.16e-01 | 1.96e+01 |
| 10% noise   | 3.77e-03 | 1.16e-01 | 1.94e+01 |

Table 6: Prediction for unknown parameters at the end of PINNs training process while considering different levels of noisy measurements in supervised data in case 3.

4 Conclusion

In this study, we demonstrated the interest of using Physics-Informed Neural Networks (PINNs) for the modeling of the rubber calendering process. As far as our knowledge, this is the first time PINNs are applied to solve a non-Newtonian fluid thermo-mechanical problem. PINNs have shown a great capability to infer all the physical fields without using any supervised data for the velocity and the pressure fields but taking some knowledge on the boundary for the velocity and the temperature. We investigated the methods using locally adaptive activation functions (L-LAAFs and N-LAAFs) and deep Kronecker networks (Rowdy Net and KNNs-tanh). The results showed that these methods help to improve significantly the accuracy of PINNs predictions. We demonstrated an important impact of the placement of the sensors (or the position of supervised data) on the prediction quality of PINNs. We investigated the effect of the distribution of collocation points on the prediction quality. We concluded that training PINNs with the collocation points that were taken randomly on a finite element mesh (which provides an a priori knowledge on high gradient location) provides much better models than training with the collocation points that were taken randomly inside the domain. We also...
tested the robustness of PINNs when dealing with noisy measurements. The results demonstrated that when training with sufficient epochs, PINNs are capable of predicting the reference solution with the same accuracy as when using high-fidelity data. Concerning the identification problem, we showed that with any case of the location of sensors, PINNs are able to approximate the unknown thermal conductivity. We also demonstrated that even when the supervised data are corrupted with noise, the quality of PINNs prediction still remains accurate as expected.

Through our work, it can be seen that there are many challenges and opportunities for further research. We see that, in the cost function, we used pre-specified weight coefficients to achieve a good performance in PINNs. However, these pre-specified coefficients were obtained through trial and error method, and they may give different performances if there are any small changes in the geometry or the supervised measurements. Thus there is a need for an adaptive method that is able to provide the best values for these coefficients after the training process [McCleny and Braga-Neto][2020, Wang et al.][2020a,b]. Besides that, we observed that in some cases, PINNs are able to give a satisfying performance qualitatively, but however still provide a very high relative error compared to the reference solution. This suggests to examine the accuracy of PINNs on more appropriate quantities which present some expected physical properties. We can think of methods of model reduction such as Proper Orthogonal Decomposition (POD) [Berkooz et al.][1993] or Dynamic Mode Decomposition (DMD) [Schmid][2010] to extract the most important modes (in terms of energy or dynamic), and then calculate the error of PINNs on these modes. Another perspective would be to combine PINNs and reduced-order methods to improve learning efficiency. We aim to purchase these lines of research in our future work.

Acknowledgements

T.N.K. Nguyen is funded by Michelin and CEA through the Industrial Data Analytics and Machine Learning chair of ENS Paris-Saclay. Part of this work has been funded by Region Ile-de-France. Part of the computations has been executed on Atos Edge computer. In addition, the authors would like to thank Christophe Millet for fruitful discussions and the anonymous reviewers whose comments and suggestions helped improve and clarify significantly this paper.

Conflict of interest

The authors declare no conflict of interest.

References

Yuexing Bai, Temuer Chaolu, and Sudao Bilige. The application of improved physics-informed neural network (ipinn) method in finance. *Nonlinear Dynamics*, 107(4):3655–3667, 2022.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18, 2018.

Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *arXiv preprint arXiv:2105.09506*, 2021a.

Shengze Cai, Zhicheng Wang, Frederik Fuest, Young Jin Jeon, Callum Gray, and George Em Karniadakis. Flow over an espresso cup: inferring 3-d velocity and pressure fields from tomographic background oriented schlieren via physics-informed neural networks. *Journal of Fluid Mechanics*, 915, 2021b.

Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Karniadakis. Physics-informed neural networks (pinns) for heat transfer problems. *Journal of Heat Transfer*, 2021c.

Han Gao, Luning Sun, and Jian-Xun Wang. Phygeonet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, 428:110079, 2021.

Raghav Gnanasambandam, Bo Shen, Jinhoon Chung, Xubo Yue, et al. Self-scalable tanh (stan): Faster convergence and better generalization in physics-informed neural networks. *arXiv preprint arXiv:2204.12589*, 2022.

R Guénette, A Fortin, J Labbé, and JP Marcotte. Iterative solvers for quadratic discretizations of the generalized stokes problem. *International journal for numerical methods in fluids*, 44(7):695–720, 2004.

QiZhi He, David Barajas-Solano, Guzel Tartakovsky, and Alexandre M Tartakovsky. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources*, 141:103610, 2020.
Physics-informed neural networks for non-Newtonian fluid thermo-mechanical problems: an application to rubber calendering process

Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Zhiwei Fang, Max Rietmann, Wonmin Byeon, and Sanjay Choudhry. Nvidia sinmet™: An ai-accelerated multi-physics simulation framework. In International Conference on Computational Science, pages 447–461. Springer, 2021.

Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Communications in Computational Physics, 28(5):2002–2041, 2020.

Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. Proceedings of the Royal Society A, 476(2239):20200334, 2020a.

Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. Journal of Computational Physics, 404:109136, 2020b.

Ameya D Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. Computer Methods in Applied Mechanics and Engineering, 365:113028, 2020c.

Ameya D Jagtap, Dimitrios Mitsotakis, and George Em Karniadakis. Deep learning of inverse water waves problems using multi-fidelity data: Application to serre–green–naghdi equations. Ocean Engineering, 248:110775, 2022a.

Ameya D Jagtap, Yeonjong Shin, Kenji Kawaguchi, and George Em Karniadakis. Deep kronecker neural networks: A general framework for neural networks with adaptive activation functions. Neurocomputing, 468:165–180, 2022b.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. Journal of Computational Physics, 426:109951, 2021.

Georgios Kissas, Yibo Yang, Eileen Hwuang, Walter R Witschey, John A Detre, and Paris Perdikaris. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering, 358:112623, 2020.

Université Laval and GIREF. Mef++, 2021. URL https://fr.wikipedia.org/wiki/MEF%2B%2B

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895, 2020.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. SIAM Review, 63(1):208–228, 2021.

Mohammadamin Mahmoudabadbozchelou, George Em Karniadakis, and Safa Jamali. nn-pinns: Non-newtonian physics-informed neural networks for complex fluid modeling. Soft Matter, 18(1):172–185, 2022.

Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. Computer Methods in Applied Mechanics and Engineering, 360:112789, 2020.

Stefano Markidis. The old and the new: Can physics-informed deep-learning replace traditional linear solvers? Frontiers in big Data, page 92, 2021.

Levi McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. arXiv preprint arXiv:2009.04544, 2020.

Levi D McClenny, Mulugeta A Haile, and Ulisses M Braga-Neto. Tensordiffeq: Scalable multi-gpu forward and inverse solvers for physics informed neural networks. arXiv preprint arXiv:2103.16034, 2021.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. arXiv preprint arXiv:2001.04385, 2020.

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.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science, 367(6481):1026–1030, 2020.

Chengping Rao, Hao Sun, and Yang Liu. Physics-informed deep learning for incompressible laminar flows. Theoretical and Applied Mechanics Letters, 10(3):207–212, 2020.

Brandon Reyes, Amanda A Howard, Paris Perdikaris, and Alexandre M Tartakovsky. Learning unknown physics of non-newtonian fluids. Physical Review Fluids, 6(7):073301, 2021.
Physics-informed neural networks for non-Newtonian fluid thermo-mechanical problems: an application to rubber calendering process

Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

Khemraj Shukla, Ameya D Jagtap, James L Blackshire, Daniel Sparkman, and George Em Karniadakis. A physics-informed neural network for quantifying the microstructural properties of polycrystalline nickel using ultrasound data: A promising approach for solving inverse problems. *IEEE Signal Processing Magazine*, 39(1):68–77, 2021a.

Khemraj Shukla, Ameya D Jagtap, and George Em Karniadakis. Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics*, 447:110683, 2021b.

Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Luning Sun and Jian-Xun Wang. Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoretical and Applied Mechanics Letters*, 10(3):161–169, 2020.

Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

Alexandre M Tartakovsky, C Ortiz Marrero, Paris Perdikaris, Guzel D Tartakovsky, and David Barajas-Solano. Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, 56(5):e2019WR026731, 2020.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536*, 2020a.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *arXiv preprint arXiv:2007.14527*, 2020b.

Kaichao You, Mingsheng Long, Jianmin Wang, and Michael I Jordan. How does learning rate decay help modern neural networks? *arXiv preprint arXiv:1908.01878*, 2019.

Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

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.

A Validation of PINNs with analytical solution

In this section, we provide the comparison of PINNs and the analytical solution in a pipe (Figure 16), using the same Stokes equation but without thermal coupling. The considered PDE is written as follows:

\[
\nabla \cdot (2\eta(\vec{u}) \mathbf{\epsilon}(\vec{u})) - \nabla p = 0 \\
\n\nabla \cdot \vec{u} = 0
\]

where the generalized strain rate \(\gamma(\vec{u}) = \sqrt{2 \sum_{i,j} \mathbf{\epsilon}_{i,j}^2}\) and the dynamic viscosity: \(\eta(\vec{u}) = K|\gamma(\vec{u})|^n - 1\). We set the boundary condition as follows: the velocity of the fluid at the wall is supposed to be zero (no-slip condition), and the input flow rate \(Q\) is fixed.

To calculate the analytical solution, we use the cylindrical coordinates \((r, \theta, z)\). We suppose that the flow is axisymmetric \(\frac{\partial}{\partial \theta} = 0\), and we search for a solution \(\vec{u} = (u_r, u_\theta, u_z)\) where \(u_r = 0, u_\theta = 0\).
By re-writing the PDE in cylindrical coordinates, the equation of conservation of mass gives us:

\[ \nabla \cdot \mathbf{u} = 0 \Leftrightarrow \frac{\partial u_z}{\partial z} = 0 \]

And the equation of conservation of energy gives us:

\[ \nabla \cdot (2\eta(\mathbf{u})\mathbf{e}(\mathbf{u})) - \nabla p = \mathbf{0} \]

\[ \Leftrightarrow \begin{cases} \frac{\partial p}{\partial r} = 0 \\ \frac{\partial p}{\partial \theta} = 0 \\ \frac{1}{r} \frac{\partial}{\partial r} \left( \frac{r \eta(\mathbf{u})}{r} \frac{\partial u_z}{\partial r} \right) - \frac{\partial p}{\partial z} = 0 \end{cases} \]

In the last obtained equation, by taking the integral three times and using the boundary condition \( u_z = 0 \) at \( r = R \), we get:

\[ u_z = \left( -\frac{\Delta P}{LK} \right)^{1/n} \frac{n}{n+1} (R^{n+1} - r^{n+1}) \]

To calculate \( u_z \) explicitly, we use the analytical formulation of the pressure difference along the pipe length:

\[ \Delta P = -2 \left( \frac{Q}{\pi R^3} \right)^n L \frac{K}{R} \]

where \( L \) is the length of the pipe and \( Q \) the flow rate.

We now use PINNs to solve this problem by assuming that the boundary condition is unknown but we dispose of the solution at some points in the pipe. The goal is to infer the pressure and the velocity at all points in the domain. Due to the symmetry of the problem, we only consider a quarter of the pipe.
We fix the number of supervised points for the velocity as $N_U = 5,000$ and the number of collocation points is $N_f = 20,000$. We suppose that both the supervised points and collocation points are randomly distributed in the pipe. We use a feedforward network of 5 layers and 100 neurons in each layer. To minimize the cost function, we use Adam optimizer with 20,000 epochs with the learning rate $lr = 10^{-3}$. The cost function is defined similarly to the cost function in section 3.4. Figure 17 presents the comparison of the analytical solution and PINNs prediction at the plan $z = cst$. The relative $L^2$ errors at all points in the pipe for $u_z$ and $P$ are 2.59e-02 and 1.87e-02, respectively.