UNIVERSITÀ DEGLI STUDI DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

Dipartimento di Ingegneria Civile, Edile e Ambientale (ICEA)

CORSO DI DOTTORATO DI RICERCA IN:

SCIENZE DELL'INGEGNERIA CIVILE ED AMBIENTALE

CURRICULUM:

MATERIALI, STRUTTURE, SISTEMI COMPLESSI E ARCHITETTURA

XXXVI CICLO

Physics-Informed Neural Network in porous media and epidemiological applications

Coordinatore: Ch.mo Prof. Massimiliano Ferronato

Supervisore: Ch.mo Prof. Massimiliano Ferronato

Co-Supervisore: Dr. Eng. Nicolò Spiezia

Dottorando: Caterina Millevoi

UNIVERSITÀ DEGLI STUDI DI PADOVA

Abstract

Department of Civil, Architectural and Environmental Engineering

Doctor of Philosophy

Physics-Informed Neural Network in porous media and epidemiological applications

Caterina Millevoi

This doctoral thesis delves into the application of *Physics-Informed Neural Network* (PINN) across diverse domains, notably in the field of porous media and epidemiology. The goal is to analyze the potential of PINNs in solving complex problems, for the purpose of integrating them with well-established methods and enhancing the capability of obtaining reliable modeling predictions. The study aims at:

- utilizing PINN for forward solution modeling and parameter estimation in hydro-poromechanics,
- extending PINN capabilities to track the temporal changes in the model parameters and provide an estimate of the model state variables in epidemiological modeling.

With the need for robust, computationally efficient tools in the field of application, PINNs emerge as a promising tool to bridge traditional physics-based modeling and modern machine learning approaches.

In the hydro-poromechanics domain, the study reveals insights into effective neural network architectures through a sensitivity analysis, shedding light on significant hyper-parameters and network complexities crucial for efficient PINN training. Additionally, a *sensor-driven* approach is introduced to accelerate convergence and enhance accuracy by integrating field data automatically during the training. The proposed method showcases promising results in real-world applications, where combining some data measured in the site can help to account for marginal effects due to several minor dynamics. The thesis also tackles the inverse problem of parameter identification in Biot's model, analyzing the potential of PINNs in estimating key geomechanical and hydraulic properties of subsurface materials.

Shifting focus to epidemiology, the work introduces novel approaches to enhance PINN applications for simulating the spread of epidemics, based on the solution of *Susceptible-Infectious-Recovered* (SIR) based epidemiological models, and estimating time-dependent transmission rates. In this context, a *split* PINN approach, involving a two-step training process, is proposed. The method proves to be a computationally efficient alternative, outperforming the traditional training of PINNs in terms of both accuracy and speed. A reduction of the SIR model is also presented, which limits the number of unknown functions and loss terms. Application to synthetic and real-world data from the Italian COVID-19 epidemic highlights the adaptability of PINNs in capturing system dynamics, showcasing improved accuracy in estimating critical time-dependent parameters and modeling the process with respect to the traditional approach.

This interdisciplinary study underscores the versatility of PINNs, providing a framework for assisting traditional methods in modeling coupled flow-deformation processes in porous media and epidemiological investigations, where the integration of data series with traditional differential models is crucial. The findings of this thesis work aim at contributing to advancements to the application of PINNs in hydro-poromechanics and epidemiology, and open avenues for future research, with the goal of combining the potential of deep learning in conjunction with physics-based models to advance predictive capabilities in complex systems.

UNIVERSITÀ DEGLI STUDI DI PADOVA

Sommario

Dipartimento di Ingegneria Civile, Edile e Ambientale

Dottorato di Ricerca

Physics-Informed Neural Network in porous media and epidemiological applications

Caterina Millevoi

Questa tesi di dottorato approfondisce l'applicazione delle *Physics-Informed Neural Network* (PINN) in diversi ambiti, in particolare nel campo dei mezzi porosi e dell'epidemiologia. L'obiettivo è analizzare il potenziale delle PINN nella risoluzione di problemi complessi, col fine di integrarle ai metodi tradizionali per ottenere modelli affidabili in grado di effettuare analisi e previsioni. Lo studio si propone di:

- fare uso delle PINN per approssimare le variabili di interesse e stimare i parametri significativi in poromeccanica,
- estendere la capacità delle PINN di tracciare i cambiamenti temporali nei parametri e nelle variabili di stato dei modelli epidemiologici.

Data la necessità di disporre di strumenti robusti ed efficienti dal punto di vista computazionale in campo applicativo, le PINN emergono come uno strumento promettente per fondere i modelli tradizionali basati sulla fisica con i più moderni approcci di apprendimento automatico.

Nell'ambito dei mezzi porosi viene esaminato, attraverso un'analisi di sensibilità, l'impatto dell'architettura delle reti neurali nella soluzione tramite PINN del modello poroelastico formulato da Biot, facendo luce sugli iperparametri significativi per un addestramento efficiente delle PINN. Inoltre, viene introdotto un approccio chiamato *sensor-driven* per accelerare la convergenza e migliorare l'accuratezza del modello PINN, aggiungendo misurazioni disponibili al training delle reti neurali. I risultati ottenuti sono promettenti per le applicazioni del mondo reale, dove l'integrazione di alcuni dati raccolti sul campo può aiutare a tenere conto degli effetti marginali dovuti a fenomeni minori. La tesi affronta in questo ambito anche il problema inverso, ovvero l'identificazione dei parametri nel modello di Biot, analizzando il potenziale delle PINN nello stimare le principali proprietà geomeccaniche e idrauliche dei materiali del sottosuolo.

In campo epidemiologico, il presente lavoro di tesi introduce nuovi approcci per migliorare l'applicazione delle PINN nella simulazione della diffusione di epidemie, basandosi sul modello compartimentale SIR per stimarne la velocità di trasmissione. In questo contesto, viene proposto un approccio *split*, che prevede un processo di addestramento delle reti in due fasi. Tale metodo risulta un'alternativa efficiente dal punto di vista computazionale, superando il tradizionale addestramento delle PINN sia in termini di accuratezza, che di velocità. Viene inoltre presentata una forma ridotta del modello SIR, che consente di diminuire il numero di funzioni incognite e di termini della funzione costo. Le performance dell'approccio proposto vengono investigate e discusse in casi test sia con dati sintetici che con dati reali della sorveglianza dell'epidemia di COVID-19 in Italia. I risultati evidenziano la capacità delle PINN di approssimare le dinamiche del processo epidemiologico e dimostrano una maggiore precisione sia nella stima dei parametri dipendenti dal tempo, che nella modellizzazione dei contagi rispetto all'approccio tradizionale.

Questo studio interdisciplinare sottolinea la versatilità delle PINN, fornendo un inquadramento di questo strumento col fine di integrare la modellizzazione di problemi accoppiati flusso-deformazione in mezzi porosi e di indagini epidemiologiche, in cui svolge un ruolo fondamentale l'integrazione di serie di dati con approcci differenziali classici. I risultati oggetto del lavoro di tesi vogliono contribuire al progredire dell'applicazione delle PINN nell'ambito dei mezzi porosi nonché dell'epidemiologia, gettando le basi per la ricerca futura, con l'obiettivo di combinare il potenziale del *Deep Learning* (DL) con modelli basati sulla fisica per migliorare le capacità predittive nella modellizzazione di sistemi complessi.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Deep Learning (DL), a subfield of *Machine Learning* (ML), has gained success in solving complex problems by utilizing artificial neural networks with multiple layers in different applications, ranging from image recognition to natural language processing [Russakovsky et al., 2015; Young et al., 2017]. At its core, DL involves training a *Neural Network* (NN) on vast amounts of data, enabling it to discern intricate patterns and make predictions. This capability has strongly impacted on the way several complex problems are approached, allowing for significant accuracy and efficiency in tasks where huge quantities of data are available [Sejnowski, 2018].

In recent years, the intersection of DL and traditional physics-based modeling has led to important advancements, paving the way for innovative approaches [Karniadakis et al., 2021]. *Physics-Informed Neural Network* (PINN) combines principles of physics with DL, leveraging NNs to approximate solutions to physical equations and handle real-world applications [Raissi et al., 2019]. PINNs have introduced a novel paradigm that integrates domain-specific knowledge with machine learning methodologies. PINNs leverage the power of NNs to approximate solutions to partial differential equations, effectively combining data-driven insights with the governing laws of physics. Indeed, the loss function is constructed to include physical constraints, incorporating information from both data points and physics equations. This amalgamation can not only enhance the predictive capabilities of models but also enable the incorporation of physical principles into the learning process. The practice to approximate unknown functions with NNs offers flexibility and adaptability, and the use of automatic differentiation to compute their derivatives in a continuous way minimizes the propagation of uncertainties. PINNs find application across diverse scientific and engineering domains, ranging from blood flow simulations [Sun et al., 2020] and turbulent flows [Mathews et al., 2021; Xiao et al., 2020] to optics [Chen et al., 2020b; Wiecha et al., 2021], molecular dynamics [Stielow and Scheel, 2021; Islam et al., 2021], geoscience [Li et al., 2021; Smith et al., 2021], and industrial processes [Yucesan and Viana, 2021]. The methodology proves versatile, addressing both forward and inverse problems in these domains, and it has the potential to become an important tool in safety-critical scenarios, digital twin applications, and multi-scale modeling.

Numerous challenges necessitate a multidisciplinary approach, where insights from diverse fields converge to address complex problems. This thesis discusses the application of PINNs in the context of hydro-poromechanics and epidemiological modeling, endeavoring to contribute to two critical domains combined by the interplay between dynamic processes and difficult-to-model phenomena. The overarching goal is examining the potential of PINN-based formulations to support the modeling of coupled flow-deformation problems in porous media and epidemiological investigations, where data integration with traditional differential approaches is crucial. The specific objectives include:

- Formulate PINN tailored to hydro-poromechanical systems and epidemiological processes.
- Investigate the efficacy of PINNs in accurately estimating parameters in the dynamic and complex contexts of both hydro-poromechanics and epidemiology.
- Apply the developed models to synthetic test cases and real-world scenarios, encompassing subsurface processes and the spread of infectious diseases within populations.

In the field of hydro-poromechanics, which includes among others applications in the geosciences and biomedicine, the coupled simulation of processes like fluid flow and solid deformation in porous media is crucial. Well-established numerical methods, such as *Finite Element* (FE) and *Finite Difference* (FD), have been widely used [Lewis and Schrefler, 1998], but PINNs can offer an alternative (or support) by integrating physics-based modeling and machine learning. Within the context of coupled hydro-poromechanics, the thesis investigates the application of PINNs for both forward and inverse modeling. Since advances in this field can have broad implications for resource management and environmental sustainability, just to mention a few, we aim at building a PINN-based model for hydro-poromechanical applications governed by Biot's poroelasticity equations which is essential to many scientific fields, such as hydrogeology [Verruijt, 1969; Gambolati et al., 2000; Gambolati and Teatini, 2015], petroleum engineering [Detournay and Cheng, 1993], soil mechanics [Verruijt, 2018], biomechanics [Mow et al., 1980], rock mechanics [Guéguen et al., 2004; Castelletto et al., 2010], and theoretical mechanics [Cheng, 2016] in civil engineering. We focus on the use of the poroelasticity model in subsurface engineering for energy resources. PINN training for multi-physics problems is demanding: challenges include convergence, stability, and the impact of neural network architecture on solution accuracy. Investigating various configurations, including NN architectures, loss development, and training procedures is of central importance [Cuomo et al., 2022]. Therefore, a sensitivity analysis of the architecture of PINNs is performed and a sensor-driven hybrid approach is proposed, which integrates data within the PINN framework aiming to speed up the training times, reduce coupled problem solution complexity, and enhance neural network efficiency [Millevoi et al., 2021]. Moreover, PINNs are introduced as a tool for inverse modeling, providing estimations of material parameters, which are of paramount importance for the characterization of the subsurface and the consequent approximation of its behavior. The deterministic approach offered by PINNs is the core of the matter, given by their capability to bridge physics-based modeling and machine learning. The method is applied on different test cases, with a special focus on heterogeneous problems, that entail discontinuous derivatives of the quantities of interest [Millevoi et al., 2023b]. The results consent us to highlight the strengths and the weaknesses of the approach and propose possible solution to the challenges encountered.

Shifting focus to epidemiological models, the last part of the thesis discusses the role of PINNs, particularly in the context of the simulation of the COVID-19 pandemic evolution. Compartmental models, such as the *Susceptible-Infectious-Recovered* (SIR) model, provide simple and efficient tools to analyze the relevant transmission processes during an outbreak, to produce short-term forecasts or transmission scenarios, and to assess the impact of vaccination campaigns [Ogilvy and G., 1927]. However, their calibration is not straightforward, since many factors contribute to the rapid change of the transmission dynamics during an epidemic. For example, there might be changes in the individual awareness, the imposition of non-pharmacological interventions and the emergence of new variants [Albi et al., 2021; Giordano et al., 2021; Marca et al., 2022]. As a consequence, model parameters such as the transmission rate are doomed to change in time, making their assessment more challenging. In this context, we propose to use PINNs to track the temporal changes in the model parameters and provide an estimate of the model state variables. The ability of PINNs to identify unknown model parameters makes them particularly suitable to solve ill-posed inverse problems, such as those arising in the application of epidemiological models. Hence, we develop a reduced-split approach for the implementation of PINNs to estimate the temporal changes in the state variables - i.e. compartmental classes of the susceptible, infectious, and recovered population - along with the transmission rate - or the reproduction number - of an epidemic based on the SIR model equation and infectious data [Millevoi et al., 2023a]. The main idea is to split the training first on the epidemiological data, and then on the residual of the system equations. The proposed modifications to the PINN algorithm aim to enhance convergence and stability, especially in estimating time-dependent transmission rates.

Thesis structure

The remainder of the thesis is organized as follows:

- **Chapter 2.** This chapter provides a comprehensive survey of relevant literature in DL and introduces the concept of PINN. The key elements of neural networks are presented and an overview of the operating principles and the evolution of NNs is given, with a far from exhaustive review of the plenty of applications in the recent years. The chapter lays the groundwork for understanding the subsequent applications in hydro-poromechanics and epidemiological modeling.

- **Chapter 3.** Focusing on coupled hydro-poromechanics, this chapter explores the application of PINNs to describe processes in geomechanics and hydrogeology. The discussion includes an initial sensitivity analysis on the different hyperparameter setting defining the architecture of PINNs in this context. The obtained results are later applied for the forward solution of coupled Biot's model, suggesting a *sensor-driven* framework to improve the training trend and the prediction accuracy.

- **Chapter 4.** In this chapter, we focus on the task of estimating material parameters in hydro-poromechanical systems. We investigate the behavior of PINNs in parameter identification in several test cases, including the simulation of the consolidation in homogeneous and heterogeneous domains with or without water pumping installation.

- **Chapter 5.** Moving on to epidemiology, this chapter showcases the application of PINNs in understanding and predicting the spread of diseases. The SIR compartmental model has been considered and the chapter discusses the challenges posed by time-dependent transmission rates, presenting modifications to the PINN algorithm for faster convergence and stability. The proposed split training and reduced model are tested and compared with the standard approach in various setting, including a real-world application with data coming from the Italian COVID-19 surveillance.

- **Chapter 6.** The final chapter summarizes the key findings, discusses the contributions of this research, and outlines potential avenues for future exploration.

Chapter 2

Deep Learning Fundamentals

In the field of *Artificial Intelligence* (AI) - i.e. the capability of computers to mimic human intelligence way of thinking to execute tasks that need cognitive abilities [Russel and Norvig, 2022] - *Deep Learning* (DL) belongs to the *Machine Learning* (ML) subclass, which refers to an AI system ability to improve performance based on experience and learn on its own by extracting patterns from raw data. Indeed, while AI ("The science and engineering of creating intelligent machines" John McCarthy, 1956) can also be accomplished by explicitly programming a machine to tackle a specific problem stated by formal instructions, a formal characterization of the task at hand is not always possible. In this context, computers are not expressly taught to perform a certain activity, but instead learn via experience [Mitchell, 1997]. Arthur Samuel in 1959 announced the advantage of ML and stated that "programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort" [Samuel, 1959]. In particular, the core of DL is the organization of the computation path from inputs to outputs into many layers. The adjective *deep* mention the use of multiple layers - i.e. compositions - to progressively extract higher-level features from the raw input. For high-dimensional data, like images, DL has significant advantages over some ML techniques.

The origins of DL date back to 1943, when McCulloc and Pitts [McCulloch and Pitts, 1943] tried to model the networks of neurons in the brain, aiming at simulating the human way of thinking in order to build an intelligent entity that can compute how to act effectively and safely in a wide variety of novel situations. The structure they aimed to reproduce is the one of biological neurons, that are made up of a central body, called *soma* - which contains the nucleus and many of the more complex components -

some ramifications, called *dendrites*, and a longer extension, the *axon*. The fine terminal branches of the axon, called *telodendria*, end with the *synapses*, that can link to other neurons and send short electrical pulses - *signals*.

In this chapter we describe in detail the core of DL, that is *Neural Network* (NN), from the fundamental structures to the basic computations underpinning NNs. Some of the most popular architectures - such as *Feedforward Neural Network* (FNN), *Convolutional Neural Network* (CNN), and *Recurrent Neural Networks* (RNN) - are introduced to finish with *Physics-Informed Neural Network* (PINN), the protagonist of the whole study and applications of this Ph.D. dissertation. This allows us to provide an overview of the topic that contextualizes the thesis work embodied in the following chapters.

The main references for this chapter are: Russel and Norvig [2022]; Murphy [2012]; Haykin [1999]; Bose and Liang [1996].

2.1 Neural Networks

2.1.1 Basics of Artificial Neurons

Due to the link with biology, the basic unit of DL is called *Neural Network* (NN). In Goodfellow et al. [2016] a NN is described as a mathematical function that simulates the link between a set of input and some corresponding output values. This mathematical function is obtained by composing simple (nonlinear) functions and allows to learn complex feature hierarchies. NNs can be utilized both for *regression* task and for *classification* of data: in the former case the network provides a continuous output, while in the latter case the values produced are discrete. The goal is to exploit them to build a model by means of dataset of input-output pairs, so that the model can learn the correlation between the two and would be able to generalize to new data. The procedure is called *training* and will be described in detail later on (Section 2.2). The key concept of NNs is the need of data, that could come from measurements and investigations, or can be built ad hoc, for example by simulations.

Perceptrons

One of the simplest artificial NN is the *perceptron* [McCulloch and Pitts, 1943; Rosenblatt, 1958]. It models a biological neuron and serves as the fundamental building block for more complex NN architectures.

A perceptron takes multiple binary inputs (0 or 1) with an associated weight: it computes a weighted sum of these inputs and then applies an activation function to the sum, whose output is also binary. The output $\hat{y}$ of a perceptron can be represented as:

$$\hat{y} = \begin{cases} 1 & \text{if } \sum_j w_j x_j + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Where w_j are the weights associated with the inputs x_j, b is a bias term. Perceptrons are limited in their ability to model complex relationships in data and can only solve linearly separable problems. However, they were a foundational concept in the development of neural networks and served as a precursor to more advanced neuron models, which can handle more complex tasks.

Activation Functions

The output shape can be generalized through the introduction of a function, called *activation function*, that defines how the weighted sum of the input is transformed into an output from nodes in a layer of the network. The following notation, with ϕ the activation function, generalizes (2.1) and is the fundamental operation of DL, called *node*, or *(artificial) neuron*:

$$\hat{y} = \phi(\sum_j w_j x_j + b) \tag{2.2}$$

Hence a perceptron is an artificial neuron using the Heaviside step function as activation function.

Whereas at the beginning the role of the activation function was to simulate the behavior of biological neurons (that can be on or off), by adjusting to 0 (off) and 1 (on) the contributions that reach the neuron, later the main goal was to give nonlinearity to the model. This will improve considerably the expressivity of the model, as a two-layer neural network with a nonlinear activation can be proven to be a universal function approximator [Cybenko, 1989; Hornik et al., 1989]. This is known as the *Universal Approximation Theorem*.

In short, a node calculates the weighted sum of its inputs and then applies a nonlinear function to produce its output. The range of the activation function is generally restricted, so that the weight values are maintained low. Activation functions are also typically differentiable, given that NNs are frequently trained using the backpropagation of error algorithm (see Section 2.1.2), which needs the derivative of the prediction error to update the model weights. The most common activation functions are displayed in Figure 2.1. Note that all of them are monotonically non-decreasing, which implies that the derivative is $\phi' \neq 0$.

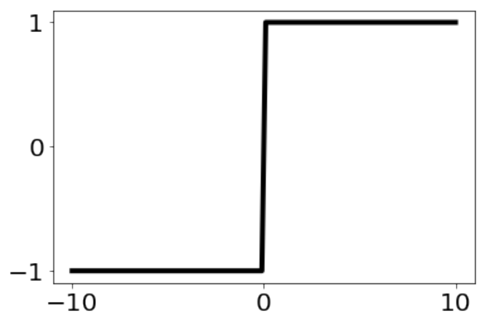
2.1.2 Building Blocks of Neural Networks

The foundational elements that serve as the bedrock upon which these intelligent systems are constructed are layers, weights and biases, forward propagation, and backward propagation. These building blocks not only define architecture and data flow of neural networks, but also yield a profound influence over their ability to perceive, learn, and take decisions.

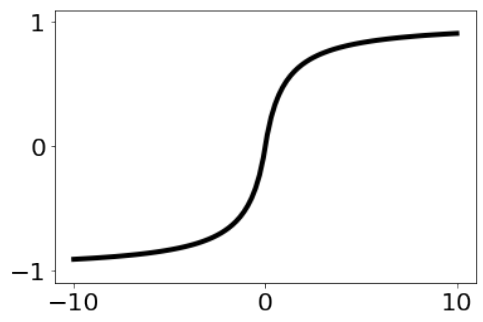
Layers

Layers form the structural framework of neural networks, akin to the layers of abstraction within the human brain. From the *input layer*, where data find their initial entry point, to the *hidden layers*, which manage complex transformations by taking input from a layer and pass output to another layer, and finally to the *output layer*, which delivers predictions, each layer has a unique role in information processing.

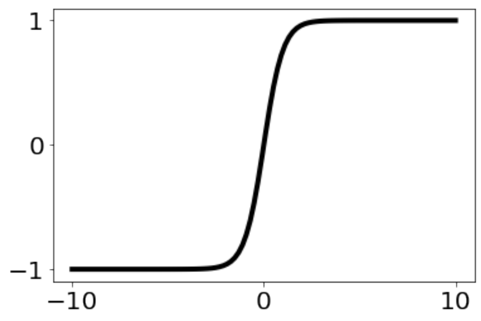
In the input layer samples are encoded as values of the input nodes. Depending on the nature of the inputs, different encodings and architectures are considered. When the values are boolean (True/False), they are processed as $0/1$, or $-1/1$. Categorical attributes are translated with one-hot encoding in vectors with 0 everywhere, except for the corresponding class entry. Numeric attributes are used as is (maybe scaled, see Section 2.6.2), and images generally as pixels matrices with the RGB notation.



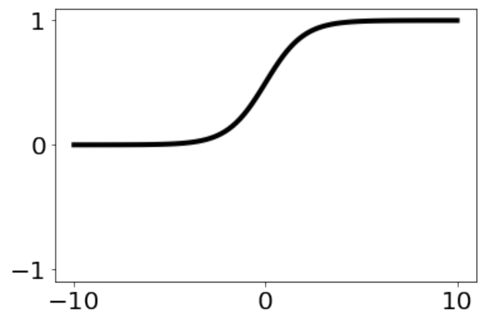
(A) sign



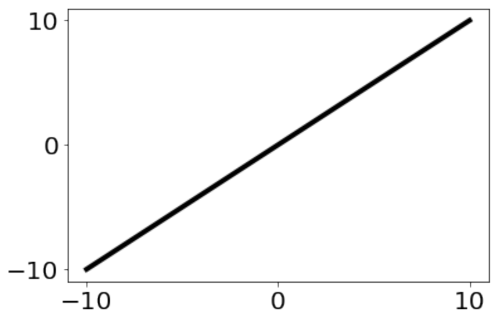
(B) softsign



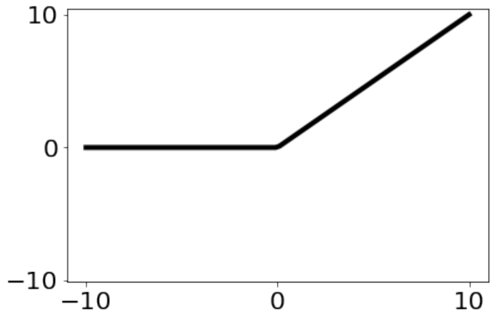
(C) tanh



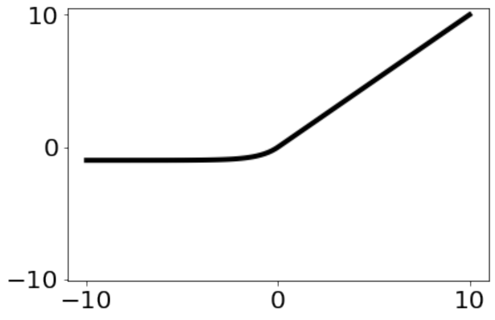
(D) logistic



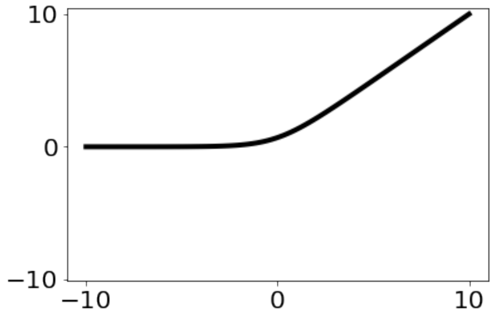
(E) linear



(F) ReLU



(G) elu



(H) softplus

FIGURE 2.1: Most common activation functions.

The output layer is where the prediction $\hat{y}$ is formulated to be later compared with the real output data y. The same encoding as for the inputs is applied to the datum y. For regression problems y is continuous, hence a linear output activation is the natural choice. In case of boolean classification, the output activation is usually a sigmoid (Figure 2.1) and $\hat{y}$ represents the probability that the example belongs to the positive class. The concept can be extended for multiclass classification with softmax (Figure 2.1) as output activation and so $\hat{y}$ the categorical distribution. Several other types of output may exist, but these three are the most typical.

In the hidden layers many halfway computations to process the input occur before producing the output. The values computed at each layer can be seen as different representations of the input. The composition of all these transformations succeeds - if all goes well - in transforming the input into the desired output. Common activations for hidden layers are sigmoid, $\tanh$, softplus and *Rectified Linear Unit* (ReLU) , all plotted in Figure 2.1. All hidden layers typically use the same activation function.

The number of nodes per hidden layer can vary, while in the input (output) layer there must be as many nodes as the input (output) dimension of the problem.

Figure 2.2 shows a *fully connected* neural network, i.e. a NN where each node is connected to all the nodes of the previous and following layers. The input layer is blue, while the output one is green. It has two hidden layers (in orange) with three nodes each (in yellow).

Weights and Biases

The weights and the biases are the parameters characterizing the neural network and they are trained through an optimization algorithm so as to minimize a loss function, that consists of a measure of the error of the approximation with respect to available data. These adjustable parameters determine the strength and direction of connections between neurons and are vital for learning.

Let n_l be the number of neurons of the l-th layer. The weights of layer l can be collected in the matrix $\mathbf{W}^{(l)} = (w_{kj}^{(l)}) \in \mathbb{R}^{n_l \times n_{l-1}}$, where index k corresponds to the outgoing node and index j to the incoming node. Hence, for example, the weight that links node 4 in the second layer with node 5 in the third layer is $w_{54}^{(3)}$.

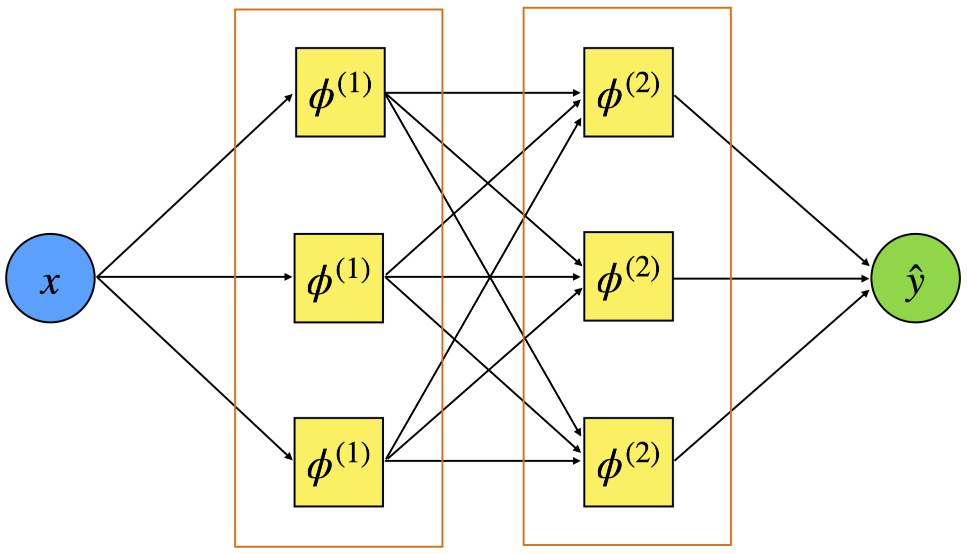


FIGURE 2.2: Fully connected *Feedforward Neural Network* with two hidden layers of three neurons each.

Forward Propagation

Consider a NN with L layers. Let $\mathbf{x}^{(1)} = (x_j^{(1)}) \in \mathbb{R}^{n_0}$ be the input of the problem with n_0 the dimension of the input. The input weighted sum of the k-th node in the first layer, for $k = 1, \ldots, n_1$, is:

$$z_k^{(1)} = \sum_j w_{kj}^{(1)} x_j^{(1)} + b_k^{(1)} \tag{2.3}$$

and the output is:

$$x_k^{(2)} = \phi^{(1)}(z_k^{(1)}) \tag{2.4}$$

By proceeding along the NN, the evolution of the signal at layer l can be described by the following *feedforward equations*, for $k = 1, \ldots, n_l$:

$$\begin{aligned} z_k^{(l)} &= \sum_j w_{kj}^{(l)} x_j^{(l)} + b_k^{(l)} \\ x_k^{(l+1)} &= \phi^{(l)}(z_k^{(l)}) \end{aligned} \tag{2.5}$$

and the NN output $\hat{\mathbf{y}} = (\hat{y}_k) \in \mathbb{R}^{n_L}$, with n_L the dimension of the output, reads:

$$\hat{y}_k = \phi^{(L)}(z_k^{(L)}) \tag{2.6}$$

In tensor notation, (2.3)-(2.6) can be expressed as the following:

$$\begin{aligned}
\mathbf{z}^{(1)} &= \mathbf{W}^{(1)}\mathbf{x}^{(1)} + \mathbf{b}^{(1)} \\
\mathbf{x}^{(2)} &= \phi^{(1)}.(\mathbf{z}^{(1)}) \\
&\dots \\
\mathbf{z}^{(l)} &= \mathbf{W}^{(l)}\mathbf{x}^{(l)} + \mathbf{b}^{(l)} \\
\mathbf{x}^{(l+1)} &= \phi^{(l)}.(\mathbf{z}^{(l)}) \\
&\dots \\
\mathbf{z}^{(L)} &= \mathbf{W}^{(L)}\mathbf{x}^{(L)} + \mathbf{b}^{(L)} \\
\hat{\mathbf{y}} &= \phi^{(L)}.(\mathbf{z}^{(L)})
\end{aligned} \tag{2.7}$$

where $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$, $\phi^{(l)}$ are weights, biases, and the activation function of the l-th layer, respectively; $\mathbf{x}^{(l)} \in \mathbb{R}^{n_{l-1}}$, $\mathbf{z}^{(l)} \in \mathbb{R}^{n_l}$. Notation $\phi.(\mathbf{x})$ means that the function ϕ is applied to each component of the vector $\mathbf{x}$.

By defining the function $\mathbf{\Sigma}^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$:

$$\mathbf{\Sigma}^{(l)}(\mathbf{x}^{(l)}) = \phi^{(l)}.(\mathbf{W}^{(l)}\mathbf{x}^{(l)} + \mathbf{b}^{(l)}) \tag{2.8}$$

it follows that a NN is formally the composition of non-linear functions:

$$\hat{\mathbf{y}}(\mathbf{x}^{(1)}) = \mathbf{\Sigma}^{(L)} \circ \mathbf{\Sigma}^{(L-1)} \circ \cdots \circ \mathbf{\Sigma}^{(1)}(\mathbf{x}^{(1)}) \tag{2.9}$$

Backpropagation

A NN can be conceptualized as a dataflow graph, or a circuit where each node represents a simple computation that decides how much information is passed on from the predecessor to the next. All DL methods use the same principle: a dataflow graph is constructed, and its weights are adjusted to fit the data.

Given the choice of the NN architecture (number of layers, number of neurons, type of activations,...), the weights are initialized, generally with random values, and the input data $\mathbf{x}$ are provided to the model so that the output $\hat{\mathbf{y}}$ is computed following (2.7).

At this point, the prediction $\hat{\mathbf{y}}$ is compared with the real output $\mathbf{y}$. Once the objective function $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$, evaluating the error committed by the NN, is chosen, an attempt is made to minimize such error as a function of the weights and the biases. Indeed, the objective function depends on $\hat{\mathbf{y}}$, hence on all the parameters of the NN, (2.9).

The minimization rests on correcting the weights proportionally to the gradient of $\mathcal{L}$. The *backpropagation algorithm* offers a practical method for computing the gradient of the error function $\mathcal{L}$, utilizing the chain rule of differentiation:

$$\frac{\partial g(f(x))}{\partial x} = g'(f(x))\frac{\partial f(x)}{\partial x} \tag{2.10}$$

After initial computation in the forward pass, the error is transmitted layer by layer backward from the output nodes, hence the name *backpropagation*.

Firstly, consider the NN in Figure 2.3 with a one-dimensional input - $n_0 = 1$, $x^{(1)} = x \in \mathbb{R}$ - and output - $n_L = 1$, $\hat{y} \in \mathbb{R}$. Feedforward equations for this NN read:

$$\begin{aligned} z^{(1)} &= w^{(1)}x + b^{(1)} \\ x^{(2)} &= \phi^{(1)}(z^{(1)}) \\ z^{(2)} &= w^{(2)}x^{(2)} + b^{(2)} \\ \hat{y} &= \phi^{(2)}(z^{(2)}) \end{aligned} \tag{2.11}$$

Equations (2.11) allow to compute the prediction $\hat{y}$ for a corresponding input x, and so the associated mismatch $\mathcal{L} = \mathcal{L}(y, \hat{y})$.

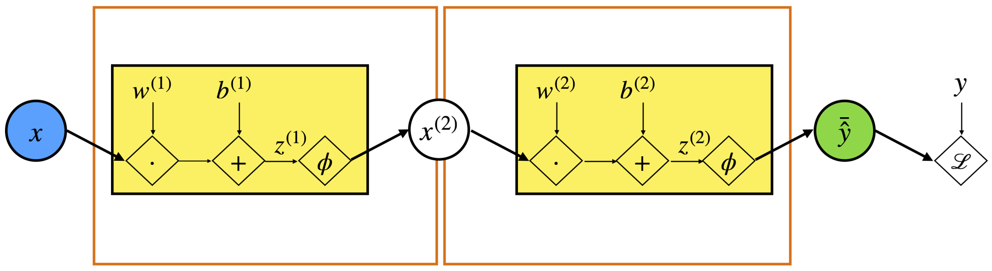


FIGURE 2.3: Representation of the NN examined for backpropagation.

By introducing the following notation:

$$\bar{w} := \frac{\partial \mathcal{L}}{\partial w} \tag{2.12}$$

the derivative of the cost function with respect to the last weight is:

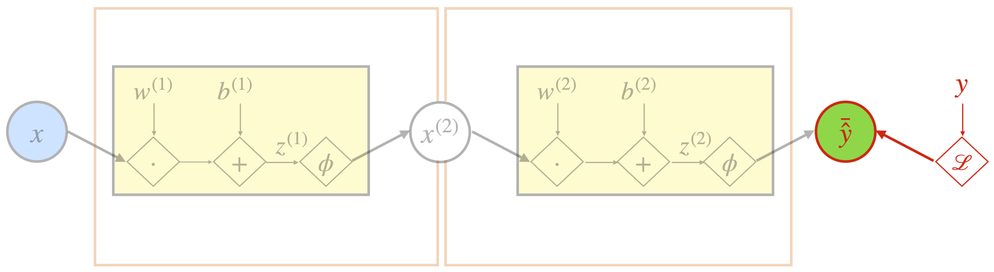
$$\bar{w}^{(2)} = \bar{z}^{(2)} \frac{\partial z^{(2)}}{\partial w^{(2)}} = \bar{\hat{y}} \frac{\partial \hat{y}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial w^{(2)}} \tag{2.13}$$

Note that the term $\bar{\hat{y}} = \dfrac{\partial \mathcal{L}}{\partial \hat{y}}$ is well-defined and easily computed for a given set of data and depends on the chosen type of the cost function.

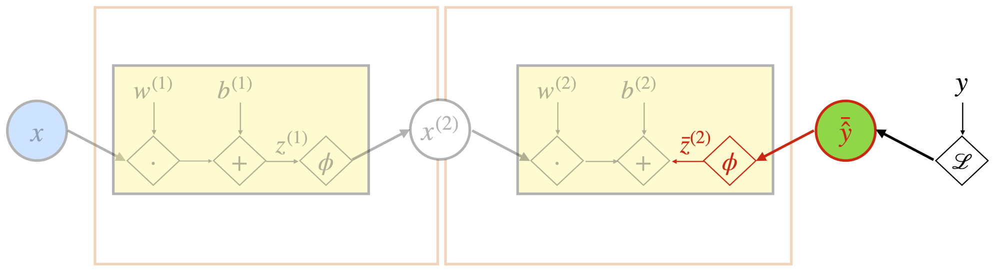
By applying the chain rule (2.10), the following equations arise:

$$\begin{aligned}
\bar{\hat{y}} &= \frac{\partial \mathcal{L}}{\partial \hat{y}} \\
\bar{z}^{(2)} &= \bar{\hat{y}} \phi'(z^{(2)}) \\
\bar{b}^{(2)} &= \bar{z}^{(2)} \\
\bar{w}^{(2)} &= \bar{z}^{(2)} x^{(2)} \\
\bar{x}^{(1)} &= \bar{z}^{(2)} w^{(2)} \\
\bar{z}^{(1)} &= \bar{x}^{(2)} \phi'(z^{(1)}) \\
\bar{b}^{(1)} &= \bar{z}^{(1)} \\
\bar{w}^{(1)} &= \bar{z}^{(1)} x
\end{aligned} \tag{2.14}$$

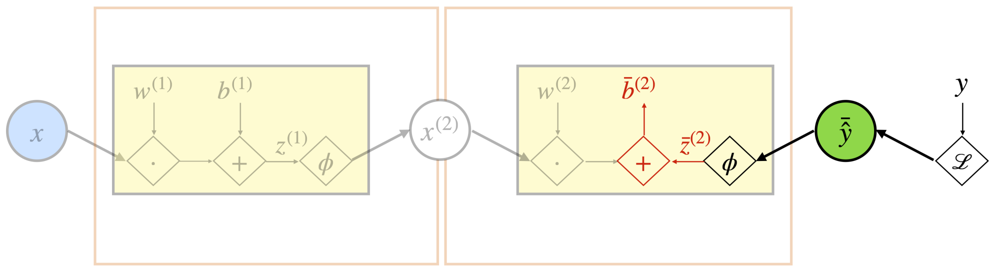
Each term of (2.14) depends on the previous ones and propagates backwards along the network, starting from the cost function (Figure 2.4).



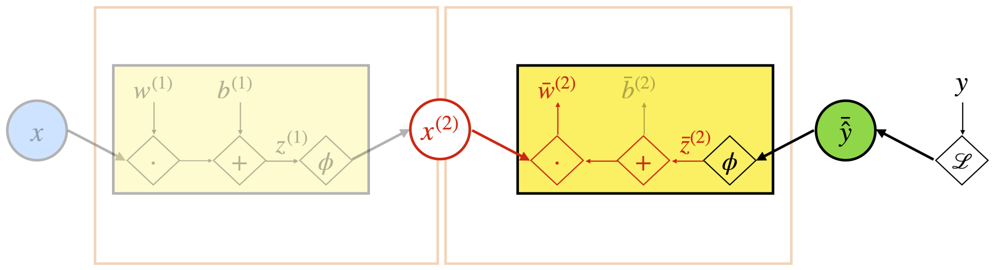
(A) $\bar{\hat{y}} = \dfrac{\partial \mathcal{L}}{\partial \hat{y}}$



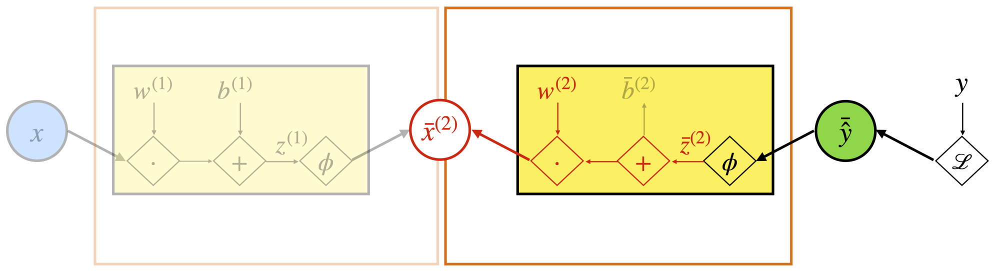
(B) $\bar{z}^{(2)} = \bar{\hat{y}}\phi'(z^{(2)})$



(C) $\bar{b}^{(2)} = \bar{z}^{(2)}$



(D) $\bar{w}^{(2)} = \bar{z}^{(2)}x^{(2)}$



(E) $\bar{x}^{(2)} = \bar{z}^{(2)}w^{(2)}$

FIGURE 2.4: Backpropagation algorithm representation.

For a general NN with L layers, the so-called *backpropagation equations* in tensor notation become:

$$\begin{aligned}
\bar{\hat{\mathbf{y}}} &= \nabla_{\hat{\mathbf{y}}}\mathcal{L} \\
\bar{\mathbf{z}}^{(L)} &= \bar{\hat{\mathbf{y}}} \odot \phi'(\mathbf{z}^{(L)}) \\
&\ldots \\
\bar{\mathbf{x}}^{(l)} &= \mathbf{W}^{(l)T}\bar{\mathbf{z}}^{(l)} \\
\bar{\mathbf{z}}^{(l)} &= \bar{\mathbf{x}}^{(l+1)} \odot \phi'(\mathbf{z}^{(l)}) \\
\bar{\mathbf{b}}^{(l)} &= \bar{\mathbf{z}}^{(l)} \\
\bar{\mathbf{W}}^{(l)} &= \bar{\mathbf{z}}^{(l)}\mathbf{x}^{(l)T} \\
&\ldots
\end{aligned} \tag{2.15}$$

where notation $\odot$ stays for the Hadamard product $(A \odot B)_{ij} = (A)_{ij}(B)_{ij}$.

NN are always described in terms of tensor operations because of their natural inclination toward this notation and computational efficiency reasons. Indeed, tensors enable highly optimized compiled code for *Graphics Processing Unit* (GPU)s and *Tensor Processing Unit* (TPU)s, by processing many inputs in parallel.

2.2 Deep Feedforward Neural Networks

2.2.1 Architecture of Feedforward Networks

A further extension of the perceptron is its multilayered version, known as *Multilayer Perceptron* (MLP), whose origins date back to 1966 [Ivakhnenko and Lapa, 1966; Amari, 1967]. MLP is a feedforward artificial neural network, renowned for its ability to discriminate input that is not linearly separable. MLPs are made up of completely connected neurons with a nonlinear type of activation function, grouped in at least three layers. The attribute *feedforward* refers to the direct acyclic conveyance of signals from input nodes to output nodes, without loops.

Deep *Feedforward Neural Network* are MLP with a huge set of nodes arranged in many layers. Nodes in adjacent layers are linked and each link contains a weight. Every node sends to the following layer an output value, that is the result of a weighted linear combination of input signals received from the previous layer nodes and the application of an activation function.

Deep FNNs are capable of handling complex tasks that traditional perceptrons cannot, thanks to the high number of hidden layers. The idea rests on building long computation paths that allow the input variables to interact in complex ways.

2.2.2 Loss Functions

Loss functions vary according to the task at hand and the objective that has to be achieved. Some of the most common error functions are described hereafter.

Mean Absolute Error (MAE) (or *L1 loss*) is one of the loss functions used in regression settings. MAE takes the average sum of the L1-norms ($||\mathbf{x}||_1 = \sum_i |x_i|$) of the difference between reference values $\mathbf{y}^j$ and predictions $\hat{\mathbf{y}}^j$:

$$MAE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{j=1}^{N} ||\mathbf{y}^j - \hat{\mathbf{y}}^j||_1 \tag{2.16}$$

being N the number of samples $(\mathbf{x}^j, \mathbf{y}^j)$, and $\hat{\mathbf{y}}^j = \hat{\mathbf{y}}(\mathbf{x}^j)$. MAE does not depend on the reflection of the coordinate sistem about a coordinate axis or its translation. The first property makes them particularly recommended in presence of *outliers* - i.e. values that are very different from the rest of the data - because it does not take into account their direction.

Another common loss function for regression is *Mean Squared Error* (MSE) (or *L2 loss*), since it is associated with the L2-norm ($||\mathbf{x}||_2 = \sqrt{\sum_i x_i^2}$):

$$MSE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{j=1}^{N} ||\mathbf{y}^j - \hat{\mathbf{y}}^j||_2^2 \tag{2.17}$$

Note that each component of the difference vector is squared, and that means that the outliers can weight more, so this can skew results. Usually, MSE works extremely well with data that can be modeled into a Gaussian distribution.

For classification purposes, the main loss function is the *Cross Entropy loss*. In the case of two classes, the datum y^j usually reads as 1 if the corresponding input $\mathbf{x}^j$ belongs to the first class, 0 if it belongs to the second. The so-called *Binary Cross Entropy* takes the following form:

$$\mathcal{L}(y, \hat{y}) = -\sum_{j=1}^{N} y^j \log(\hat{y}^j) + (1 - y^j)\log(1 - \hat{y}^j) \tag{2.18}$$

where the prediction $\hat{y}^j$ is the probability that $\mathbf{x}^j$ belongs to the first class. The straightforward extension to m categorical classes, referred to as *Categorical Cross Entropy loss* is:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{j=1}^{N}\sum_{i=1}^{m} y_i^j \log(\hat{y}_i^j) \tag{2.19}$$

Here, $\mathbf{y}^j = (y_i^j) \in \mathbb{R}^m$ is a one-hot encoded vector that has the k-th entry equal to 1 if $\mathbf{x}^j$ belongs to the k-th class, otherwise equal to 0, and $\hat{\mathbf{y}}^j = (\hat{y}_i^j)$ is the vector of the predicted probabilities that $\mathbf{x}^j$ belongs to those classes.

2.2.3 Gradient Descent and Optimization

A NN can be simply thought as a function aiming at representing the relationship between input x and output y, say $\hat{y} = f(x)$. This function relies on a number of weights (and biases) w that modulate the signal of a layer during the conveyance to the following. Therefore, it can be expressed as $\hat{y} = f(x, w)$, and accordingly the cost function $\mathcal{L}(y, \hat{y}) = \mathcal{L}(y, f(x, w))$. The best model is the one that minimizes the mismatch between data and predictions, hence it can be found by using an appropriate optimization algorithm. The most popular optimization technique used for this task is the *Gradient Descent* (GD) - including all its variants - because of its flexibility in a wide variety of problems.

Gradient Descent

The basic idea underlying the GD algorithm relies on applying an iterative procedure that looks for the global minimum of a function by finding local minima along a sequence of restricted (mono-dimensional) problems. The local gradient of the objective function is computed with respect to a vector of parameters and the algorithm moves in the descending direction of the gradient.

In the specific case of NNs, the parameters θ are the weights and the biases of the linear combinations in the neurons. They are generally initialized with random values, then they are updated at each iteration following:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta) \tag{2.20}$$

with the goal of reducing the loss function $\mathcal{L}$ until the algorithm reaches the global minimum. A key GD hyperparameter is the *learning rate* η, which controls the step size in the gradient descending direction. If η is too small, the descent is slow and algorithm takes many iterations to attain convergence. By distinction, if it is too large, there is a risk of passing over the minimum trough, jumping from one side to the other of a local convex hull.

For each input-output pair, the cost and its gradient are computed and the weights are consequently changed to move close to the global minimum. Once the algorithm has seen the whole dataset an *epoch* has been performed, and it starts again the weight update for the next epoch. This strategy that considers the entire dataset through each iteration in order to discover the best answer, may be time-consuming and computationally expensive.

A useful variant of GD, called *Stochastic Gradient Descent* (SGD), addresses some of its drawbacks. Instead of using all the observations, each SGD iteration computes the gradient using one randomly selected partition of the dataset that was shuffled. The computational time can be drastically decreased, but this technique might produce noisier outcomes than GD.

Mini Batch Gradient Descent is a hybrid variant of GD and SGD. At each epoch, this method splits the dataset into small shuffled subsets (*mini batches*), computes the gradients for each batch, and avoids iterating over the full dataset of observations. Mini Batch GD is summarized in Algorithm 1. Usually, mini batches improve the algorithmic scalability, hence its performance for large datasets and in parallel computational environments.

Algorithm 1 Mini Batch Gradient Descent algorithm

```
1: Initialize $\theta$
2: Set $\eta$
3: while stop condition do
4:     Choose randomly a mini-batch from input-output dataset
5:     Compute $\mathcal{L}$
6:     Compute $g = \nabla_\theta \mathcal{L}(\theta)$
7:     Update $\theta \to \theta - \eta g$
8: end while
```

Adaptive Moment Estimation

Adaptive Moment Estimation (Adam) optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments of the gradient to adapt the learning rate for each weight of the neural network.

Kingma and Ba [2015] introduced Adam that was found to behave in a more effective way in terms of generalizing performance with respect to vanilla SGD when the gradients are small. According to the authors, the method is "straightforward to implement, computationally efficient, has little memory requirements, is invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data and/or parameters". Adam provides an optimization algorithm that can handle sparse gradients on noisy problems and non-stationary objectives [Pomerat et al., 2019]. The adaptive learning rate qualifies it for a wide range of optimization tasks.

Algorithm 2 Adaptive Moment Estimation algorithm

```
1: Initialize $\theta, m, v$
2: Set $\eta, \rho_1, \rho_2, \epsilon$
3: while stop condition do
4:     Update $\tau \to \tau + 1$
5:     Choose randomly a mini-batch from input-output dataset
6:     Compute $\mathcal{L}$
7:     Compute $g = \nabla_\theta \mathcal{L}(\theta)$
8:     Store $m \to \rho_1 m + (1 - \rho_1) g$
9:     Store $v \to \rho_2 v + (1 - \rho_2) g \odot g$
10:    Correct $\hat{m} = \frac{m}{1-\rho_1^\tau}$
11:    Correct $\hat{v} = \frac{v}{1-\rho_2^\tau}$
12:    Compute $g' = \eta \frac{1}{\sqrt{\hat{v}+\epsilon}} \odot \hat{m}$
13:    Update $\theta \to \theta - \eta g'$
14: end while
```

Algorithm 2 provides an outline of the method. Default settings for the hyper-parameters are $\eta = 0.001$, $\rho_1 = 0.9$, $\rho_2 = 0.999$, and $\epsilon = 10^{-8}$.

Adaptive learning rates are determined for each parameter using past gradients (first moment estimate, m), and squared gradients (second moment estimate, v). The exponential decay rates ρ_1 and ρ_2 determine how much past gradients and squared gradients influence current updates. The first and second moment estimate are then bias-corrected to $\hat{m}$ and $\hat{v}$, respectively.

The approach adjusts the learning rates for each parameter individually, allowing it to handle sparse gradients and converge quickly on a wide range of optimization problems. Adam offers several other benefits, including the fact that its step sizes are roughly limited by the step-size hyper-parameter, that the magnitudes of weight updates are invariant to gradient rescaling, and that it inherently performs a kind of step-size annealing. Because of its robust performance and adaptive nature, Adam - and some variants of it [Reddi et al., 2018] - have become a popular choice for optimizing deep neural networks.

2.2.4 Training a Feedforward Network

Here, the *training*, i.e. the process aimed at building the model by computing the weights and biases, is described in detail.

Once the architecture of FNN is designed, with its input, hidden, and output layers, and appropriate activation functions are chosen, the NN parameters have to be computed. Training a neural network is a critical step, where the network learns how to make accurate predictions by adjusting its internal parameters - weights and biases - based on the provided training data.

This learning process diversifies into supervised, unsupervised, and reinforcement learning, depending on the type of information at hand.

During the training of a NN under *supervised learning*, the input is fed to the network, which will output a value that is compared with the desired output. The error between the two is computed, and if there is a discrepancy between the intended output and the prediction, the NN is updated based on this error signal to reduce the discrepancy between the desired and actual outputs.

When the training is performed under *unsupervised learning*, similar-type inputs are combined to form clusters. When the NN is applied to a new input pattern, it produces an output response that indicates the class the input belongs in. No feedback is provided regarding what the intended outcome should be or whether it is proper, so the network itself has to figure out how to identify patterns and features from the input data, and how to relate the input to the output.

As the name suggests, *reinforcement learning* is used to reinforce or strengthen the network over some critic information. The NN is provided with some feedback during this kind of training: this puts it in line with supervised learning in certain ways, although the feedback received here is evaluative rather than instructive. The network adjusts in response to the feedback to get better critic information in future.

Training begins with the forward propagation of data through the network, following (2.7). The input data is passed through the input layer and processed layer by layer through the hidden ones until it reaches the output layer. At each neuron, weighted sums of inputs are calculated, and activation functions are applied to produce output values. These values are then passed to the next layer.

To measure how well the NN predictions match the actual target values in the training data, a loss function computes a single scalar value that reflects the difference between the predicted outputs and the true targets. By means of the backpropagation equations (2.15), the gradients of the loss function with respect to each weight and bias in the network are computed. These gradients indicate the direction and magnitude of changes needed to minimize the loss. By applying the chain rule of calculus (2.10), backpropagation algorithm efficiently propagates the error backward through the network, providing information on how each weight and bias should be adjusted.

Observe that during the training the loss function is minimized as a function of the parameters of the NN. Once the gradients are computed, optimization algorithms, like GD or Adam, are applied to update the NN parameters iteratively: they adjust weights and biases in the direction that reduces the loss, gradually bringing the NN predictions closer to the target values.

Training occurs over multiple iterations. In each epoch, the entire training dataset is passed through the network, and the weights and biases are updated based on the computed gradients. The number of epochs depends on factors such as the complexity of the task and the convergence of the training process. It may require experimentation to find the optimal number.

The minimization will find, at the end, the optimal values of the parameters at which the prediction error is minimal. At this point, the NN is trained and the model is ready for making predictions based on new inputs.

2.3 Convolutional Neural Networks

A *Convolutional Neural Network* (CNN) is designed primarily for processing two-dimensional grid-like data, such as images, with high degree of invariance to translation, skewing, scaling, and other forms of distortion. The structure of CNN makes it possible to preserve the meaning of adjacency of the elementary components in the samples (e.g. pixels in images), that would be lost if the internal layers are fully connected.

In CNNs, each hidden node receives input from only a small local region of the datum and the weights that connect it to a node in the hidden layer are the same for each hidden node. As a consequence, adjacency is respected (at least locally), spatial invariance is detected, and the number of weights is cut-down.

2.3.1 Convolutional Layers

A *convolutional layer* is a fundamental building block in a CNN, specialized for detecting local patterns and features within the input data. Convolutional layers are crucial for image recognition tasks because they allow the network to learn hierarchical features. The initial layers may detect simple features like edges and corners, while subsequent layers can learn to combine these features to recognize more complex structures like shapes and objects.

At the core of a convolutional layer is the convolution operation. It involves sliding a small filter (also known as a *kernel*) across the input data. This filter is a smaller grid of learnable weights. As the filter moves across the input, it performs element-wise multiplications between its values and the corresponding values in the input data, producing a weighted sum. The weighted sum, often referred to as the *feature map*, represents the presence of specific patterns or features in the input. These patterns might be edges, textures, or more complex structures, depending on the characteristics of the filter.

Convolutional layers use shared weights within a single layer. This means that the same filter is applied to different parts of the input data. Sharing weights helps the network generalize better by detecting the same features across different regions of the input and reduces the number of parameters to be tuned.

Typically, a convolutional layer does not use just one filter but multiple filters in parallel. Each filter specializes in detecting different features or patterns. The number of kernels corresponds to the number of channels; when dealing with images in RGB notation, along the NN there are no longer three dedicated color channels, but a collection of feature maps, one for each kernel, carrying information from one feature.

After the convolution operation, each feature map often passes through an activation function, usually the ReLU. This introduces non-linearity into the network and helps capture complex relationships within the data.

2.3.2 Pooling Layers

In many CNN architectures, after one or more convolutional layers, there may be *pooling layers*, such as max-pooling or average-pooling. These layers reduce the spatial dimensions of the feature maps, making them smaller and more manageable while preserving important features. As a result, the model has fewer parameters overall, which improves computational efficiency. Pooling summarizes a set of adjacent units from the preceding layer with a single value, resulting in down sampling the input. The activation function is usually linear.

Average-pooling computes average value of the neighborhoods, resulting in a coarser resolution of the input. On the other hand, *max-pooling* calculates the maximum value, looking for a feature that exists somewhere in the receptive field. Layers of this kind facilitate multiscale recognition and reduce the weights required in subsequent layers, thus involving lower computational cost and faster learning.

At each convolutional or pooling layer the spatial resolution is decreased, while the number of feature mappings is raised, compared to the previous layer.

2.3.3 Applications of CNNs

Convolutional layers enable CNNs to automatically extract relevant features from input data, making them highly effective for tasks like image classification, object detection, and image segmentation.

Image classification is the task of assigning a label or class to an input image. It is a supervised learning problem, where a model is trained on a labeled dataset of images and their corresponding class labels, and then used to predict the class label of new, unseen images.

In computer vision, *object detection* looks for occurrences of a specific class of objects of interest (like people, structures, or cars), mainly in digital images and videos.

The division of a digital image into various segments is the goal *image segmentation*. The process reduces complexity and provides more understandable representations of images, frequently used to identify objects and boundaries in images, as it gives each pixel a label so that pixels with the same label have specific properties.

2.4 Recurrent Neural Networks

A *Recurrent Neural Networks* (RNN) differs from a FNN in that it has at least one feedback loop. This means that the output of a neuron is supplied as input to at least one neuron of the previous layer, and has a strong impact on the learning capability of the network and its performance. RNN structure makes them suitable to face a problem with sequential data, since feedback loops allow to retain in memory the state from one iteration to the following.

Feedback loops work as in Figure 2.5, and can be unfolded along the axis of consequentiality - which usually is the time axis - and considered as a concatenation of different steps of the same RNN, taking as input the current input $\mathbf{x}_t$ and the output of the previous step $\mathbf{h}_{t-1}$. By doing this, the network can keep track of the past history of the data.

Every RNN has the shape shown in Figure 2.5, with a potential difference introduced in the architecture of the A-module. For basic RNNs, called *Vanilla Recurrent Neural Networks*, a single layer with one node constitutes the module (Figure 2.6).

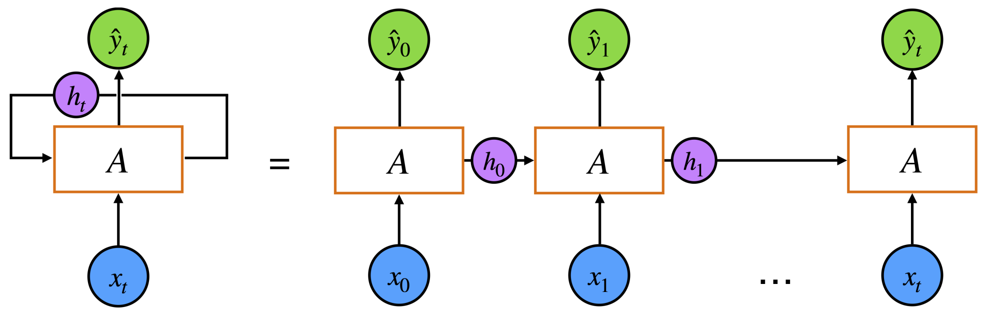


FIGURE 2.5: Representation of a RNN.

At each time t, the network carries out usual linear combination with weights $\mathbf{W}$ and biases $\mathbf{b}$ of a vector that now is the concatenation $[\mathbf{h}_{t-1}; \mathbf{x}_t]$. Then it applies the activation function ϕ (often $\tanh$):

$$\begin{aligned} \mathbf{z}_t &= \mathbf{W}[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b} \\ \mathbf{h}_t &= \hat{\mathbf{y}}_t = \phi(\mathbf{z}_t) \end{aligned} \tag{2.21}$$

The hidden state at time t, denoted by $\mathbf{h}_t$, depends on the hidden state at the previous time step $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$: $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$. In general, $\mathbf{h}_t$ can differ from the output $\hat{\mathbf{y}}_t$ generated by the RNN at the current time t, which is likewise a function of $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$.

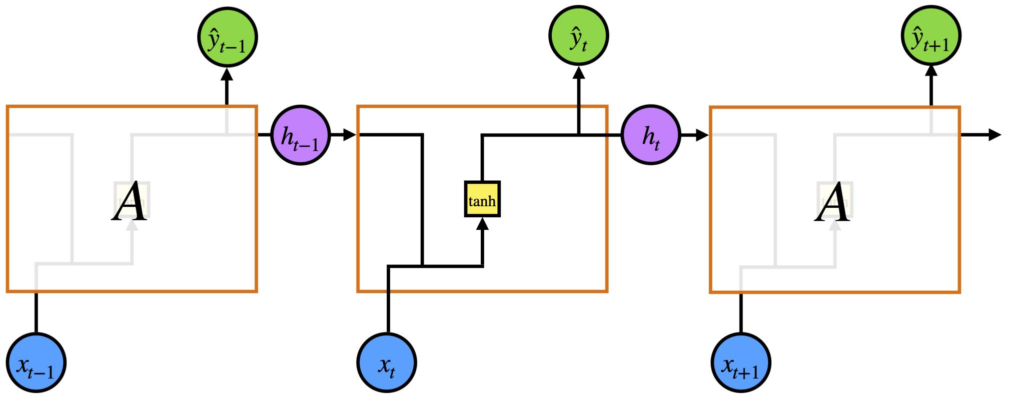


FIGURE 2.6: Vanilla RNN.

2.4.1 Vanishing Gradient Problem

A common problem that arises during the training of deep NNs with many layers by using methods based on GD and backpropagation is the *Vanishing Gradient problem*. At each iteration, such methods update every weight of the NN on the basis of the partial derivative of the loss function with respect to the weight itself. When the derivative is extremely small, the weights remains almost unvaried, so the training could never converge to a good solution.

In 1991, Sepp Hochreiter found the Vanishing Gradient problem being the cause of the poor success of RNN training [Hochreiter, 1991]. The recurrent nature of the architecture makes this type of NN extremely exposed to vanishing gradients and entails a gradual loss of the signal of the current error resulting in hardly learned long-term dependencies [Pascanu et al., 2013].

As said before, $\hat{\mathbf{y}}_t$ and $\mathbf{h}_t$ are functions of current input and previous time step hidden state. For the sake of simplicity, the case in which $\hat{\mathbf{y}}_t = \mathbf{h}_t$ is examined, and it could be easily extended to the more general one. It holds that $\hat{\mathbf{y}}_t = f(\hat{\mathbf{y}}_{t-1}, \mathbf{x}_t, \theta)$, with θ the parameters defining the RNN. By introducing notation $f(t) := f(\hat{\mathbf{y}}_{t-1}, \mathbf{x}_t, \theta)$, the differential read as:

$$d\hat{\mathbf{y}}_t = \nabla_\theta f(t) d\theta + \nabla_{\hat{\mathbf{y}}} f(t) d\hat{\mathbf{y}}_{t-1} \tag{2.22}$$

During GD optimization, the parameters θ are updated following (2.20) by means of $\bar{\theta} = \nabla_\theta \mathcal{L}(\theta) = \nabla_\theta \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$. At time T, by recursively applying (2.22), the expression for $\bar{\theta}$ is:

$$\begin{aligned}\bar{\theta} &= \bar{\hat{\mathbf{y}}}_T d\hat{\mathbf{y}}_T \\ &= \bar{\hat{\mathbf{y}}}_T \nabla_\theta f(T) d\theta + \nabla_{\hat{\mathbf{y}}} f(T) d\hat{\mathbf{y}}_{T-1} \\ &= \bar{\hat{\mathbf{y}}}_T \nabla_\theta f(T) d\theta + \nabla_{\hat{\mathbf{y}}} f(T)(\nabla_\theta f(T-1) d\theta + \nabla_{\hat{\mathbf{y}}} f(T-1) d\hat{\mathbf{y}}_{T-2}) \\ &\dots \\ &= \bar{\hat{\mathbf{y}}}_T (\nabla_\theta f(T) + \nabla_{\hat{\mathbf{y}}} f(T) \nabla_\theta f(T-1) + \dots) d\theta\end{aligned} \tag{2.23}$$

To further simplify the notation, consider the case of scalar input x_T and output y_T, $\phi = \tanh$. Forward equations read as:

$$\begin{aligned} z_T &= w_1 \hat{y}_{T-1} + w_2 x_T + b \\ h_T &= \hat{y}_T = \tanh(z_T) \end{aligned} \tag{2.24}$$

In this case, $\hat{y}_T = f(\hat{y}_{T-1}, x_T, w_1, w_2, b) = \tanh(w_1\hat{y}_{T-1} + w_2x_T + b)$. Consider the weight w_1:

$$\begin{aligned}\bar{w}_1 &= \bar{\hat{y}}_T \frac{\partial \hat{y}_T}{\partial w_1} \\ &= \bar{\hat{y}}_T(\tanh'(z_T)\hat{y}_{T-1} + \tanh'(z_T)w_1\frac{\partial \hat{y}_{T-1}}{\partial w_1}) \\ \dots \\ &= \bar{\hat{y}}_T(\tanh'(z_T)\hat{y}_{T-1} + w_1\tanh'(z_T)\tanh'(z_{T-1})\hat{y}_{T-2} + \dots)\end{aligned} \tag{2.25}$$

Therefore, the contribution of $\hat{y}_{T-s}$ in the update of w_1 is controlled by:

$$w_1^s \prod_{i=1}^{s} \tanh'(z_{T-i+1}) \tag{2.26}$$

Given that $\tanh'(x) \leq 1, \forall x$, (2.26) is bounded above by w_1^s, If $w_1 < 1$, for large s, w_1 tends to 0, and $\hat{y}_{T-s}$ does not affect the update.

The same generalized equations hold for vector inputs and outputs, and a general activation function.

2.4.2 Long Short-Term Memory

A possible solution to the vanishing gradient problem is the *Long Short-Term Memory* (LSTM) architecture (Figure 2.7). Designed in 1997 by Sepp Hochreiter e da Jürgen Schmidhuber [Hochreiter and Schmidhuber, 1997] - and gradually improved over the years - LSTMs have the capability to catch long-term dependancies on data.

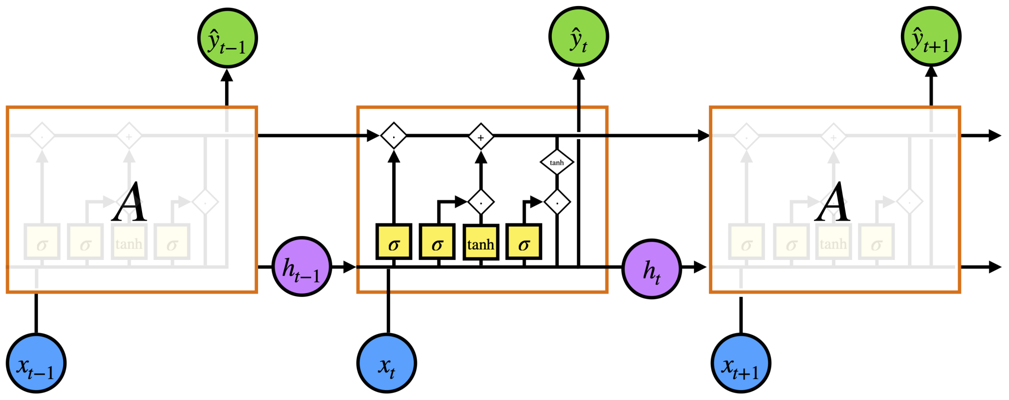


FIGURE 2.7: Long Short-Term Memory

Their main peculiarity are the *gate* structure and the *cell state* $\mathbf{C}_t$, that travels along the entire NN as a conveyor belt of information. As shown in Figure 2.8, a gate is a layer that applies the logistic function $\sigma(z) = \frac{1}{1+e^{-z}}$ (see Figure 2.1) and then multiplies two vectors component-wise. This calculation consents to manage the memory of the NN and ponder data storage: the logistic function ranges from 0 to 1, hence, when it multiplies a value, it balances the amount of information to draw: from 0 (nothing) to 1 (everything).

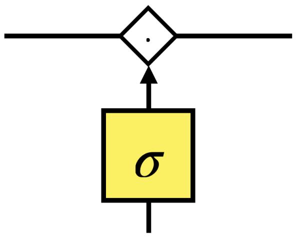


FIGURE 2.8: Gate structure

$\mathbf{C}_t$ is updated at each time by removing or adding new knowledge, or simply read through input, forget, and output gate, respectively.

The *forget gate* draws a vector $\mathbf{f}_t$ with the same number of components as $\mathbf{C}_t$, with values between 0 and 1, based upon the input $\mathbf{x}_t$ and the hidden state $\mathbf{h}_{t-1}$. The components of $\mathbf{f}_t$ decide the percentage of information in the cell state that is to be kept (1 means remember everything, 0 completely forget).

The network then decides the new information to add. A first layer with hyperbolic tangent activation outputs a vector $\tilde{\mathbf{C}}_t$ of the sellable values to add; later the *input gate* decides which values to actually add, by means of vector $\mathbf{i}_t$ - with entries in the $[0, 1]$ range too.

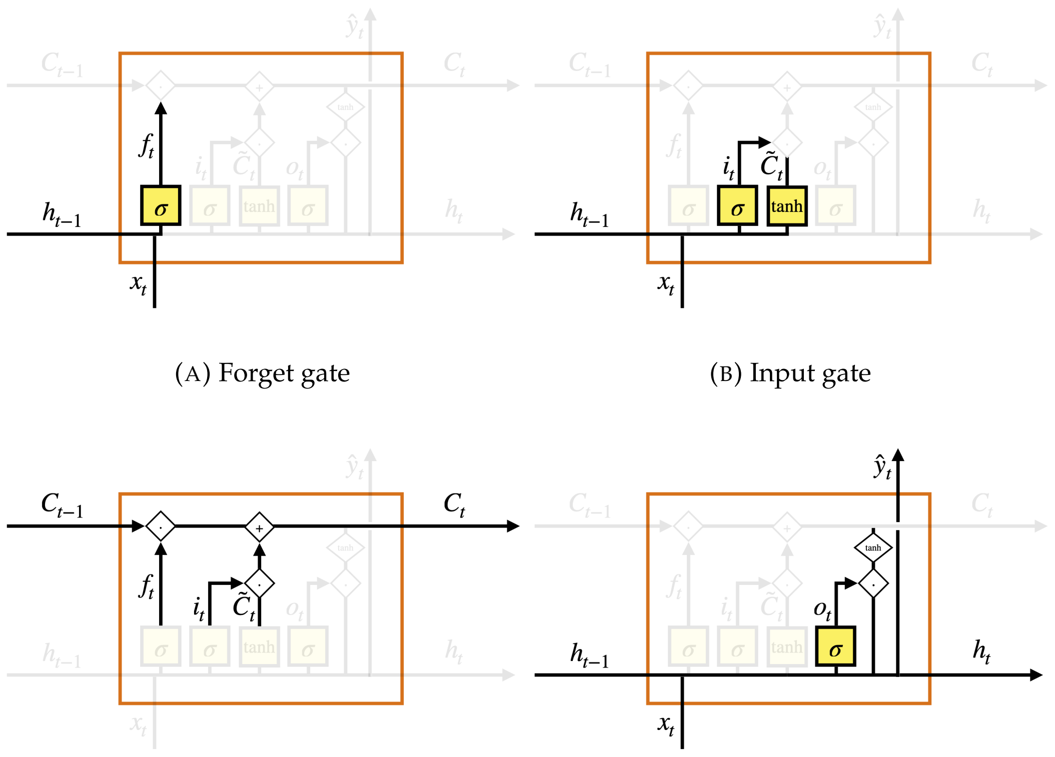
To compute $\mathbf{C}_t$, the cell state at the previous time-step $\mathbf{C}_{t-1}$ is multiplied by $\mathbf{f}_t$ - to forget the chosen knowledge - and $\mathbf{i}_t \odot \tilde{\mathbf{C}}_t$ is added - to memorize the information needed.

Finally, the $\mathbf{o}_t$ vector in the *output gate* chooses which part of $\mathbf{C}_t$ values to output.

The whole process is summarized in Figure 2.9 and in the following:

$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_i) \\
\tilde{\mathbf{C}}_t &= \tanh(\mathbf{W}_C[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_C) \\
\mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_o) \\
\mathbf{h}_t &= \hat{\mathbf{y}}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)
\end{aligned}
\tag{2.27}
$$

where $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_C$, $\mathbf{W}_o$ are weights matrices and $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_C$, $\mathbf{b}_o$ bias vectors.



(A) Forget gate (B) Input gate

(C) Cell state update (D) Output gate

FIGURE 2.9: LSTM data-flow

2.4.3 Applications of RNNs

A popular application of RNNs is in the field of *Natural Language Processing* (NLP). NLP is a field of study that looks at ways to utilize computers to interpret and handle natural language text or speech for practical purposes. The goal of NLP is to learn more about how people interpret and utilize language so that the right tools and methods may be created to help computers comprehend and manipulate natural languages in order to complete the tasks that are needed [Chowdhury, 2003].

Speech recognition, also known as *automatic speech recognition* or *voice recognition*, is a technology that enables a computer or machine to convert spoken language into written text. It is a subfield of NLP and plays a crucial role in various applications, making it possible for machines to understand and process human speech. The use of RNNs has improved the accuracy and robustness of speech recognition systems, making them highly effective in real-world applications like voice-controlled devices, automatic transcription services, voice assistants (e.g., Siri, Alexa, Google Assistant), and accessibility tools for individuals with disabilities [Deng et al., 2013]. Overall, speech recognition technology enables seamless human-machine interaction through spoken language, making it an essential component of modern communication and automation systems.

A *time series* is a time ordered sequence of data. Time series can be distinguished in *univariate* if the data are scalars (e.g. temperature values during the day or the number of downloads per minute of an application), or *multivariate* if the components are vectors (e.g. the speed of a car as a function of time or stereo recordings with two audio channels). RNNs are mainly applied to time series to solve problems such as forecasting, anomaly detection, and pattern recognition.

Forecasting refers to predict future samples in a sequence and is a sort of (nonlinear) regression problem, since the goal is predicting a continuous quantity through some features of the time series.

The goal of *anomaly detection* is to identify any deviation from a regular pattern. If the kind of anomalies is known, the problem can be addressed as classification. Otherwise, a regression model needs to be trained to predict future values and find differences between predictions and actual values; any considerable discrepancy is recognized as anomaly.

Finally, *pattern recognition* is the classification of time serie by automatically detecting regularities and patterns.

In all these cases, the time-dependence of the datum is a basic feature that is to be carefully considered, especially during training and evaluation of the model. A shuffled split of data into train and test will loose the consequentiality, so it is the norm to split time series into two smaller ones in a way that all the test data follow the train ones, as in Figure 2.10.

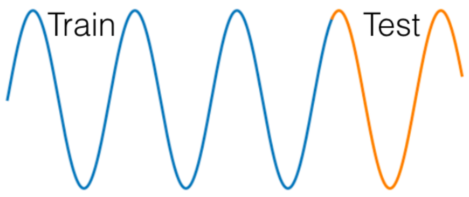


FIGURE 2.10: Time series split for training and validation of RNN.

2.5 Physics-Informed Neural Networks

Physics-Informed Neural Network (PINN) is a particular NN that makes it possible to incorporate information from the physics in addition to data by means of the governing PDEs [Raissi et al., 2019]. Indeed, the so-called loss function that has to be minimized contains not only data and prediction mismatch (as in traditional DL), but also the residual of the governing equations.

With respect to standard neural networks, PINNs usually allow to obtain reasonable approximations with smaller datasets, as the governing physical laws are added as constraints. Furthermore, the residual contribution acts as a regularization term and increases the generalization ability of the neural network, thus allowing to deal with noisy data and outliers without a significant lack of robustness.

In particular, PINN addresses two major challenges of deep learning application in Earth system science, as identified in Reichstein et al. [2019]: (i) physical consistency, thanks to the residual term of the loss function providing a physical constraint, and (ii) limited labels, as the information given by the PDE balances the lack possibly coming from data, since Earth measurements are often sparse and noisy.

2.5.1 PINN loss function

For time dependent problems, the input of the first layer, $\mathbf{x}^{(1)} \in \mathbb{R}^{n_0}$, reads $\mathbf{x}^{(1)} = (\mathbf{x}, t)$ and $n_0 = n+1$, with $\mathbf{x} \in \Omega \subset \mathbb{R}^n$ and $n = 1, 2, 3$ the spatial dimension. Then, the neural network $\hat{\mathbf{u}}(\mathbf{x}, t) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ with L layers can be written as:

$$\hat{\mathbf{u}}(\mathbf{x}, t) = \mathbf{\Sigma}^{(L)} \circ \mathbf{\Sigma}^{(L-1)} \circ \cdots \circ \mathbf{\Sigma}^{(1)}(\mathbf{x}, t). \tag{2.28}$$

with $\mathbf{\Sigma}^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ the function defined in (2.8).

Assume that a general PDE holds true in $\Omega \times [0, +\infty)$:

$$\mathbf{u}_t + \mathcal{N}[\mathbf{u}, \lambda] = 0, \quad \mathbf{x} \in \Omega \subset \mathbb{R}^n, \quad t \geq 0, \tag{2.29}$$

where $\mathcal{N}[\cdot]$ is a non linear differential operator in space, that could depend on some parameters λ, and the subscript $\cdot_t$ indicates the derivative with respect to time. The solution $\mathbf{u}(\mathbf{x}, t) : \Omega \times [0, +\infty) \to \mathbb{R}^m$, with m the output space size, is approximated by a neural network $\hat{\mathbf{u}}(\mathbf{x}, t)$, whose training is performed by minimizing a cost function that includes not only the errors with respect to data, but also the residual of Equation (2.29), thus constraining the model to both fit the data and comply with the expected physics. Furthermore, while data approximation is a point-wise learning mechanism, the residual entails also local knowledge of the solution, given by physical processes and dynamics involving partial derivatives [Kadeethum et al., 2020a].

Let us denote by $\mathcal{NN}$ the function set of all the neural networks $\hat{\mathbf{u}}(\mathbf{x}, t)$ defined in $\Omega \times [0, +\infty)$. Using the *Mean Squared Error* (MSE) (2.17) as loss measure, the training of the model aims at minimizing the functional $\mathcal{L} : \mathcal{NN} \to [0, \infty)$:

$$\mathcal{L}(\hat{\mathbf{u}}) = w_d \frac{1}{N_d} \sum_{i=1}^{N_d} \|\hat{\mathbf{u}}(\mathbf{x}_d^i, t_d^i) - \mathbf{u}^i\|_2^2 + w_c \frac{1}{N_c} \sum_{i=1}^{N_c} \|\hat{\mathbf{u}}_t(\mathbf{x}_c^i, t_c^i) + \mathcal{N}[\hat{\mathbf{u}}(\mathbf{x}_c^i, t_c^i), \lambda]\|_2^2, \tag{2.30}$$

with $\{\mathbf{u}^i\}_{i=1}^{N_d}$ the set of N_d training data for $\hat{\mathbf{u}}(\mathbf{x}, t)$, located at the data points $\{\mathbf{x}_d^i, t_d^i\}_{i=1}^{N_d}$; $\{\mathbf{x}_c^i, t_c^i\}_{i=1}^{N_c}$ the set of N_c collocation points for the residual computation; w_d and w_c proper weights to balance the two contributions in (2.30) (Figure 2.11).

It is worth notice that, depending on the nature of the problem, the NN for the

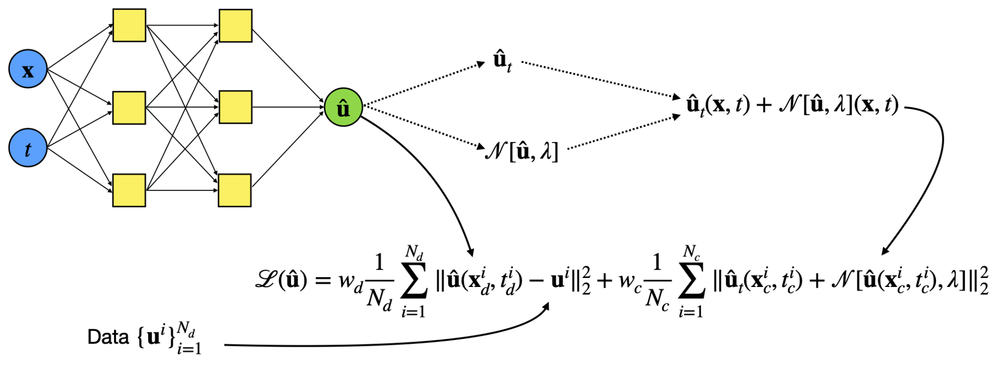

$\mathbf{x}$
t
$\hat{\mathbf{u}}$
$\hat{\mathbf{u}}_t$
$\mathcal{N}[\hat{\mathbf{u}}, \lambda]$
$\hat{\mathbf{u}}_t(\mathbf{x}, t) + \mathcal{N}[\hat{\mathbf{u}}, \lambda](\mathbf{x}, t)$
$\mathcal{L}(\hat{\mathbf{u}}) = w_d \frac{1}{N_d} \sum_{i=1}^{N_d} \|\hat{\mathbf{u}}(\mathbf{x}_d^i, t_d^i) - \mathbf{u}^i\|_2^2 + w_c \frac{1}{N_c} \sum_{i=1}^{N_c} \|\hat{\mathbf{u}}_t(\mathbf{x}_c^i, t_c^i) + \mathcal{N}[\hat{\mathbf{u}}(\mathbf{x}_c^i, t_c^i), \lambda]\|_2^2$
Data $\{\mathbf{u}^i\}_{i=1}^{N_d}$

FIGURE 2.11: Scheme of the PINN approach for the solution of Equation (2.29). The NN $\hat{\mathbf{u}}$ is trained by minimizing the loss function in (2.30), with the contribution of the training data and of the residual.

solution approximation, $\hat{\mathbf{u}}$, could be of any kind with the minimal or even null change in the model implementation. This means that $\hat{\mathbf{u}}$ might be either a CNN [Zhang, 2022], or a RNN [Asrav and Aydin, 2023], or another different type of NN.

2.5.2 Applications of PINNs

The combination of *Machine Learning* (ML) with physical modeling has been identified as one of the most promising and challenging approaches for modeling real-world systems [Reichstein et al., 2019; Karniadakis et al., 2021], since it could result in a potentially significant breakthrough for efficiency, accuracy and generalization capability of numerical methods for PDE solution.

The peculiarity of PINN is that they allow to incorporate information from the physics in addition to data by means of the use of the governing PDEs. Once a PINN has been trained, it can provide an almost real-time solution [Jagtap et al., 2020; Shin et al., 2020; Wang et al., 2020; Fuks and Tchelepi, 2020], thus drastically improving tasks such as sensitivity analysis, model calibration, uncertainty quantification, parametric design, optimization and so on.

The nature of PINN makes them suitable for different classes of problems, which can be classified into two main groups: feedforward solution of differential equations (*direct problem*), and parameter identification (*inverse problem*).

In the first class, PINNs are used as a pure PDE solver, where the user only knows the governing equations alongside with the initial and boundary conditions. By distinction, in the second class some parameters of the PDE are assumed to be unknown, but it is assumed that some data inside the domain are available.

A hybrid approach - similar to a data assimilation approach - where PDE solution is performed leveraging also on known data inside the domain, may also be possible and makes PINNs attractive from a practical point of view.

2.6 Practical Aspects of Deep Learning

There are many real-world considerations and practical challenges that arise when applying DL techniques to solve specific problems or tasks. Implementing deep learning in practical applications involves addressing a range of issues beyond just the theoretical understanding of neural networks. This section higlights some key practical aspects of deep learning.

2.6.1 Overfitting and techniques for regularization

Overfitting means that the model has only memorized the right answer corresponding to training inputs, rather than learn the general connection that links inputs to outputs. This can happen when either the training is carried out for too long, or on a training dataset that is not a representative sample of the problem at hand, or the model is too complex for the amount of available training data. In presence of overfitting, a ML model learns to perform exceptionally well on the training data but fails to generalize to unseen data or test data. In other words, the model becomes too specialized in capturing noise or random variations in the training data, rather than learning the underlying patterns. Overfitting can be recognized in case of a high training accuracy but a significantly lower test accuracy.

Overfitting is a common challenge in deep learning and machine learning, and regularization techniques are used to address it. The choice of regularization and its hyper-parameters depends on the specific problem and dataset, and often requires experimentation to find the best combination for a given task. Among the several possible techniques, *L1-regularization* (*Lasso*) and *L2 regularization* (Ridge) are usually the most noteworthy. They add penalty terms to the loss function during training, corresponding to the L1-norm or the L2-norm, respectively, of the NN weights, thus tending to reduce the weight size. L1-regularization encourages sparsity by pushing some weights to exactly zero, effectively performing feature selection. L2-regularization penalizes large weights but does not force them to zero, thus usually yielding encourages a smoother weight ditribution.

Other techniques include *early stopping*, which involves monitoring the model performance on a validation dataset during training, and stops the latter when the validation performance starts to degrade (e.g., validation loss increases); reducing model complexity (i.e. the number of layers, neurons, or using a simpler architecture); *dropout* that randomly sets to zero a fraction of neurons in a layer at each forward and backward pass to prevent the network from relying too heavily on any single neuron and encourage it to learn more robust features; or the above-mentioned residual term in the loss function, in the case of PINNs.

2.6.2 Data preprocessing and normalization

Data preprocessing in deep learning is the process of preparing and cleaning raw data to make them suitable for training and feeding into a neural network. It is a crucial step in the machine learning pipeline, as the quality and suitability of the data can significantly impact the performance and effectiveness of the neural network.

This preprocess aims to ensure that the data are in a format that can be effectively used by the neural network for training and inference - i.e. NN evaluation for predictions. The specific preprocessing steps can vary depending on the nature of the data, the task, and the neural network architecture being used. For example, categorical data (non-numeric data) needs to be encoded into a numerical format for neural networks to process. This can be achieved through techniques like *one-hot encoding* (creating binary columns for each category) or *label encoding* (assigning unique numerical values to categories). Proper data preprocessing helps improve model performance, reduce overfitting, and ensure the network can learn meaningful patterns from the data.

In classification tasks, if the classes are imbalanced (one class has significantly fewer samples than others), it may be necessary to apply techniques such as oversampling (creating more samples of the minority class), undersampling (reducing the number of samples in the majority class), or using appropriate loss functions and sampling strategies. In computer vision tasks, data augmentation techniques like rotation, flipping, cropping, and adding noise to images can be used to artificially increase the diversity of the training data and improve model generalization.

For time-series data, appropriate time-based features, such as lag values or moving averages, may be created to capture temporal dependencies.

Scaling the training data to the same order of magnitude is of utmost importance for convergence achievement and acceleration, otherwise the loss function would be stretched along a direction and the algorithm would converge slowly or even miss the convergence. In general, features are scaled to take values from 0 to 1, e.g. with a min-max scaling, or are standardized by removing the mean and scaling to unit variance, since NNs often perform better when the input features are on a similar scale.

2.6.3 Strategies for model evaluation and validation

During training, it is strongly recommended to monitor the NN performance on a separate set of data, called *validation dataset*. This helps assess how well the network generalizes to new data and avoids overfitting. It is a good practice to save a percentage of the available data (usually 30%) to check the efficiency of the model and verify if the problem has overfitted the training data. Once the training is complete, the NN final performance is evaluated on this dataset, called *test dataset*, to provide an unbiased estimate of its capabilities.

Therefore, the dataset is typically divided into training, validation, and test sets. The training set is used to train the model, the validation set helps in hyper-parameter tuning and model selection, and the test set is used to evaluate the model performance on unseen data.

2.6.4 Hyper-parameter tuning

Hyper-parameter tuning is the process of finding the best set of hyper-parameters for a NN to achieve optimal performance on a specific task. Hyper-parameters are parameters that are not learned from the training data, but are set before training begins.

Tuning them is crucial because they greatly influence the training process and its ability to generalize to unseen data.

Here are some of the key hyper-parameters that are often tuned for neural networks:

- Learning rate η that controls the step size of the weight updates during training. A high learning rate may cause the training process to diverge, while a very low learning rate may result in slow convergence (Section 2.2.3). Techniques like learning rate schedules and adaptive learning rate methods - e.g., Adam (Section 2.2.3) - can help find a suitable learning rate during training. The choice itself of optimizer and any learning rate scheduling strategies (e.g., learning rate decay) can affect training speed and convergence too.

- The architecture of the NN, including the number of hidden layers and the number of neurons in each layer, is a critical hyper-parameter. Deeper networks with more neurons can capture complex patterns, but are more prone to overfitting. The choice of architecture depends on the complexity of the task and the available training data. For specific network architectures - like CNNs or RNN -, there exist other hyper-parameters related to their structures to be tuned (e.g., convolutional filter sizes, recurrent sequence lengths).

- The choice of activation functions can significantly impact the NN ability to learn and generalize. Different activation functions are suitable for different types of problems and architectures.

- The batch size determines the number of training examples used in each forward and backward pass during training. Smaller batch sizes can lead to noisy updates, while larger batch sizes may require more memory and computation. The optimal batch size depends on hardware constraints and the dataset.

- Hyper-parameters related to regularization techniques, such as the strength of L1 and L2 regularization (if used), dropout rates, and early stopping criteria, need to be tuned to prevent overfitting.

- The initialization of weights can impact how quickly the network converges and whether it gets stuck in local minima. Techniques like Xavier (Glorot) initialization, [Glorot and Bengio, 2010], and He initialization, [He et al., 2015], are commonly used.

Chapter 3

PINN for forward solution in hydro-poromechanics

Coupled hydro-poromechanics describes important processes occurring in many different fields, in particular in sectors of geomechanics, hydrogeology, and biomechanics. Timely challenges, such as gas, water, CO2, hydrogen injection and/or withdrawal from deep reservoirs, or the interaction between biological tissues and fluid flow, can be accurately described only by considering a fully coupled formulation. In essence, hydro-poromechanics consists of the simultaneous action of fluid flow and solid deformation in fully or partially saturated porous media. The native theory has been developed starting from K. Terzaghi's one-dimensional consolidation in 1925 [Terzaghi, 1925] and the three-dimensional extension by M. A. Biot in 1941 [Biot, 1941] with successive thermodynamically robust formulations available nowadays [Coussy, 1995; Wang, 2000]. Nevertheless, the numerical solution of hydro-poromechanical partial differential equations is still an active subject of research, as they typically involve multi-physics and multi-scale systems, and efficiency and accuracy are challenging tasks. Many numerical strategies, based on either FE and FD methods, have been applied, also in the case of complex materials and high dimensional problems, e.g. [Wang and Hsu, 2009; Ferronato et al., 2010; Castelletto et al., 2015].

Though the above mentioned numerical methods remain the preferred choice as solvers for this system of PDEs, in recent years *Physics-Informed Neural Network* (PINN) gained an increasing attention [Raissi et al., 2019, 2017a,b; Zhang et al., 2019].

The application of PINNs, in their two classical approaches, i.e. direct and inverse problems, has been already investigated in many fields, such as fluid mechanics [Cai et al., 2022; Yang et al., 2019; Lou et al., 2021], solid mechanics [Haghighat et al., 2021; Dourado and Viana, 2020], heat diffusion [Cai et al., 2021], acoustic [Song et al., 2021], additive manufacturing [Zhu et al., 2021], health science [Sahli Costabal et al., 2020; Yin et al., 2021; Kissas et al., 2020]. Also in the field of porous materials, the interest in PINNs has grown rapidly in recent years, focusing in particular on aspects related to fluid diffusion and transport [He et al., 2020; Almajid and Abu-Al-Saud, 2022; Zhang, 2022; Fraces and Tchelepi, 2021; Yang and Foster, 2021; Tartakovsky et al., 2020; Wang et al., 2021a; Bekele, 2021]. However, the PINN training turned out to be a slow and tricky activity, especially for multi-physics problems with coupled governing equations, due to the multi-objective optimization problem resulting from the multiple-term loss function. For these reasons, the use of PINNs for the solution of the coupled hydro-poromechanical problem still presents several issues that deserve attention [Bekele, 2020; Kadeethum et al., 2020b]. For example, to fix this problem, a stress-split sequential training is proposed in [Haghighat et al., 2022] and later applied taking the temperature into account, too, [Amini et al., 2022].

In this chapter we focus on the development of a PINN-based model for hydro-poromechanical applications, in particular governed by Biot's poroelasticity equations, discussing the principal aspects needed for an efficient and robust formulation. Initially, we investigate the role of PINN architecture and hyper-parameter selection in the construction of an effective model. Subsequently, to take advantage of PINN ability to integrate data and reduce the complexity of the solution of the coupled hydro-poromechanical problem, a so-called *sensor-driven* hybrid approach is proposed, where data are provided at locations inside the domain even to solve a forward problem. In other words, the idea is to exploit the possible availability of sensor-measured data in practical applications so as to accelerate the PINN convergence to the problem solution. This strategy, similar to a data assimilation approach, makes PINNs more attractive from a practical point of view. In order to check carefully the accuracy of the proposed approach, the study focuses on well-known benchmark problems, which have been used to assess the numerical implementation and performance.

3.1 PINN solution to Biot's model

We focus on Biot's poroelasticity equations, governing the interaction between a granular material and the fluid filling its pores [Biot, 1941]. The set of PDEs consists of a stress equilibrium equation coupled with a fluid flow equation, which result from a conservation law of linear momentum and mass, respectively. Incorporating Terzaghi's effective stress principle, which links the grain forces to the fluid pore pressure, the equilibrium equation for an isotropic poroelastic medium and the continuity equation for the fluid mass balance can be written as:

$$\mu \Delta \mathbf{u} + (\lambda + \mu)\nabla \mathrm{div}\mathbf{u} = \alpha \nabla p + \mathbf{b}, \tag{3.1}$$

$$-\,\mathrm{div}(\boldsymbol{\kappa}\nabla p) + \frac{\partial}{\partial t}(\phi\beta p + \alpha \mathrm{div}\mathbf{u}) = f, \tag{3.2}$$

where $\mathbf{u}$ is the medium displacement, p is the fluid pore pressure, $\mathbf{b}$ is the body force, and f is a flow source or sink. Equation (3.2) has been obtained by coupling the continuity of pore fluid with Darcy's law:

$$\boldsymbol{\kappa}^{-1}\mathbf{v} + \nabla p = 0, \tag{3.3}$$

where $\mathbf{v}$ is Darcy's velocity. The material parameters are Lamé's moduli λ and μ, the Biot coefficient α, Darcy's conductivity tensor $\boldsymbol{\kappa}$, the medium porosity ϕ, and the fluid compressibility β. As usual, ∇ and Δ denote the gradient and the Laplacian operator, respectively.

Let $\Omega \subset \mathbb{R}^n$ be the domain of the coupled partial differential system in (3.1)-(3.2) and Γ its boundary. The problem of finding the unknowns $\mathbf{u}$ and p is well-posed if proper boundary:

$$\begin{cases} \mathbf{u}(\mathbf{x},t) = \mathbf{u}_D(\mathbf{x},t), & \forall \mathbf{x} \in \Gamma_u, t > 0, \\ \boldsymbol{\sigma}(\mathbf{x},t)\mathbf{n}(\mathbf{x}) = \mathbf{t}_N(\mathbf{x},t), & \forall \mathbf{x} \in \Gamma_\sigma, t > 0, \\ p(\mathbf{x},t) = p_D(\mathbf{x},t), & \forall \mathbf{x} \in \Gamma_p, t > 0, \\ \mathbf{v}(\mathbf{x},t)\cdot\mathbf{n}(\mathbf{x}) = q_N(\mathbf{x},t), & \forall \mathbf{x} \in \Gamma_q, t > 0, \end{cases} \tag{3.4}$$

and initial conditions:

$$\begin{cases} \mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}), & \forall \mathbf{x} \in \Omega \cup \Gamma, \\ p(\mathbf{x},0) = p_0(\mathbf{x}), & \forall \mathbf{x} \in \Omega \cup \Gamma, \end{cases} \tag{3.5}$$

apply. In Equations (3.4) and (3.5), $\Gamma_u \cup \Gamma_\sigma = \Gamma_p \cup \Gamma_q = \Gamma$, with $\Gamma_u \cap \Gamma_\sigma = \Gamma_p \cap \Gamma_q = \emptyset$, $\boldsymbol{\sigma}$ is the total stress tensor, and $\mathbf{n}$ is the outer normal to Γ, while the right-hand side functions $\mathbf{u}_D$, $\mathbf{t}_N$, p_D, q_N, $\mathbf{u}_0$, and p_0 are known.

In our problem, a neural network for each unknown is set up, i.e., one for the pressure, $\hat{p}(\mathbf{x},t)$, and one for every component of the displacement, $\hat{u}_j(\mathbf{x},t)$ with $j = 1, n$. This is motivated by results obtained in [Haghighat et al., 2021], which suggest using many different NNs with single output instead of a unique one with multiple outputs, even if entailing a higher number of parameters. The use of distinct networks is more suitable, as the outputs of the problem have different physical natures. Denoting by $\hat{\mathbf{u}}$ the vector with components $\hat{u}_j$, the loss function reads:

$$\mathcal{L}(\hat{\mathbf{u}},\hat{p}) = \mathcal{L}_d + \sum_{j=1}^{n} \mathcal{L}_{equ,j} + \mathcal{L}_{cont} + \sum_{j=1}^{n} \mathcal{L}_{\Gamma_u,j} + \sum_{j=1}^{n} \mathcal{L}_{\Gamma_\sigma,j} + \mathcal{L}_{\Gamma_p} + \mathcal{L}_{\Gamma_q} + \mathcal{L}_{IC}. \tag{3.6}$$

The terms in the loss function are defined as follows.

The loss on training data $\mathcal{L}_d$ is:

$$\mathcal{L}_d = \sum_{j=1}^{n} w_{d,u,j} \frac{1}{N_d} \sum_{i=1}^{N_d} \|\hat{u}_j(\mathbf{x}_d^i, t_d^i) - u_j^i\|_2^2 + w_{d,p} \frac{1}{N_d} \sum_{i=1}^{N_d} \|\hat{p}(\mathbf{x}_d^i, t_d^i) - p^i\|_2^2, \tag{3.7}$$

where $\{\mathbf{x}_d^i, t_d^i\}_{i=1}^{N_d}$ are the points where data are imposed. The weights $w_{d,\cdot}$ are selected to ensure the dimensional consistency of Equation (3.7) and so as to normalize the set of data $\{p^i, \mathbf{u}^i\}_{i=1}^{N_d}$ between 0 and 1. Also the terms associated to the residual and boundary conditions must be multiplied by suitable weights $w_\cdot$, so that each contribution is comparable to others in the global loss function (3.6). This prevents from a term to largely prevail over the others in the minimization process and generally is a good practice for neural networks training, too. In practice, scaled neural networks $\hat{p}_s$ and $\hat{\mathbf{u}}_s$ are built to fit the scaled data, while the neural network approximations $\hat{p}$ and $\hat{\mathbf{u}}$ are obtained by the inverse denormalization. The contributions $\mathcal{L}_{equ,j}$ and $\mathcal{L}_{cont}$:

$$\begin{aligned}\mathcal{L}_{equ,j} &= w_{equ,j}\frac{1}{N_c}\sum_{i=1}^{N_c}\|\mu(\Delta\hat{\mathbf{u}})_j(\mathbf{x}_c^i,t_c^i)+(\lambda+\mu)(\nabla\mathrm{div}\hat{\mathbf{u}})_j(\mathbf{x}_c^i,t_c^i)-\alpha(\nabla\hat{p})_j(\mathbf{x}_c^i,t_c^i)+\mathbf{b}_j(\mathbf{x}_c^i,t_c^i)\|_2^2,\\ \mathcal{L}_{cont} &= w_{cont}\frac{1}{N_c}\sum_{i=1}^{N_c}\|-\mathrm{div}(\boldsymbol{\kappa}\nabla\hat{p})(\mathbf{x}_c^i,t_c^i)+\frac{\partial}{\partial t}(\phi\beta\hat{p}(\mathbf{x}_c^i,t_c^i)+\alpha\mathrm{div}\hat{\mathbf{u}}(\mathbf{x}_c^i,t_c^i))-f(\mathbf{x}_c^i,t_c^i)\|_2^2,\end{aligned} \tag{3.8}$$

are the loss terms arising from the residual of the governing physical equations on the set of collocation points $\{\mathbf{x}_c^i,t_c^i\}_{i=1}^{N_c}$, whereas the remaining terms in (3.6) are needed to impose the boundary conditions (3.4):

$$\begin{aligned}\mathcal{L}_{\Gamma_u,j} &= w_{\Gamma_u,j}\frac{1}{N_{\Gamma_u}}\sum_{i=1}^{N_{\Gamma_u}}\|\hat{u}_j(\mathbf{x}_{\Gamma_u}^i,t_{\Gamma_u}^i)-u_{D,j}(\mathbf{x}_{\Gamma_u}^i,t_{\Gamma_u}^i)\|_2^2,\\ \mathcal{L}_{\Gamma_\sigma,j} &= w_{\Gamma_\sigma,j}\frac{1}{N_{\Gamma_\sigma}}\sum_{i=1}^{N_{\Gamma_\sigma}}\|(\hat{\boldsymbol{\sigma}}(\mathbf{x}_{\Gamma_\sigma}^i,t_{\Gamma_\sigma}^i)\mathbf{n}(\mathbf{x}_{\Gamma_\sigma}^i))_j-\mathbf{t}_{N,j}(\mathbf{x}_{\Gamma_\sigma}^i,t_{\Gamma_\sigma}^i)\|_2^2,\\ \mathcal{L}_{\Gamma_p} &= w_{\Gamma_p}\frac{1}{N_{\Gamma_p}}\sum_{i=1}^{N_{\Gamma_p}}\|\hat{p}(\mathbf{x}_{\Gamma_p}^i,t_{\Gamma_p}^i)-p_D(\mathbf{x}_{\Gamma_p}^i,t_{\Gamma_p}^i)\|_2^2,\\ \mathcal{L}_{\Gamma_q} &= w_{\Gamma_q}\frac{1}{N_{\Gamma_q}}\sum_{i=1}^{N_{\Gamma_q}}\|\boldsymbol{\kappa}\nabla\hat{p}(\mathbf{x}_{\Gamma_q}^i,t_{\Gamma_q}^i)\cdot\mathbf{n}(\mathbf{x}_{\Gamma_q}^i)-q_N(\mathbf{x}_{\Gamma_q}^i,t_{\Gamma_q}^i)\|_2^2,\end{aligned} \tag{3.9}$$

and the initial conditions (3.5):

$$\mathcal{L}_{IC}=\sum_{j=1}^{n}w_{IC,u,j}\frac{1}{N_{IC}}\sum_{i=1}^{N_{IC}}\|\hat{u}_j(\mathbf{x}^i,0)-u_{0,j}(\mathbf{x}^i)\|_2^2+w_{IC,p}\frac{1}{N_{IC}}\sum_{i=1}^{N_{IC}}\|\hat{p}(\mathbf{x}^i,0)-p_0(\mathbf{x}^i)\|_2^2. \tag{3.10}$$

Here, $\hat{\boldsymbol{\sigma}}=\boldsymbol{\sigma}(\hat{\mathbf{u}})$ indicates the approximation of the total stress, that is defined by the material constitutive law, $\{\mathbf{x}_{\Gamma_u}^i,t_{\Gamma_u}^i\}_{i=1}^{N_{\Gamma_u}}$, $\{\mathbf{x}_{\Gamma_\sigma}^i,t_{\Gamma_\sigma}^i\}_{i=1}^{N_{\Gamma_\sigma}}$, $\{\mathbf{x}_{\Gamma_p}^i,t_{\Gamma_p}^i\}_{i=1}^{N_{\Gamma_p}}$ and $\{\mathbf{x}_{\Gamma_q}^i,t_{\Gamma_q}^i\}_{i=1}^{N_{\Gamma_q}}$ are points on the boundary, and $\{\mathbf{x}^i,0\}_{i=1}^{N_{IC}}$ are points in the domain Ω at $t=0$.

The presented PINN formulation is used in the next sections to solve two classical benchmarks in coupled hydro-poromechanics.

3.2 Identification of PINN architecture

In this section, we investigate the optimal construction of PINNs for two classical coupled problems, namely Terzaghi's problem (1D) and Mandel's problem (2D), for which analytical solutions are available. In particular, we analyze the optimal choice of the hyper-parameters and evaluate the accuracy of the PINN model used as a forward solver for the governing differential problem.

It is well-known that an established procedure to choose the architecture of a neural network does not exist, but it mainly remains at the modeler's experience. A general rule of thumb is the deeper the network, the higher the ability to capture complex links, but of course the slower the training time. Many factors can come into play, influencing in different ways the difficulty of the procedure. Some of them are specifically problem-dependent, e.g., the number of data and collocation points, or the hyper-parameters regulating the training, such as batch size and number of epochs. However, other factors basically depend on the class of problem and the structure of the governing equations. Driven by these motivations, we extensively analyze the founding elements of the architecture of PINNs for coupled hydro-poromechanics, with the aim at investigating its robustness and accuracy with respect to the hyper-parameter set selection.

Among the hyper-parameters needed to set-up a PINN model, the most influential ones are:

- number of layers,
- number of neurons per layer,
- type of activation function.

For each item, we consider the set of possible entries reported in Table 3.1 with the effect of all possible combinations.

For the sake of simplicity, we assume that all hidden layers have the same number of neurons and the same activation function. For PINN models, it is preferable to use differentiable activation functions, because the neural networks have to approximate a smooth PDE solution and are automatically differentiated to compute the loss terms (3.8). For this reason, along with $\tanh$, we use the elu activation function, defined as:

$$elu(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}, \tag{3.11}$$

with $\alpha = 1$, instead of the more common *Rectified Linear Unit* (ReLU) function.

TABLE 3.1: Hyper-parameter set of values.

Hyper-parameter	Description	Set
Layers	Number of hidden layers	$\{4, 8, 12\}$
Neurons	Number of neurons per layer	$\{20, 40\}$
Act fun	Activation function of each neuron in the hidden layers	$\{\tanh, elu\}$

In the following sections, we present the results obtained for a one-dimensional and a two-dimensional formulation of the problem, comparing the accuracy of the approximations for the different combinations of hyper-parameters.

A full factorial combination of the different values for the number of layers, neurons and the activation function type has been considered, with the networks obtained by each architecture properly trained. The quality of every realization is evaluated by computing the weighted L2-norm of the difference between the NN prediction, $\hat{\mathbf{y}}$, and the analytical solution, $\mathbf{y}$:

$$E(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\|\hat{\mathbf{y}} - \mathbf{y}\|_2}{\|\mathbf{y}\|_2} \tag{3.12}$$

The use of such weighted norm allows to deal with dimensionless and comparable errors. Then, the population of $E(\hat{\mathbf{y}}, \mathbf{y})$ data is statistically processed in order to identify the impact of the hyper-parameters, their mutual influence, the robustness of the proposed PINN architecture with respect to their variation, and the setup providing the most accurate outcome.

Different libraries have been recently developed to implement the PINN construction, e.g. [Hennigh et al., 2021; Lu et al., 2021; Koryagin et al., 2019; Chen et al., 2020a]. The model considered in this work has been implemented using SciANN (Scientific Computational with Artificial Neural Networks), a recent TensorFlow and Keras wrapper specific to scientific computations with PINN [Haghighat and Juanes, 2021]. This *Application Programming Interface* (API) makes it possible to build in a quite simple way the PINN structure with a readable code and to use automatic differentiation to compute at machine precision the derivatives of the neural network approximation of the solution $\mathbf{u}$. Originally designed to optimize the *Gradient Descent* (GD) algorithm in the NN training automatic differentiation turned out to be extremely useful in PINN implementation [Baydin et al., 2015]. The possibility of using automatic differentiation has been inherited by SciANN from TensorFlow and Keras, resulting in faster and more robust

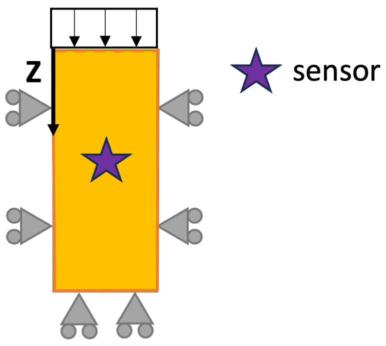


FIGURE 3.1: Sketch of the setup for Terzaghi's problem with indication of sensor location.

codes, as it avoids the numerical discretization and the related rounding error propagation. This is particularly attractive when dealing with noisy data, usually resulting from real measurements. The library also supports running computations on a variety of devices, including CPUs and GPUs.

3.2.1 1D: Terzaghi's problem

Terzaghi's problem consists of a poroelastic fluid-saturated column with a constant loading P_H applied instantaneously on top at time $t = 0$, as shown schematically in Figure 3.1. The column has height H and at the basement there are zero flux and null displacement. Only through the upper boundary free drainage is allowed. In a one-dimensional configuration and assuming the z-axis positive downward (Figure 3.1), equations (3.1) and (3.2) read:

$$(\lambda + 2\mu)\frac{\partial^2 u}{\partial z^2} = \alpha \frac{\partial p}{\partial z}, \tag{3.13}$$

$$-\kappa \frac{\partial^2 p}{\partial z^2} + \frac{\partial}{\partial t}\Big(\phi\beta p + \alpha \frac{\partial u}{\partial z}\Big) = 0, \tag{3.14}$$

with the following boundary conditions:

$$\begin{aligned} &p(0,t) = 0, && (\lambda + 2\mu)\frac{\partial u}{\partial z}(0,t) = -P_H, && z = 0, \\ &\frac{\partial p}{\partial z}(H,t) = 0, && u(H,t) = 0, && z = H. \end{aligned} \tag{3.15}$$

TABLE 3.2: Material parameters definition.

Parameter	Name	Unit	Definition
M	Biot modulus	MPa	$M = [\phi\beta + (\alpha - \phi)c_{br}]^{-1}$
K_u	Undrained bulk modulus	MPa	$K_u = \lambda + 2\mu/3 + \alpha^2 M$
c_M	Vertical uniaxial compressibility	MPa^{-1}	$c_M = [\lambda + 2\mu]^{-1}$
c	Consolidation coefficient	m^2/s	$c = k/[\gamma_w(M^{-1} + \alpha^2 c_M)]$
B	Skempton's coefficient	–	$B = \alpha M/K_u$
ν_u	Undrained Poisson's ratio	–	$\nu_u = [3\nu + \alpha B(1-2\nu)]/[3 - \alpha B(1-2\nu)]$

The initial overpressure $p_0(z)$ and displacement $u_0(z)$ caused by the instant load P_H read [Wang, 2000; Wang and Hsu, 2009; Ferronato et al., 2010; Castelletto et al., 2015]:

$$p_0(z) = \begin{cases} 0 & z = 0 \\ \dfrac{\alpha M}{K_u + 4\mu/3} P_H & \text{otherwise} \end{cases}, \qquad (3.16)$$
$$u_0(z) = \frac{1}{K_u + 4\mu/3} P_H (H - z),$$

and the analytical solutions are:

$$p(z,t) = \frac{4}{\pi} p_0 \sum_{m=0}^{\infty} \frac{1}{2m+1} \exp\Big[-\Big(\frac{(2m+1)\pi}{2H}\Big)^2 ct\Big] \sin\Big[\frac{(2m+1)\pi z}{2H}\Big], \qquad (3.17)$$
$$u(z,t) = c_M p_0 \Big\{ -\frac{8H}{\pi^2} \sum_{m=0}^{\infty} \frac{1}{(2m+1)^2} \exp\Big[-\Big(\frac{(2m+1)\pi}{2H}\Big)^2 ct\Big] \cos\Big[\frac{(2m+1)\pi z}{2H}\Big] + (H - z) \Big\} + u_0.$$

The definition of the material parameters arising in (3.16) and (3.17) is summarized in Table 3.2, where c_{br} denotes the solid grain compressibility, $k = k_s$ the hydraulic conductivity, γ_w the fluid specific weight, ν the drained Poisson ratio, and $\lambda = \lambda_s$ and $\mu = \mu_s$ the Lamé constants. The poroelastic medium is supposed to be homogeneous with $H = 15$m and $P_H = 10^{-2}$MPa. All material parameter values are reported in Table 3.3.

In the 1D formulation (3.13)-(3.14), the unknown functions are the fluid pore pressure p and the vertical displacement u. As stated in Section 3.1, they are approximated by two distinct neural networks, with the loss function built following (3.6). The terms of the loss function give their contribution on different portions of the space-time domain $[0, H] \times [0, +\infty)$. For this reason, training points have to be properly distributed

TABLE 3.3: Material properties.

Parameter	Name	Value
λ_s	Lamé constant (sand)	40 MPa
μ_s	Lamé constant (sand)	40 MPa
λ_c	Lamé constant (clay)	4 MPa
μ_c	Lamé constant (clay)	4 MPa
α	Biot coefficient	1.0
ϕ	Medium porosity	0.375
β	Fluid compressibility	$4.4 \cdot 10^{-4}$ MPa^{-1}
k_s	Hydraulic conductivity (sand)	10^{-5} m/s
k_c	Hydraulic conductivity (clay)	10^{-7} m/s
γ_w	Fluid specific weight	$9.81 \cdot 10^{-3}$ MN/m^3
ν	Drained Poisson ratio	0.3
c_{br}	Solid grain compressibility	0 MPa^{-1}

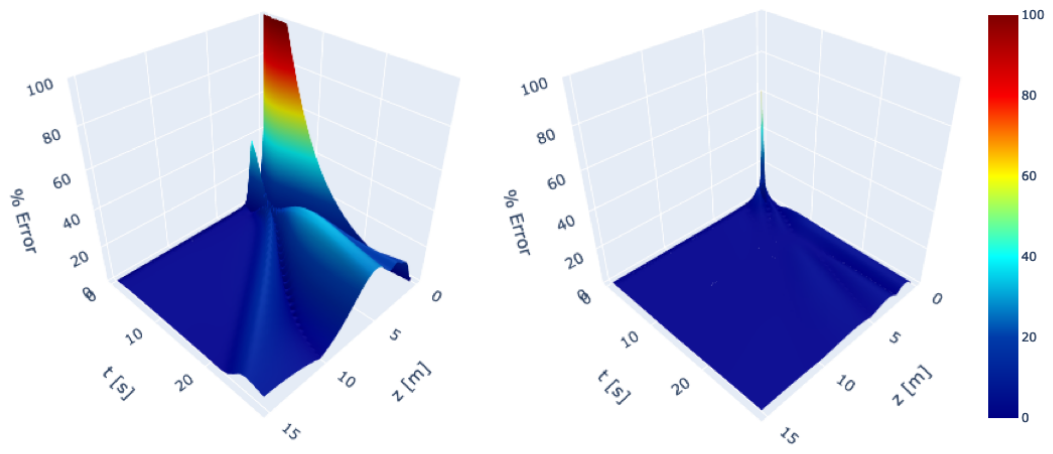


FIGURE 3.2: Terzaghi's problem: relative L^2-norm pressure error (equation (3.12)) with a uniform (left) and logarithmic (right) training point distribution in time.

to cover the domain completely. They are generated with DataGeneratorXT of SciANN which builds a grid with a number of points distributed in both the domain and the boundaries, so as to guarantee that each loss term is proportionally sampled and the optimizer performs better.

The numerical time domain is bounded at $T = 1000$s, which represents a value where steady state conditions are practically achieved for the selected material parameters. First of all, we notice that a uniform distribution of training points in time is not effective, since significant oscillations of the initial boundary bands can be observed

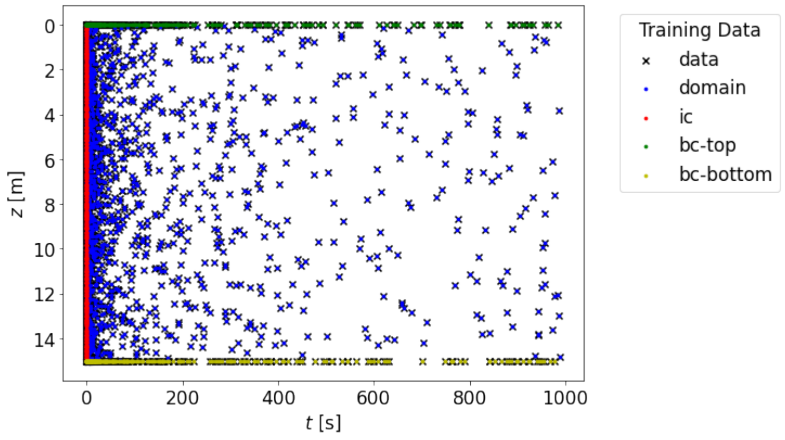


FIGURE 3.3: Terzaghi's problem: training point distribution in the space-time domain. Blue ones are collocation points, while green, red and black ones are used to impose initial and boundary conditions at $z = 0$ and $z = H$, respectively. Pressure and displacement data are given on them all.

(Figure 3.2). Such sharp oscillations are substantially smoothed generating a logarithmic random sample in time. This is motivated by the shape of the analytical solution (3.17) that is exponentially decaying in time. This behavior is common for every coupled hydro-poromechanical problem, independently of application dimension and the specific boundary conditions.

The total number N_d of training data is set to 3000, half of which is equally distributed between boundary and initial conditions. This is an empirical choice balancing the need of limiting the computational burden of the training processes and the size of the approximation errors. Therefore, we suppose to have pressure and displacement data $\{p^i, u^i\}_{i=1}^{N_d}$ obtained by evaluation of the analytical solutions (3.17) over training points $\{z_d^i, t_d^i\}_{i=1}^{N_d}$ spread all over the domain. The points located inside the domain are used as collocation points, while the ones on the space-time boundaries allow to evaluate (3.15) and (3.16), thus resulting in $N_c = 1500$, $N_{IC} = 750$, and $N_{BC} = 750$ split into top and bottom (Figure 3.3).

We consider all the possible 144 architectures generated by varying the number of layers in $\{4, 8, 12\}$, the number of neurons in $\{20, 40\}$ and the type of the activation function, either $\tanh$ or elu, for both networks (Table 3.1). The number of epochs is set to 5000 with a batch size of 500. Note that the size of the batches has to be quite big so as to contain points over the boundaries too. This is necessary because the gradient updates have to consider also information for the boundary terms of the loss function (3.6). An early stopping is added to quit the training if the loss function does not improve for 500 epochs. Adam's algorithm is used with a learning rate constantly equal to 0.001 until the 2000th epoch and then exponentially decreasing to 0.0001 until the last epoch [Brownlee, 2016]. The weights of both the neural networks are initialized with Glorot normal initializer of Keras.

As mentioned before, to improve the training procedure all data have been normalized between 0 and 1 by Scikit-learn Min-Max scaler. First, the scaled networks $\hat{p}_s$ and $\hat{u}_s$ have been trained over these scaled data, and then denormalized to obtain $\hat{p}$ and $\hat{u}$:

$$\begin{aligned} \hat{p} &= \hat{p}_s p_{sc} + p_{min}, \\ \hat{u} &= \hat{u}_s u_{sc} + u_{min}, \end{aligned} \tag{3.18}$$

where $p_{sc} = p_{max} - p_{min}$, $u_{sc} = u_{max} - u_{min}$, with p_{min}, p_{max} and u_{min}, u_{max} the minimum and maximum pressure and displacement values, respectively. Also, each term of the loss function is multiplied by an appropriate weighting factor to balance their role in the training process. In this case, the weights have been set as follows: $w_{equ,z} = w_{\Gamma_p} = p_{sc}$, $w_{cont} = w_{\Gamma_{u},z} = u_{sc}$, $w_{\Gamma_{\sigma},z} = P_H$, $w_{\Gamma_q} = 100 p_{sc}$, $w_{IC,u,z} = \frac{1}{K_u + 4\mu/3} P_H H$, and $w_{IC,p} = \frac{\alpha M}{K_u + 4\mu/3} P_H$. The choice is also motivated by the different unit of measure of the loss function contributions, see (3.13)-(3.16). Notice that w_{Γ_q} is magnified 100 times with respect to other similar weights to better enforce boundary conditions at $z = H$. The seed generating the random points is the same in all simulations in order to have reproducible and comparable results.

To evaluate the accuracy, the trained networks are computed over a uniform grid with 1501×1001 points in the $z - t$ domain $[0, 15]$m $\times$ $[0, 1000]$s. The errors are evaluated by numerically computing the dimensionless weighted L^2-norm (3.12) over such a uniform grid and plotted in Figure 3.4 for all 144 architectures. Mean value and standard deviation of the errors are equal to $\mu_p = 3.291 \times 10^{-2}$ and $\sigma_p = 1.719 \times 10^{-2}$ for p, $\mu_u = 5.622 \times 10^{-2}$ and $\sigma_u = 3.327 \times 10^{-2}$ for u. Overall, we can observe that the PINN accuracy appears to be quite satisfactory for almost any of the investigated architectures. The displacement errors are larger than the pressure ones on average, with a similar statistical distribution. It is interesting to note that for both p and u the smallest

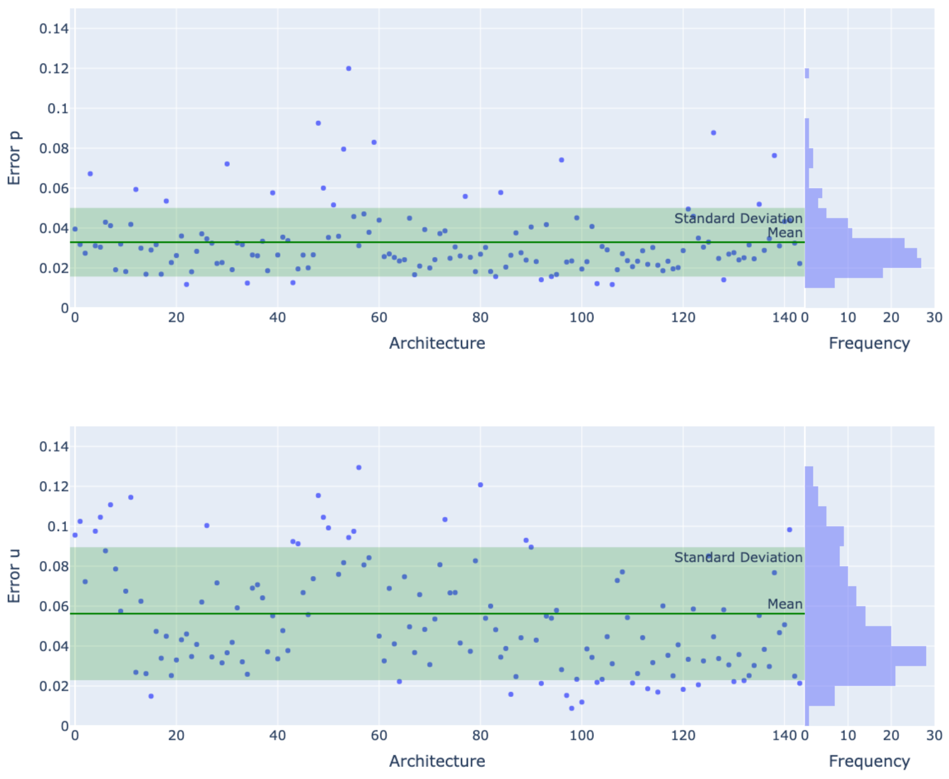


FIGURE 3.4: Terzaghi's problem: errors in the pressure (top) and displacement (bottom) reconstruction for the different architectures. On the rightmost frames, the error statistical distribution is provided.

errors occur in architecture indices between 100 and 120, which correspond to the use of elu activation function, 12 layers and 20 neurons for u. These plots highlight the robustness of the PINN method, that for a large number of combinations gives similar and satisfactory error values. Outliers are quite few and even in the worse cases the errors are at most around 12%. The full factorial combination of the values in Table 3.1 have been done with three repetitions for each architecture, obtained using three different random seeds. Graphs in Figure 3.4 preserve the same allocation of the errors for all the different seeds, so that it is possible to conclude that the choice of the seed has a negligible impact.

Figure 3.5 contains the main effect plots of the mean errors obtained with the different hyper-parameters. They show that the strongest impact on the PINN accuracy is given by the choice of the activation function for $\hat{u}$ and the number of its neurons, with elu and 20 neurons being more effective. The choices for the $\hat{p}$ architecture are not so significant for the accuracy in the displacement approximation. Conversely, for

TABLE 3.4: Response optimization results.

	Layers p	Layers u	Neurons p	Neurons u	Act fun p	Act fun u
Terzaghi	10	12	40	20	elu	elu
Mandel	4	8	20	20	tanh	tanh

pressure approximation elu is again more appropriate. It is also possible to conclude that a suitable value for the number of layers of both the NNs is between 8 and 12.

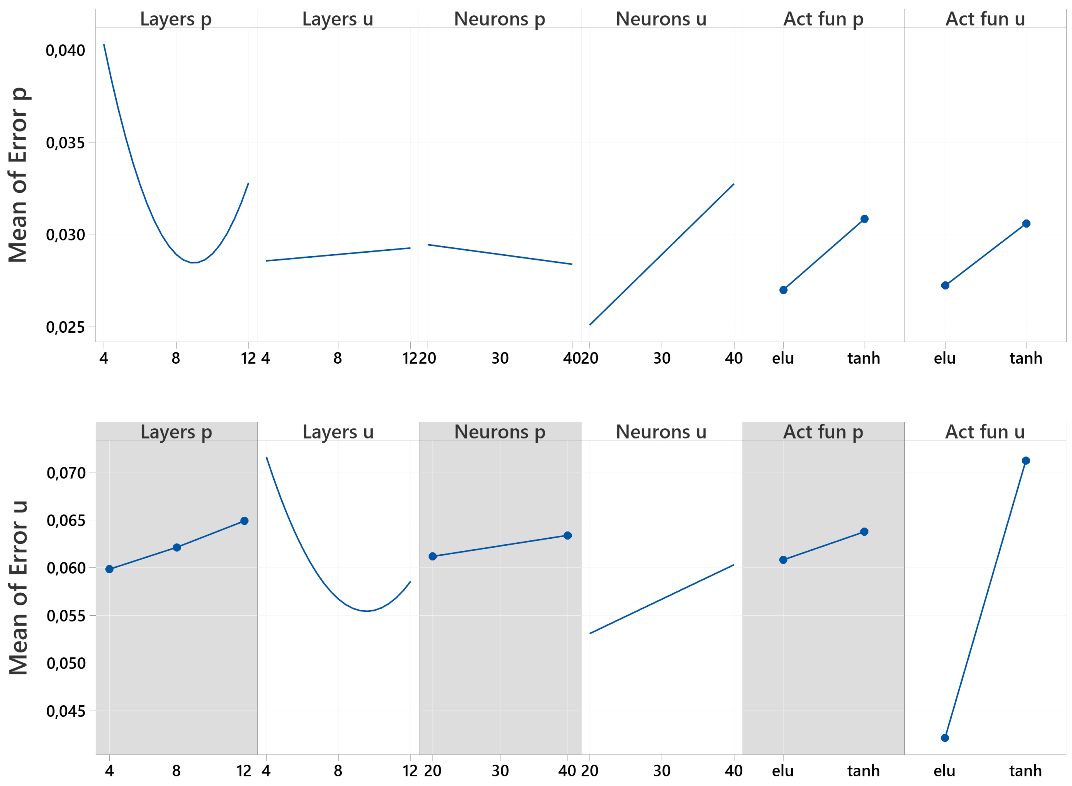


FIGURE 3.5: Terzaghi's problem: main effect plots of the sensitivity analysis on all hyper-parameters for the pressure (top) and displacement (bottom) approximations. A grey background denotes a term not statistically significant in the model.

Finally, a classical response optimization process [Box and Wilson, 1951], applied to both p- and u-error provides the architecture in Table 3.4 as the best one. The response optimization is ideal for optimizing and exploring deployed predictive data mining models. It performs a discrete search in the independent variable space, so as to enrich the plumbed space of the hyper-parameters, until a set of independent values are discovered for which the model yield minimizes the responses. Note that the optimal values for $\hat{u}$ are elu activation function, 12 layers and 20 neurons, as confirmed also from Figure 3.4 and 3.5.

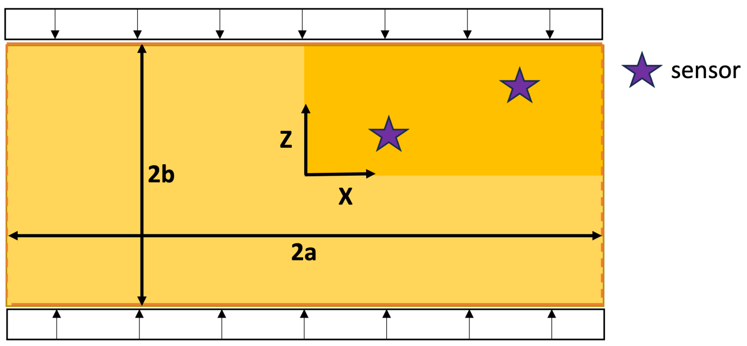


FIGURE 3.6: Sketch of the setup for Mandel's problem with indication of sensor locations, only in the portion of the domain considered for symmetry reasons.

3.2.2 2D: Mandel's problem

Mandel's problem simulates the consolidation of an infinitely long poroelastic slab, which is sandwiched between two rigid, impermeable and frictionless plates. The poroelastic fluid-saturated slab is homogeneous and isotropic and both plates are suddenly squeezed at time $t = 0$ by a constant compressive vertical load per unit length $2F$ (Figure 3.6). Let $2a$ and $2b$ be the slab sizes. The left and right boundaries ($x = \pm a$) are stress-free, drained and always at ambient pressure ($p = 0$), while top and bottom boundaries ($z = \pm b$) have a prescribed stress and no flux. As the domain is symmetric, only the highlighted quarter of the $x - z$ plane in Figure 3.6 is considered.

Equations (3.1) and (3.2) in two dimensions for Mandel's configuration take the form:

$$\begin{aligned}(\lambda+2\mu)\frac{\partial^2 u_x}{\partial x^2}+\mu\frac{\partial^2 u_x}{\partial z^2}+(\lambda+\mu)\frac{\partial^2 u_z}{\partial x\partial z}&=\alpha\frac{\partial p}{\partial x},\\ \mu\frac{\partial^2 u_z}{\partial x^2}+(\lambda+2\mu)\frac{\partial^2 u_z}{\partial z^2}+(\lambda+\mu)\frac{\partial^2 u_x}{\partial x\partial z}&=\alpha\frac{\partial p}{\partial z},\end{aligned} \tag{3.19}$$

$$-\kappa\left(\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial z^2}\right)+\frac{\partial}{\partial t}\left(\phi\beta p+\alpha\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_z}{\partial z}\right)\right)=0, \tag{3.20}$$

where the Darcy conductivity $\boldsymbol{\kappa}$ is assumed to be the identity tensor $\mathbf{1}$ multiplied by the scalar κ. The boundary conditions read:

$$\begin{array}{llll}
\frac{\partial p}{\partial x}(0,z,t)=0, & u_x(0,z,t)=0, & \frac{\partial u_z}{\partial x}(0,z,t)=0, & x=0,\\
p(a,z,t)=0, & \frac{\partial u_x}{\partial x}(a,z,t)=\frac{F\nu}{2\mu a}, & \frac{\partial u_z}{\partial x}(a,z,t)=0, & x=a,\\
\frac{\partial p}{\partial z}(x,0,t)=0, & \frac{\partial u_x}{\partial z}(x,0,t)=0, & u_z(x,0,t)=0, & z=0,\\
\frac{\partial p}{\partial z}(x,b,t)=0, & \frac{\partial u_x}{\partial z}(x,b,t)=0, & \frac{\partial u_z}{\partial z}(x,b,t)=-\frac{F(1-\nu)}{2\mu a}, & z=b.
\end{array} \tag{3.21}$$

The instantaneous load application at $t=0$ causes the initial overpressure $p_0(x,z)$ and horizontal and vertical displacements $u_{x,0}(x,z)$ and $u_{z,0}(x,z)$ [Ferronato et al., 2010; Castelletto et al., 2015]:

$$\begin{aligned}
p_0(x,z) &= \begin{cases} 0 & x=a \\ \frac{1}{3a}B(1+\nu_u)F & \text{otherwise} \end{cases},\\
u_{x,0}(x,z) &= \frac{F\nu_u}{2\mu}\frac{x}{a},\\
u_{z,0}(x,z) &= -\frac{F(1-\nu_u)}{2\mu}\frac{z}{a}.
\end{aligned} \tag{3.22}$$

The analytical solution reads [Castelletto et al., 2015]:

$$\begin{aligned}
p(x,z,t) &= 2p_0\sum_{n=1}^{\infty}\frac{\sin\alpha_n}{\alpha_n-\sin\alpha_n\cos\alpha_n}\left(\cos\frac{\alpha_n x}{a}-\cos\alpha_n\right)\exp\left(-\frac{\alpha_n^2 ct}{a^2}\right),\\
u_x(x,z,t) &= \left[\frac{F\nu}{2\mu a}-\frac{F\nu_u}{\mu a}\sum_{n=1}^{\infty}\frac{\sin\alpha_n\cos\alpha_n}{\alpha_n-\sin\alpha_n\cos\alpha_n}\exp\left(-\frac{\alpha_n^2 ct}{a^2}\right)\right]x\\
&+\frac{F}{\mu}\sum_{n=1}^{\infty}\frac{\cos\alpha_n}{\alpha_n-\sin\alpha_n\cos\alpha_n}\sin\frac{\alpha_n x}{a}\exp\left(-\frac{\alpha_n^2 ct}{a^2}\right),\\
u_z(x,z,t) &= \left[-\frac{F(1-\nu)}{2\mu a}+\frac{F(1-\nu_u)}{\mu a}\sum_{n=1}^{\infty}\frac{\sin\alpha_n\cos\alpha_n}{\alpha_n-\sin\alpha_n\cos\alpha_n}\exp\left(-\frac{\alpha_n^2 ct}{a^2}\right)\right]z,
\end{aligned} \tag{3.23}$$

with α_n the positive roots of the non-linear equation:

$$\tan\alpha_n=-\frac{\nu-1}{\nu_u-\nu}\alpha_n. \tag{3.24}$$

See Table 3.2 and 3.3 for the material parameter definitions and values. We choose $a = b = 1$m and the force F equal to 10^{-2}MN/m.

In Mandel's problem there are three unknown functions, i.e., fluid pressure $p(x, z, t)$, horizontal displacement $u_x(x, z, t)$, and vertical displacement $u_z(x, z, t)$. Each one is approximated by a neural network with the loss function built as in (3.6), so as to honour both the governing equations (3.19)-(3.20) and the auxiliary conditions in (3.21)-(3.22). The problem domain in space and time is $\Omega = [0, a] \times [0, b] \times [0, +\infty)$. Given the material parameter values of Table 3.3, the domain is limited along the time direction to $T = 10$s. Recalling the observations made for Terzaghi's problem, we empirically select $N_d = 6000$ data points using SciANN DataGeneratorXYT to build the random samples $\{x_d^i, z_d^i, t_d^i\}_{i=1}^{N_d}$ with a logarithmic scale in time. The procedure is forced to produce half training points inside the domain and the remaining ones equally distributed along the boundaries. In particular, 375 points are set on each side of the space domain boundary and 1500 at $t = 0$.

The same hyper-parameter domain as Terzaghi's problem (Table 3.1) is considered for the sensitivity analysis. We use 5000 epochs with a batch size set equal to 1000 and the learning rate of Adam's optimization algorithm to 0.001 for the first 2000 epochs, then exponentially decreasing to 0.0001 until the last epoch. As for Terzaghi's problem, the weights of the networks are initialized with Glorot normal initialization and the training data are scaled with the min-max scaler. Then, three NNs, namely $\hat{p}_s$, $\hat{u}_{x,s}$, and $\hat{u}_{z,s}$, are built to fit the scaled data and other three networks, $\hat{p}$, $\hat{u}_x$, and $\hat{u}_z$, are derived as in (3.18). The values for the remaining weights are: $w_{equ,x} = w_{equ,z} = w_{\Gamma_p} = w_{\Gamma_q} = p_{sc}$, $w_{cont} = w_{\Gamma_{u,x}} = u_{x,sc}$, $w_{\Gamma_{u,z}} = u_{z,sc}$, $w_{\Gamma_{\sigma,x}} = w_{\Gamma_{\sigma,z}} = \frac{F}{2\mu}$, $w_{IC,u,x} = \frac{F\nu_u}{2\mu}$, $w_{IC,u,z} = -\frac{F(1-\nu_u)}{2\mu}$, and $w_{IC,p} = \frac{B(1+\nu_u)F}{3a}$.

After training, the network accuracy is evaluated on an equally spaced grid in $[0, 1]m \times [0, 1]m \times [0, 10]s$ of $101 \times 101 \times 201$ points. The L^2-errors of Equation (3.12) are plotted in Figure 3.7. As for Terzaghi's problem, the errors denote quite a limited spread, providing evidence again of the PINN robustness. It appears also that displacements are very marginally affected by the hyper-parameter selection, with the worse outliers characterized by an error around 7%. By distinction, there exists a marked difference for the fluid pressure, where the realizations which correspond to the tanh

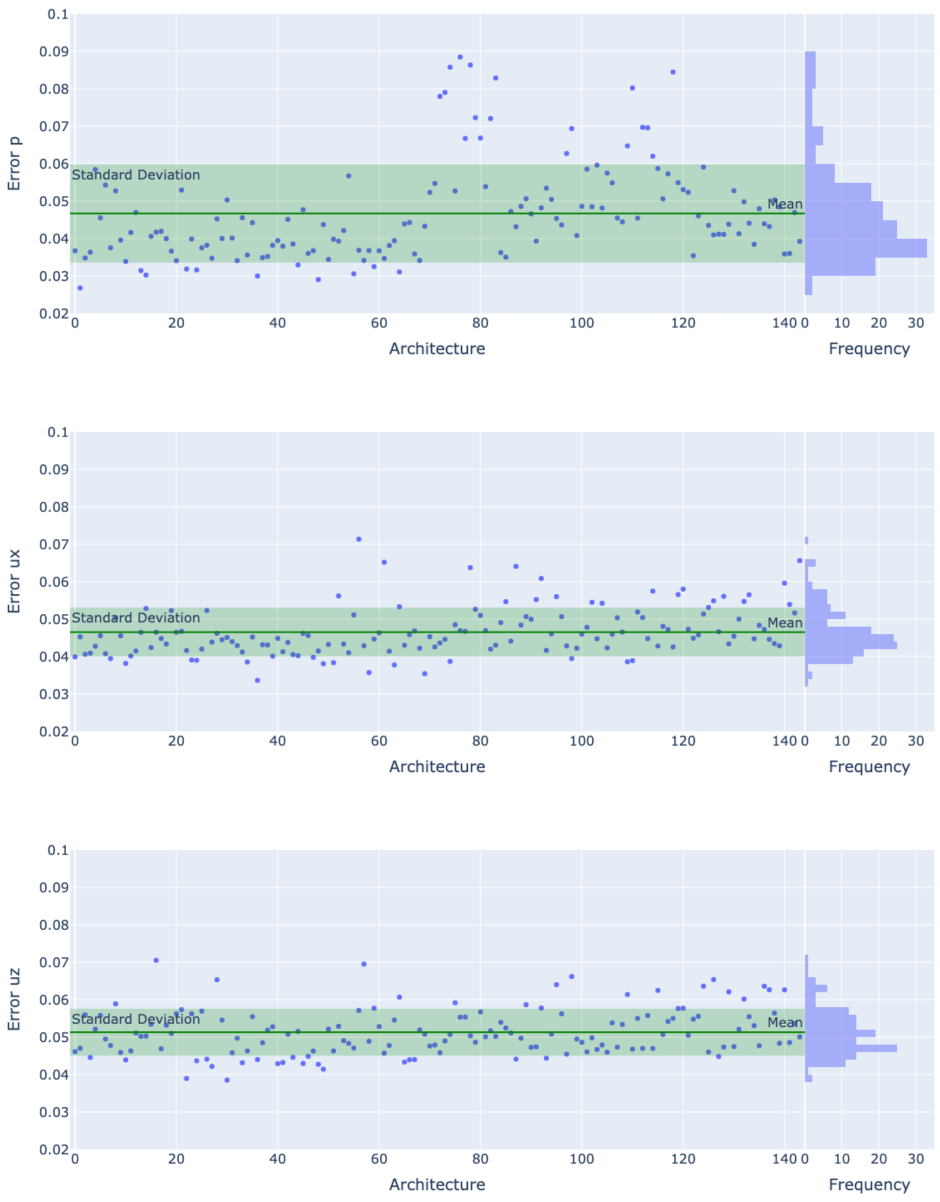


FIGURE 3.7: Mandel's problem: errors in the pressure (top), horizontal (middle) and vertical displacement (bottom) reconstruction for the different architectures. On the rightmost frames, the error statistical distribution is provided.

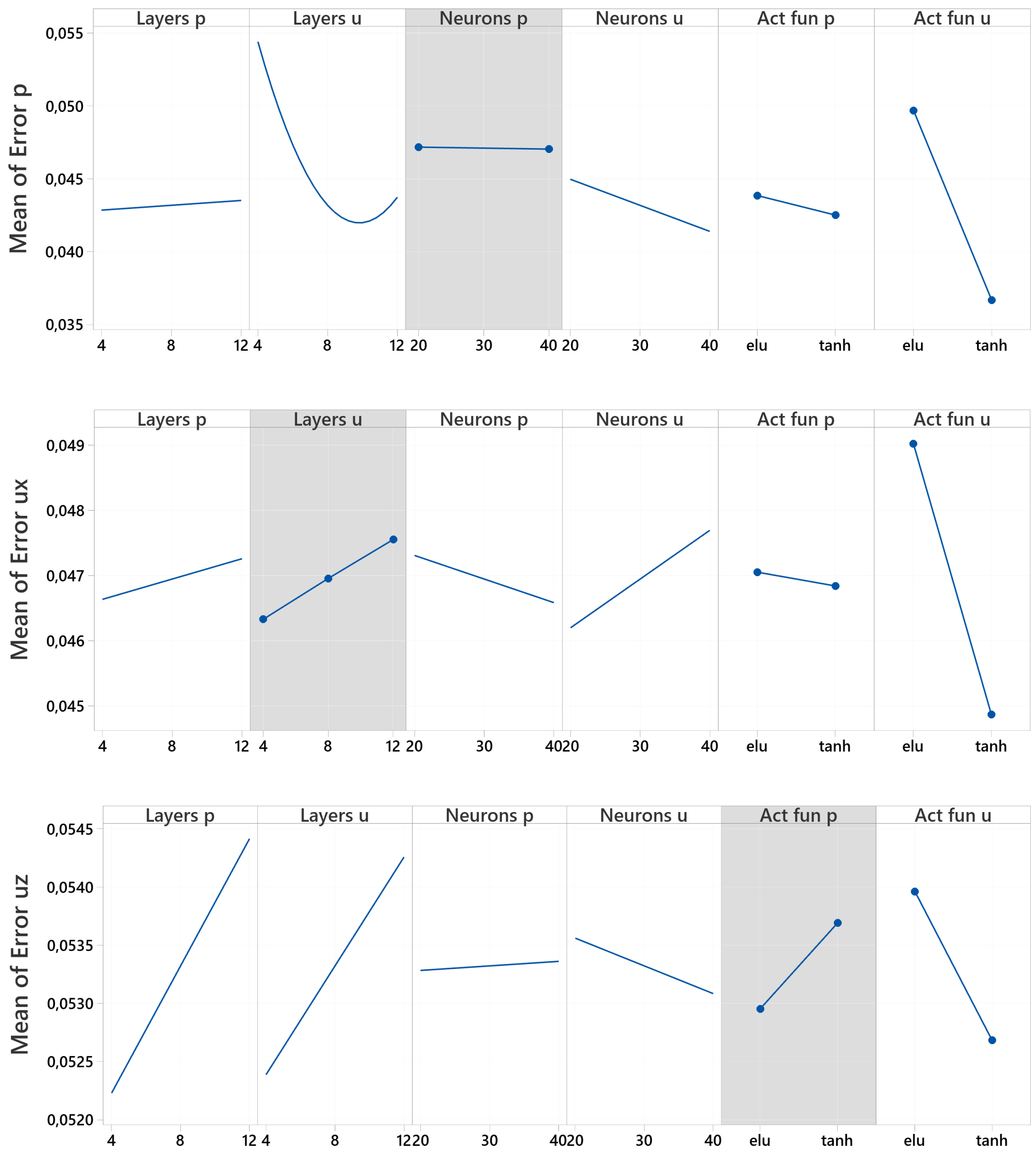


FIGURE 3.8: Mandel's problem: main effect plots of the sensitivity analysis on all hyper-parameters for the pressure (top), horizontal (middle) and vertical displacement (bottom) approximation. A grey background denotes a term not statistically significant in the model.

activation function for displacements (from no. 1 to no. 72) exhibit a more consistent behavior than the remaining ones. Means and standard deviations of the errors are $\mu_p = 4.671 \times 10^{-2}$, $\mu_{u_x} = 4.653 \times 10^{-2}$, $\mu_{u_z} = 5.131 \times 10^{-2}$, and $\sigma_p = 1.314 \times 10^{-2}$, $\sigma_{u_x} = 6.522 \times 10^{-3}$, $\sigma_{u_z} = 6.268 \times 10^{-3}$, i.e. the same order of magnitude as Terzaghi's problem. As for the 1D case, this chart shows the robustness of the method, since around 70% of the possible architectures leads to errors inside tight $\mu_. \pm \sigma_.$ intervals.

The effect plots for Mandel's problem are shown in Figure 3.8. According to these diagrams the activation function of the displacements is the hyper-parameter mostly affecting the overall network behavior. In this case, it appears preferable to set it at $\tanh$. The other hyper-parameter showing an appreciable influence on the network accuracy is the number of layers for both displacement and pressure, the latter exclusively for vertical displacement errors. The response optimization process gives the architecture provided in Table 3.4.

3.3 PINN application with a sensor-driven condition

The quality and effectiveness of the PINN architectures identified in the previous section are tested on a more challenging situation, with the aim of applying a PINN strategy to something closer to a real-world application. First, we use the PINN model as a standard forward PDE solver with no other information but the initial and boundary conditions. Second, we add a few pieces of information as training data, arising from a synthetic "sensor" installed at some fixed locations to monitor the evolution of the process. This is quite a common situation that may occur in real-world applications, where data can be recorded at some points of the space-time domain. Third, to better replicate the condition driven by measured data, we add some noise to the synthetic data. The key idea here is to show that even a few training data points from a very limited subset of the domain can substantially improve the computational efficiency of the numerical procedure for minimizing the loss function in the forward PINN solution of a PDE system. At the same time, exploiting the available, and even noisy, data can significantly improve the quality of the PINN prediction. This approach may be effective in a practical situation, where the general behavior of a physical process is known, but many different immeasurable phenomena may affect the outcome. The integration of some data measured in the site can help to account for marginal effects due to several minor dynamics. In that event, the solution is not forced to strictly satisfy exclusively the main process, but to balance it with other natural occurrences. We denote the applications investigated in this section as *sensor-driven* simulations, in the sense that we assume that data within the domain are available only where a physical sensor is installed.

The procedure is here applied to artificially-created synthetic examples, but the objective is to assess the PINN performance in view of its application to a real case, where monitoring data describing the evolution of the physical process are available at some specified locations. Both the one-dimensional and the two-dimensional configurations of Section 3.2 are tested and compared with a strictly forward solution.

3.3.1 1D: Terzaghi's problem

Following the outcome of Table 3.4, we build the PINN model by using a pressure network with 10 layers, 40 neurons and elu activation function, while the displacement network has 12 layers, 20 neurons and elu activation function. This PINN model is used to solve the classical 1D Terzaghi's consolidation problem in the setting of Figure 3.1 with the initial and boundary conditions of Equations (3.15)-(3.16). Since we assume to use PINN as a forward PDE solver with no additional information available, the number of collocation points has to be significantly increased with respect to the analysis carried out in Section 3.2. In particular, to achieve a sufficient accuracy we needed $N_c = 20000$ collocation points inside the domain where the PDE residual is evaluated, $N_{IC} = 6000$ initial points and $N_{BC} = 9000$ points uniformly distributed in time along the z-boundary. The training is performed with Adam's algorithm for 30000 epochs, with a batch size equal to 1000 and a learning rate constantly equal to 0.001 for the first 2000 epochs and then with an exponential decay to 0.0001 until the 5000th epoch and to 0.00001 until the end. The loss function is the same as in (3.6), but the weights vary following a *Neural Tangent Kernel* (NTK) guided gradient descent. This is chosen because it has been verified that this method generally improves convergence [Wang et al., 2021b]. All the other parameters have been chosen the same as in Section 3.2.1, with no early stopping. The accuracy of the outcome is satisfactory (Figure 3.9) with relative L^2-errors for pressure and displacement at about 2% and 0.7%, respectively.

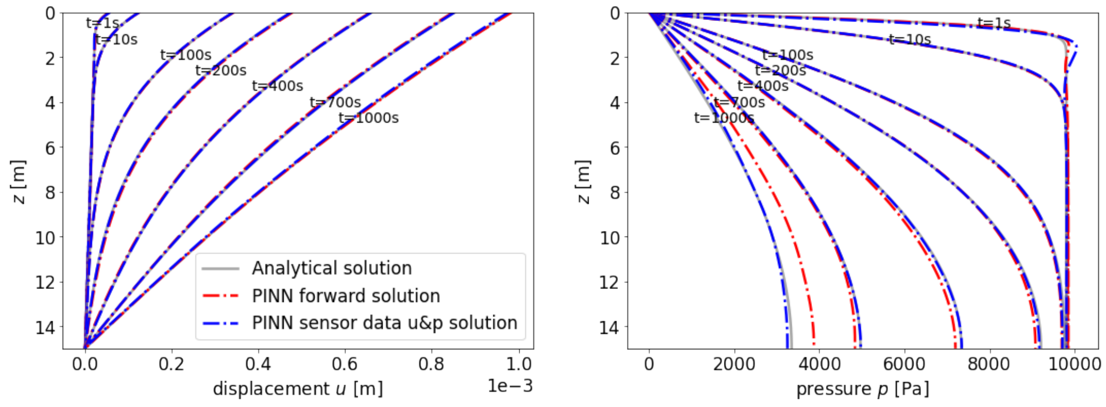
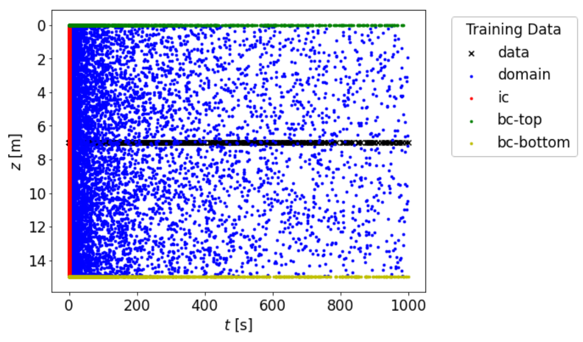


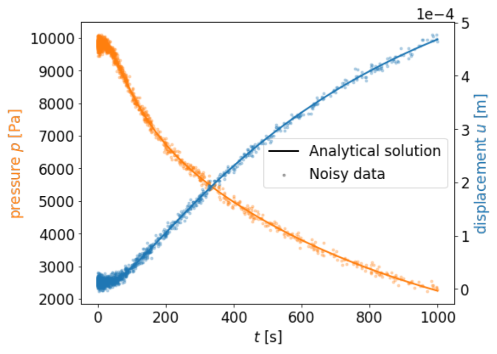
FIGURE 3.9: Terzaghi's problem: comparison between analytical and PINN solution using the model as a forward PDE solver.

Now, we assume that a sensor is located at $z = 7$m collecting pressure and displacement values in time, as in Figure 3.1. Exploiting the data coming from one single point in the physical domain, the number of collocation, initial, and boundary points used for the PDE residuals and the auxiliary conditions can be reduced to $N_c = 10000$, $N_{IC} = 3000$, and $N_{BC} = 4000$, with no other data points inside the domain, except for $N_d = 3000$ points at $z = 7$m (Figure 3.10a). Pressure and displacement data at the sensor location are synthetically obtained from the analytical solution (3.17). In order to better reproduce a realistic setting, noise has been also added to the exact values by introducing an error with a Gaussian distribution, zero mean and a standard deviation equal to 5% of exact values (Figure 3.10b), similarly to what is done in [Raissi et al., 2019].

The training is carried out with the same parameters as before, but now it can be stopped after 10000 epochs. The trends of the loss functions provided in Figure 3.11 clearly show the improvement allowed by the introduction of one "sensor" point only in space with respect to a simple forward solution in terms of both speed of convergence and accuracy. The mean of the L^2-norms of the errors on the validation set of three runs is about 0.5% for the predicted pressure and 0.2% for the vertical displacement (Table 3.5). Notice that also in case of noisy data we have an improvement both in the accuracy and in the convergence speed, hence in the overall computational load.

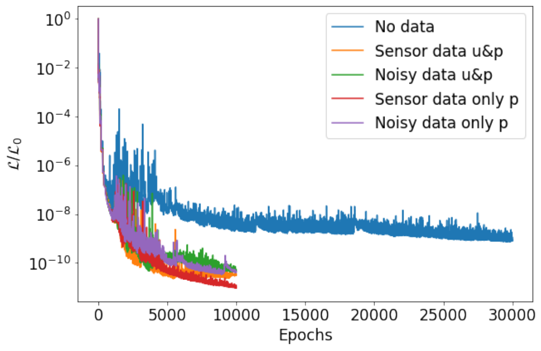


(A) Training points

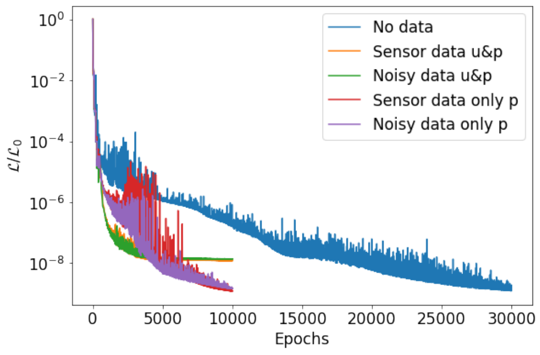


(B) Noisy data

FIGURE 3.10: Terzaghi's problem: (a) training points in the *sensor-driven* simulation in Terzaghi's problem with data given only at $z = 7$m; (b) 5% noise is added to the training sensor data to better reproduce a realistic setting.



(A) *Terzaghi*



(B) *Mandel*

FIGURE 3.11: Comparison between the losses relative to the initial value $\mathcal{L}_0$ of the forward and *sensor-driven* frameworks.

We considered also the case where only partial measurements are available, i.e. only pressure points are used within the domain at $z = 7$m with no data for displacements. This is usually closer to a real-world hydro-poromechanical problem, since in many applications, for example in subsurface engineering, it is generally easier and less expensive to monitor the fluid pressure in time with respect to deep displacements. In these last cases the accuracy obviously decreases, but the approximation errors still remain acceptable from an engineering application point of view, as it can be noticed in Table 3.5, with the related loss functions still efficiently minimized (Figure 3.11a).

TABLE 3.5: Errors for the different model applications. The mean error computed from three runs and different random seeds are provided.

	Terzaghi 1 sensor at $z = 7$m		**Mandel** 2 sensors at $(x, z) = (0.3, 0.3)$m, $(0.7, 0.7)$m		
	Error p	**Error u**	**Error p**	**Error ux**	**Error uz**
No data	2.279×10^{-2}	7.205×10^{-3}	2.655×10^{-1}	1.921×10^{-1}	1.268×10^{-1}
Sensor data u&p	4.891×10^{-3}	2.731×10^{-3}	4.311×10^{-2}	3.521×10^{-2}	5.571×10^{-2}
Noisy data u&p	1.556×10^{-2}	3.106×10^{-3}	2.177×10^{-2}	3.518×10^{-2}	6.799×10^{-2}
Sensor data only p	6.654×10^{-3}	1.314×10^{-2}	2.595×10^{-2}	1.446×10^{-1}	1.265×10^{-1}
Noisy data only p	1.411×10^{-2}	1.051×10^{-2}	4.327×10^{-2}	1.437×10^{-1}	1.275×10^{-1}

3.3.2 2D: Mandel's problem

The same analysis as in Section 3.3.1 is applied to Mandel's problem. We consider the PINN architecture defined by the hyper-parameters in Table 3.4, using for fluid pore pressure 4 layers with 20 neurons and $\tanh$ activation, and for horizontal and vertical displacement 8 layers with 20 neurons and $\tanh$ activation. We refer to the problem setting of Figure 3.6 and carry out an initial test with no other data but the initial and boundary conditions (3.21)-(3.22). The networks are trained over $N_c = 40000$ collocation points for the PDE residuals, $N_{IC} = 8000$ initial points, and $N_{BC} = 12000$ points equally distributed along the four spatial boundaries. The loss function is built as stated before with the NTK weighting. The training is performed for 30000 epochs with Adam's algorithm, a 0.001 learning rate constant until the 2000th epoch, with an exponential decay to 0.0001 for 3000 epochs and then to 0.00001 until the end. The size of the batch has been chosen equal to 2000, while the other choices follow those of the previous Section 3.2.2 without early stopping.

Figure 3.12 shows a comparison between the PINN approximation and analytical solution. The outcome is only fairly satisfactory, with discrepancies that can be appreciated for both the pressure and the displacement solution. In this case, the relative L^2-error norm is around 26% for pressure, 14% for horizontal displacement, and 13% for vertical displacement.

If we add a few pieces of information, the training process can be substantially improved. We suppose that two sensors are located at $(x, z) = (0.3, 0.3)$m and $(x, z) = (0.7, 0.7)$m collecting pressure and displacement values in time, as in Figure 3.6. As before, the synthetic measurements in time taken from the analytical solution (3.23) at the sensor locations are also perturbed by some noise with the same characteristics

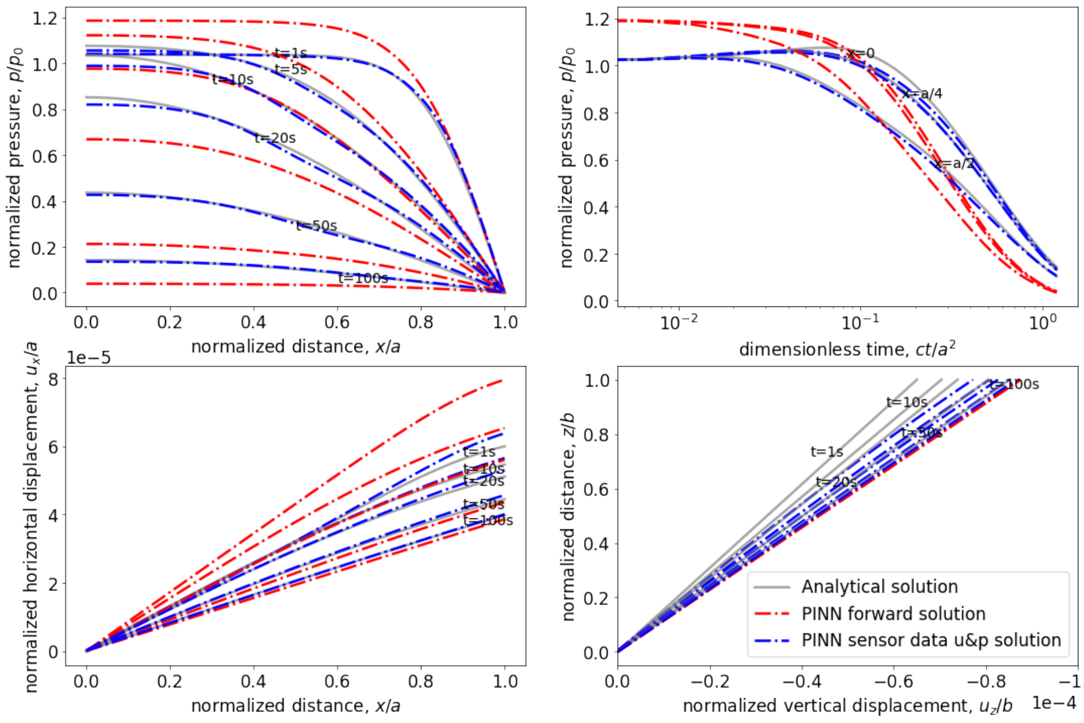


FIGURE 3.12: Mandel's problem: comparison between analytical and PINN solution using the model as a forward PDE solver.

as for Terzaghi's problem. The networks are now trained over $N_c = 20000$, $N_{IC} = 6000$, and $N_{BC} = 8000$ points, while $N_d = 6000$ training data points split into 3000 data points are collected at each location in time. As already observed in Terzaghi's application, the introduction of a sensor-driven condition significantly improves both the accuracy and the convergence speed. The loss minimization process requires only 10000 epochs with the behavior shown in Figure 3.11b, while the relative L^2-errors decrease to 4.3%, 3.5% and 5.6% for pressure, horizontal, and vertical displacement, respectively (Table 3.5).

As done before, we considered also the case where only pressure measurements are available. A summary of the errors is reported in Table 3.5. In Mandel's case, the errors are larger with respect to Terzaghi's case, but the advantage of the sensor-driven framework is still evident (Figure 3.11b and Figure 3.12). In particular, in this example, the precision reached by the forward solution is not fully satisfactory. Assuming that the information is available both for the pressure and displacements field, the results are again quite accurate. As expected, in the case where only pressure data are available, the accuracy for pressure remains quite good, but the errors for the approximations of the displacements are higher.

3.3.3 Discussion

The implementation of PINN models for problems governed by Biot's equations of hydro-poromechanics has been investigated. By this approach, prior knowledge (theoretical, empirical, mathematical, observational, etc.) of a problem is used to enhance the overall modeling procedure. However, the construction of effective neural networks for an accurate representation of the physical process depends on the selection of a number of hyper-parameters and is often quite expensive, ultimately remaining on the modeler's experience rather than on robust indications. The extensive experimentation carried out to analyze the structure of effective architectures for the accurate PINN training in poromechanics shows that some hyper-parameters are more influential than others in controlling the PINN model accuracy. Such a knowledge can limit the usually time-demanding empirical process required to build the approximating networks, driving the user into the design of NN architectures for the problem of interest. On the other side, PINN implementation on coupled problems suffers from well-known drawbacks as the high computational cost and the difficulties related to the multi-objective loss minimization. These limitations make the method non-competitive with most traditional and well-established PDE solvers based on discretization methods. However, PINNs have the advantage of allowing for an automatic data integration in the PDE solution stage. A sensor-driven framework has been introduced with the purpose of both accelerating the convergence of the minimization process and increasing the accuracy of the method. The numerical experiments on synthetic test cases show that integrating very few samples in space can substantially improve the PINN performance as a forward PDE solver. The results that follow are worth summarizing.

- The most significant hyper-parameter is the activation function of the neural networks approximating displacements. Care must be taken in its setting, restricting the choices to $\tanh$ and elu.
- The NNs for poromechanical applications do not seem to require a very complex structure to achieve approximation relative errors lower than 10^{-2}. Twelve layers

or less, along with no more than 40 neurons per layer, is enough.

- The magnitude of the approximation errors appears to be acceptable ($< 10\%$) for a wide range of hyper-parameter combinations, providing evidence of a good matching between analytical and PINN solutions.
- The analysis emphasizes the robustness of the PINN approach, also due to the presence of the PDE residuals in the loss function acting as regularization terms. This means that there is a high chance to reach good accuracy once identified only the most significant hyper-parameters.
- The proposed sensor-driven architecture has resulted in improvements with respect to a pure forward solution in terms of both computational training time and model accuracy.
- The application of the analysis to more practical cases, such as assuming the availability of a limited number of observations, has provided promising results, since it has led to accurate approximations from data given only on very few (1 or 2) fixed locations. This appears to be a good starting point for the assimilation of data in real-world problems' modeling.

Chapter 4

PINN for inverse solution in hydro-poromechanics

In the context of hydro-poromechanics, the need to estimate the main material parameters governing subsurface processes is definitely compelling. Since most of the measurement techniques are indirect and representative of the medium only at a very local level, the need for robust and efficient inverse solutions becomes of paramount importance.

When the required parameters have a measurable physical meaning, a major investigation branch concerns either in-situ observations used to study reservoir deformation with specialized measurement equipment, such as surface deformation monitoring by satellite techniques or well-logs equipped with radioactive markers [Ferronato et al., 2013], or laboratory tests, properly developed for characterization of geological formations [Mantica et al., 2016].

Otherwise, inverse modeling and data assimilation methods must be employed in order to exploit observations for approximating those parameters that are not directly linked to measurable quantities, [Zoccarato et al., 2016; Guzman et al., 2014; Wilschut et al., 2011; Jahandideh et al., 2021; Fokker et al., 2015]. Disposing of fast and accurate estimates of quantities of interest is extremely useful - if not even necessary - in a real-time or multi-query context, since the former requires the computation of the model in a very short amount of time, while the latter requires the model to be solved a huge number of times corresponding to various parameter instances sampled from parameter space.

Inverse problems, pervasive in hydrogeology, geomechanics, and reservoir engineering, require the unraveling of subsurface properties from observed data. Traditional approaches often grapple with computational challenges, limited data availability, and the inherent nonlinearities of the subsurface processes [Zoccarato et al., 2019; Bottazzi and Rossa, 2017]. In this context, *Physics-Informed Neural Network* (PINN) can be applied to decipher the complexities of hydro-poromechanical systems, by exploiting its cutting-edge property to fuse physics-based modeling and machine learning together. PINN can not only bridge the gap between data-driven and physics-based methodologies but also promise versatility in solving inverse problems. Indeed, if the governing equations of two different problems have the same structure, the residual loss term of PINNs does not change, resulting in a straightforward implementation of PINN to several applications, which only needs a slight change perhaps in the boundary and initial conditions.

We showcase PINN application in crafting alternative inverse solutions in the hydro-poromechanical field. Through insightful case studies, this chapter aims to provide an overview of an easy-to-implement tool, which highlights the current strengths and limitations of PINN application for parameter estimation of subsurface processes. The use of PINN in the context of addressing inverse problems in coupled hydro-poromechanics still needs investigations and in-depth analyses. Therefore, the results obtained in this work can be regarded as a preliminary outcome that has to be further considered in the next future.

The setting is the same as in Chapter 3, but this time the aim is to estimate not only the state variables, but also the geomechanical and hydraulic parameters involved in the Biot's model, represented by the Lamé constants and the hydraulic conductivity.

4.1 1D: homogeneous problem

In this section, we delve into the results obtained from the parameter identification of the Biot's model (3.1)-(3.2), focusing on a one-dimensional homogeneous setting, i.e. the Terzaghi's problem presented in Section 3.2.1. The system under consideration consists of a fluid-saturated sandy column subjected to a constant load, thus reproducing Terzaghi's classical scenario. The corresponding Lamé constants, λ_s, μ_s, and the hydraulic conductivity, k_s, (see Table 3.3) are additional unknowns of the problem, that can be estimated with the aid of some available data.

To quantify the impact of the number of training data points, we conduct a comparative analysis of parameter prediction accuracy between two cases:

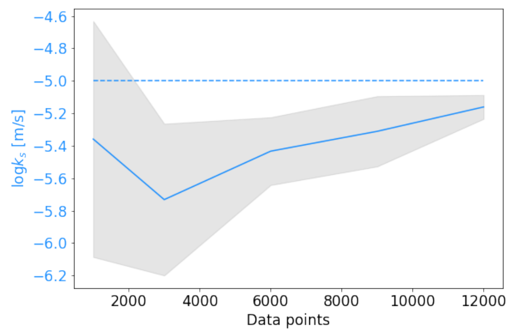
- an ideal framework in which we suppose to have data spread over the entire domain,
- a more realistic framework in which data are located at one fixed location, simulating the presence of a sensor as in the sensor-driven condition proposed in Section 3.3.

The model is built as stated in Section 3.2.1, using the loss function (3.6) where the weights vary following a NTK. The difference is that the hydraulic conductivity $k_s = 10^{-5}$ m/s and parameter $\lambda_s + 2\mu_s$ - which is the inverse of the vertical uniaxial compressibility c_M - with $\lambda_s = 40$ MPa, $\mu_s = 40$ MPa the Lamé constants, are unknown constant parameters. For their estimation, we use an object of the SciANN parameter class with the additional non-negative constraint, [Haghighat and Juanes, 2021]. The object consists of a scalar value that is updated during the training in the same way as the NN weights. In accordance with the results in Table 3.4, we built the model using a displacement NN that has 12 layers, 20 neurons, and elu activation function, and a pressure NN that has 10 layers, 40 neurons, and the same activation. With the initial and boundary conditions from Equations (3.15)-(3.16), this model is applied to the classical 1D Terzaghi's consolidation problem in Figure 3.1.

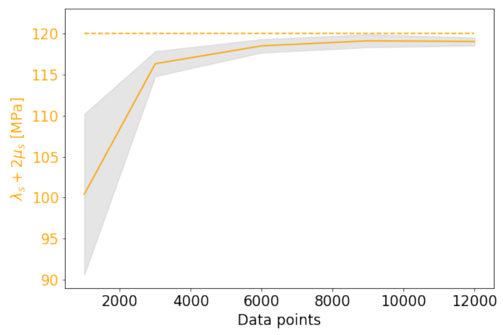
The training is performed with Adam's algorithm for 15000 epochs, with a batch size equal to 1000 and a learning rate constantly equal to 0.001 for the first 2000 epochs and then with an exponential decay to 0.0001 until the 5000th epoch and to 0.00001 until the end.

4.1.1 Spread training data points

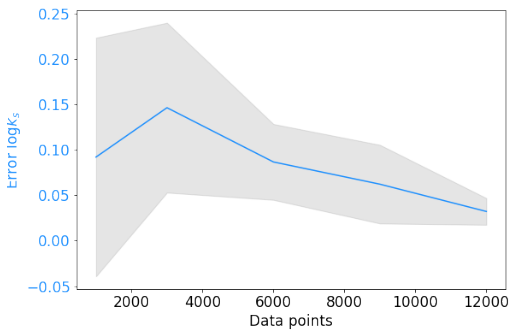
In the first scenario, we explore an ideal framework where data points of pressure p and displacements u are assumed to be randomly distributed across the entire domain (Figure 3.3). This scenario represents an optimal condition where comprehensive knowledge of the system is available.



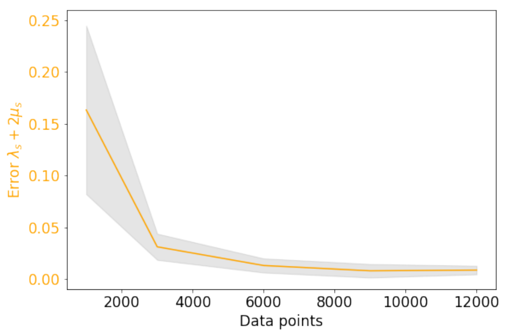
(A)



(B)



(C)



(D)

FIGURE 4.1: Accuracy of parameter predictions in an ideal framework with training data points spread across the domain (top) and corresponding mean error (4.1) trend (bottom) with the change in number of training data points. Grey bands provide the confidence interval for one standard deviation.

Figure 4.1 illustrates the accuracy of parameter predictions under this idealized condition. Hydraulic conductivity can take extremely low values and what is of interest in its estimation is the order of magnitude; hence, we report the base-10 logarithm of the predicted values. The solid lines correspond to the mean outcome from 10 different runs, while grey bands provide the confidence interval for one standard deviation. The parameter errors are relative absolute errors between predictions, $\hat{\lambda}$, and real values, λ:

$$\text{Error}(\hat{\lambda}, \lambda) = \frac{|\hat{\lambda} - \lambda|}{|\lambda|}. \tag{4.1}$$

As depicted in Figure 4.1, the PINN model exhibits good accuracy in capturing the Lamé constants and hydraulic conductivity across the entire domain. The even distribution of data points ensures a robust training regimen, with a clear correlation between estimate improvement and number of training data points.

4.1.2 Sensor training data points

In a more realistic framework, we simulate a sensor-driven condition where data points are available only at a single fixed location, akin to the presence of a sensor (Figure 3.10a). This scenario reflects practical limitations in data acquisition, as sensors might be strategically placed in a very few locations only due to resource constraints or specific measurement objectives.

Figure 4.2 presents the results obtained under this sensor-driven condition. The plots illustrate that, even with a limited number of data points in a localized region, the PINN model demonstrates accuracy in predicting the geomechanical and hydraulic parameters. The network effectively learns the system characterization, showcasing its ability to extrapolate information beyond the sparse data points. The trend of the accuracy is no more consistent with the amount of training data points, as if at a certain point the data information becomes redundant. Here, we are increasing the number of measurements in time, while keeping constant the space location of the sensor. From the practical consequence perspective, the results suggest that the amount of measurement in time is not the crux of the problem. Actually, at some points it appears that the boost of training data points located at fixed points in the domain adversely affects the estimations of the unknown parameters. This is probably due to the unbalance between the terms in the loss function, with the information of sensor data - narrowed to a small portion of the domain - hiding the physical knowledge describing the entire domain.

A comparison of Figures 4.1 and 4.2 indicates that while the mean and standard deviation errors are marginally higher in the sensor-driven condition, the PINN model remains robust in both scenarios, with a similar behavior under different training conditions. This underscores the versatility of PINN in handling diverse data configurations.

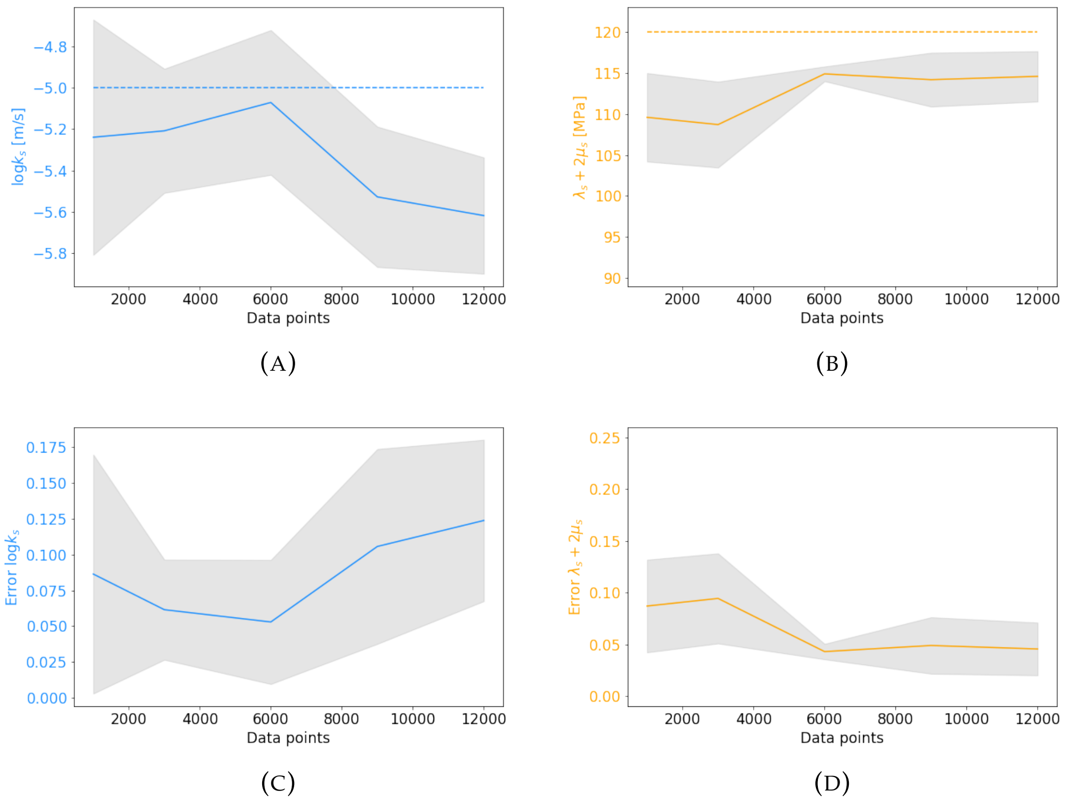


FIGURE 4.2: Accuracy of parameter predictions in a sensor-driven condition (top) and corresponding mean error (4.1) trend (bottom) with the change in number of training data points. Grey bands provide the confidence interval for one standard deviation.

The results presented here show the PINN effectiveness in identifying parameters of Biot's model under varying data conditions in a one-dimensional homogeneous setting. The model adaptability to sparse, sensor-driven data advises its applicability in real-world scenarios, where data availability may be limited.

In the subsequent sections, we delve deeper into the nuanced aspects of the parameter identification and discuss the broader implications of these findings in the context of hydro-poromechanics.

4.2 2D: homogeneous problems

We present the outcomes of the parameter identification for Biot's model (3.19)-(3.20), governing the coupled mechanics and fluid flow processes in a two-dimensional setting. The simulations involve a 2D spatial domain subjected to consolidation. The parameters of interest include the Lamé constants, representing geomechanical properties, and the hydraulic conductivity, characterizing fluid flow within the porous medium. The focus is on two distinct test cases:

- the simulation of 2D consolidation within a fluid-saturated porous medium with homogeneous properties,
- an extended scenario in which we introduce a pumping well within the domain undergoing 2D consolidation.

4.2.1 Case 1: Homogeneous consolidation

In our first test case, we investigate the parameter identification of Biot's model in the context of two-dimensional consolidation within a fluid-saturated, homogeneous porous medium. The medium is made of a homogeneous material, e.g. sand, and undergoes consolidation due to external loading, P_H. Our focus is on estimating the parameters characterizing the geomechanical and hydraulic properties of the sandy medium, λ_s, μ_s, and k_s. Figure 4.3 provides a visual representation of the model setup.

The equations describing the problem are (3.19)-(3.20) in $\Omega = [0, L]\times[0, H]\times[0, +\infty)$, with $\lambda = \lambda_s$, $\mu = \mu_s$, and $k = k_s$. The initial conditions read:

$$p_0(x, y) = \begin{cases} 0 & z = H \\ P_H & \text{otherwise} \end{cases}, \qquad u_{x,0}(x, z) = 0, \qquad u_{z,0}(x, z) = 0, \tag{4.2}$$

and the boundary conditions:

$$\begin{aligned}
&\frac{\partial p}{\partial x}(0, z, t) = 0, && u_x(0, z, t) = 0, && \frac{\partial u_z}{\partial x}(0, z, t) = 0, && x = 0,\\
&\frac{\partial p}{\partial x}(L, z, t) = 0, && u_x(L, z, t) = 0, && \frac{\partial u_z}{\partial x}(L, z, t) = 0, && x = L,\\
&\frac{\partial p}{\partial z}(x, 0, t) = 0, && \frac{\partial u_x}{\partial z}(x, 0, t) = 0, && u_z(x, 0, t) = 0, && z = 0,\\
&p(x, H, t) = 0, && \frac{\partial u_x}{\partial z}(x, H, t) = 0, && (\lambda_s + 2\mu_s)\frac{\partial u_z}{\partial z}(x, H, t) = -P_H, && z = H.
\end{aligned} \tag{4.3}$$

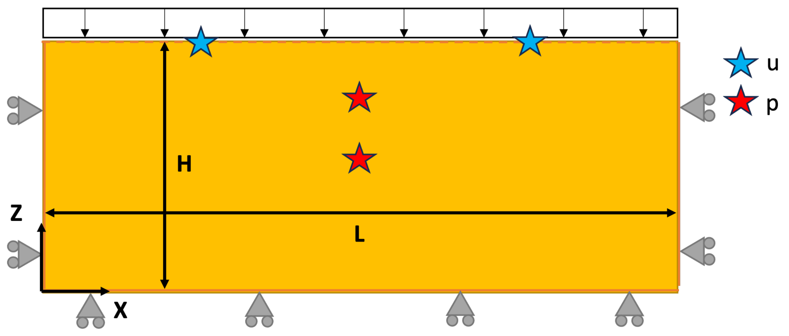


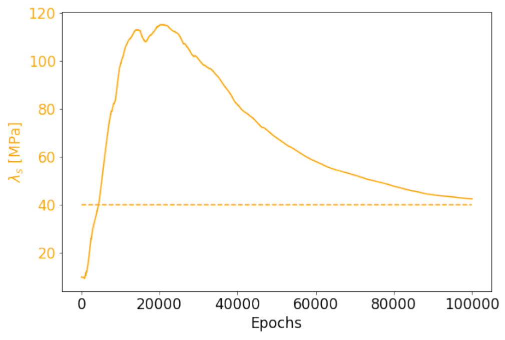
FIGURE 4.3: Case 1: Sketch of the setup for 2D consolidation in a homogeneous setting with indication of sensor locations.

We set the external load $P_H = 90$ kPa, and the domain dimensions $L = 100$ m and $H = 30$ m. All the other parameters are chosen as in Table 3.3. The numerical final time is $T = 218160$ s.

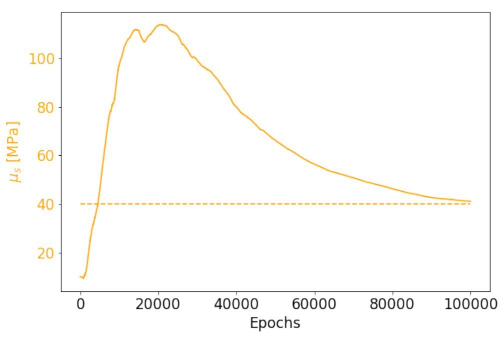
The PINN model consists of three NNs - $\hat{p}_s$, $\hat{u}_{x,s}$, $\hat{u}_{z,s}$ - trained by minimizing the misfit with pressure and displacement scaled data obtained by solving the problem with a FE method over training points $\{x_d^i, z_d^i, t_d^i\}_{i=1}^{N_d}$ corresponding to ideal sensors located as in Figure 4.3. In particular, we suppose to have the datum of surface displacements in time at $x = 25$ m and $x = 75$ m, and pressure measurements in $(x, z) = (50, 21)$ m and $(x, z) = (50, 15)$ m. In order to reduce the number of loss-terms, the Dirichlet boundary conditions in (4.3) have been imposed in a strong form by adjusting the NN architecture, as proposed in [Sukumar and Srivastava, 2022; Berrone et al., 2023]:

$$\begin{aligned} \hat{p}_s &= \hat{p}_s p_{BC} - \frac{p_{min}}{p_{sc}}, \\ \hat{u}_{x,s} &= \hat{u}_{x,s} u_{x,BC} - \frac{u_{x,min}}{u_{x,sc}}, \\ \hat{u}_{z,s} &= \hat{u}_{z,s} u_{z,BC} - \frac{u_{z,min}}{u_{z,sc}}, \end{aligned} \tag{4.4}$$

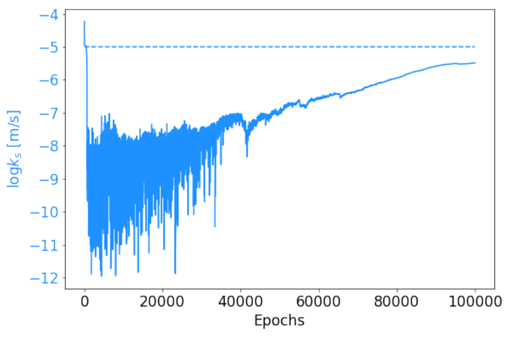
with $p_{BC} = H - z$, $u_{x,BC} = (L - x)x$, and $u_{z,BC} = z$. The corresponding unscaled NNs (3.18) serve to compute both the residual of the governing equations (3.19)-(3.20) and the auxiliary conditions in (4.2)-(4.3). Hence, we minimize the model loss function built as in (3.6) with $N_C = 10000$ collocation points, $N_{IC} = 3000$ initial points, and $N_{BC} = 4000$ boundary points. We made use of 25 data points for each sensor, for a total of $N_d = 100$ training data points. The training lasts 100000 epochs.



(A)



(B)



(C)

FIGURE 4.4: Case 1: Estimation of the unknown parameters λ_s (A), μ_s (B), and k_s (C) during the training for 2D consolidation in a homogeneous porous medium.

As depicted in Figure 4.4, the parameter identification process yields accurate predictions for the geomechanical and hydraulic parameters governing the process. Table 4.1 contains the errors (3.12)-(4.1) in the PINN approximations of the problem unknowns. The model is able to solve the governing equations while simultaneously estimating the parameters involved in with errors up to 10%. The identified parameters enable a comprehensive understanding of the consolidation process in the homogeneous porous medium (Figure 4.5). This validates the effectiveness of the methodology in capturing the behavior of the system.

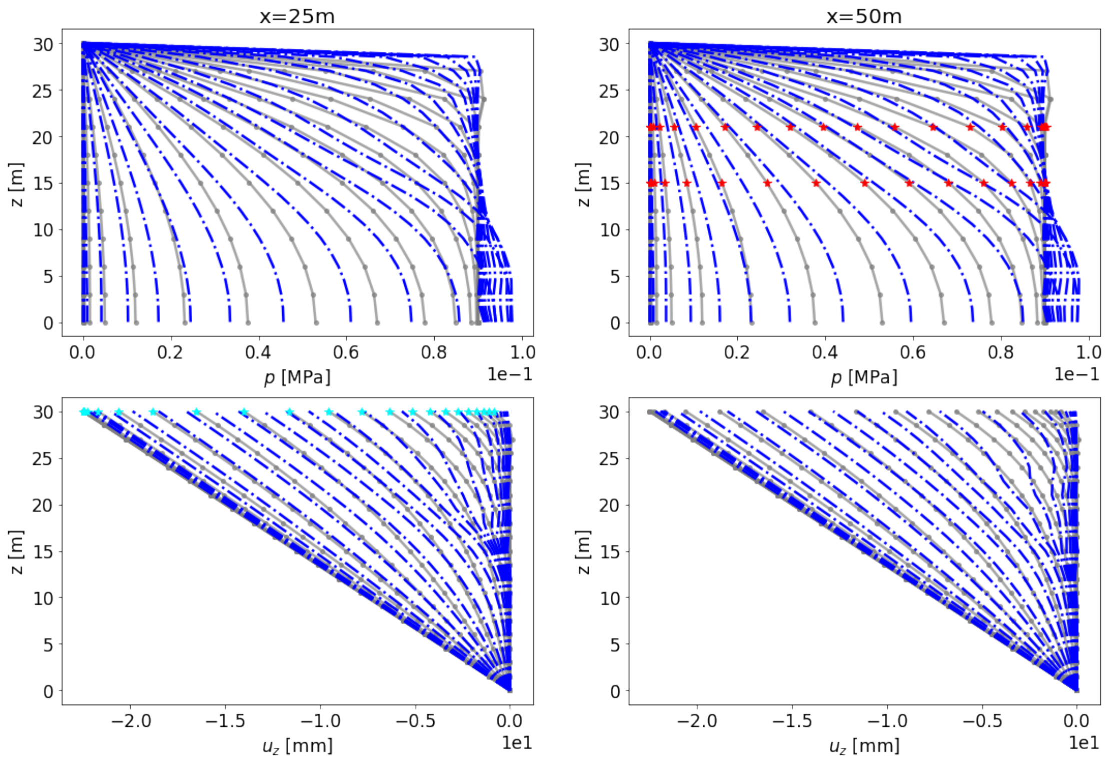


FIGURE 4.5: Case 1: Comparison between analytical and PINN solution in a two-dimensional homogeneous consolidation setting.

4.2.2 Case 2: Homogeneous consolidation with a pumping well

In the second test case we extend the previous scenario by introducing a water pumping well within the domain undergoing the consolidation, adding complexity to the problem dynamics (Figure 4.6). The simulation involves the same 2D spatial domain of the previous case, but the presence of the well modifies the flow and deformation

TABLE 4.1: Errors (3.12)-(4.1) for the different model applications in a two-dimensional homogeneous framework.

	Case 1	**Case 2**
Error p	8.621×10^{-2}	1.301×10^{-1}
Error ux	1.251×10^{-2}	1.601×10^{-3}
Error uz	1.291×10^{-2}	5.985×10^{-2}
λ_s **real** [MPa]	40	40
λ_s **pred** [MPa]	42.53	40.54
Error λ_s	6.330×10^{-2}	1.358×10^{-2}
μ_s **real** [MPa]	40	40
μ_s **pred** [MPa]	41.05	39.81
Error μ_s	2.631×10^{-2}	4.824×10^{-3}
$\log k_s$ **real** [m/s]	-5	-
$\log k_s$ **pred** [m/s]	-5.49	-
Error $\log k_s$	9.775×10^{-2}	-

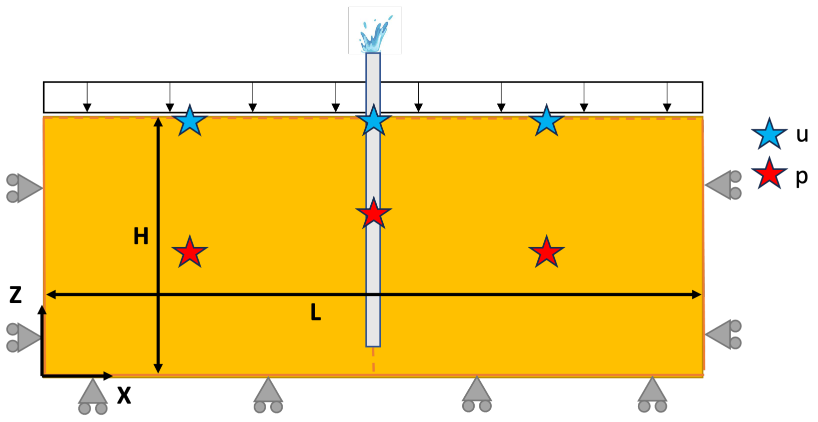


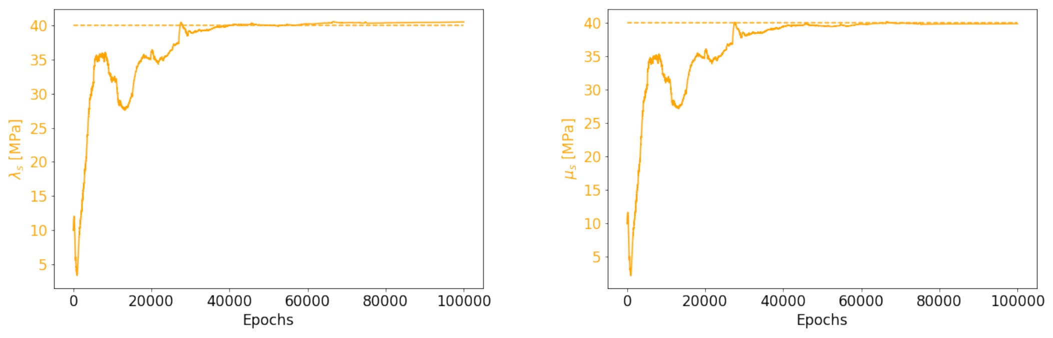
FIGURE 4.6: Case 2: Sketch of the setup for 2D consolidation in a homogeneous setting with a pumping well. The stars indicate sensor locations.

patterns. Our objective is to identify the material parameters considering the influence of the pumping well on fluid flow and consolidation.

The well is located at $x = 50$ m, starting from $z = 3$ m to the top surface. The presence of the well entails an additional constraint for pressure:

$$p(50, z, t) = 0, \qquad x = 50,\ 0 \leq z \leq 3 \tag{4.5}$$

that is imposed as Dirichlet boundary condition in a strong way, by constructing a proper distance function p_{BC} in (4.4). The same equations (3.19)-(3.20) and boundary conditions (4.3) of Case 1 hold, so the loss function is the same. The hyper-parameters defining the training and the model structure have been chosen as in the previous case, the only difference being the number and location of the sensors, which now are 6 in



(A) (B)

FIGURE 4.7: Case 2: Estimation of the unknown parameters λ_s (A) and μ_s (B) during the training for 2D consolidation in a homogeneous porous medium with a pumping well.

total. We suppose to have 3 sensors measuring surface displacements at $x = 25, 50, 75$ m and 3 piezometers in the same x-locations at different heights ($z = 15, 18$ m), as in Figure 4.6. In this setting, the presence of the well perturbs the flow dynamics and challenges the estimation of the hydraulic conductivity. We have tested different scenarios of sensor collocation and k_s was always predicted null and interfering with the Lamé constant predictions. Postponing the problem discussion to a future work, we focus on the estimation of Lamé constants λ_s and μ_s only.

Figure 4.7 illustrates the results obtained for this scenario, showcasing the model ability to discern the geomechanical parameters in the presence of a pumping well too. The state variables exhibit variations near the well, reflecting the localized influence of pumping on consolidation dynamics (Figure 4.8); nevertheless the model is capturing the general behavior fairly well.

To quantitatively assess the accuracy of parameter identification, we report the relative absolute error (4.1) for each parameter in Table 4.1, along with the L^2-errors (3.12) for p, u_x, u_z. The results highlight the model ability to accurately identify parameters in both test cases, with slightly increased error values in Case 2, due to the additional complexity introduced by the pumping well. PINN proved to be a robust tool in handling the added complexity introduced by the well. The identified parameters showcase the model adaptability to dynamic changes in the porous medium. This capability to adapt to varying conditions highlights the versatility of the parameter identification methodology.

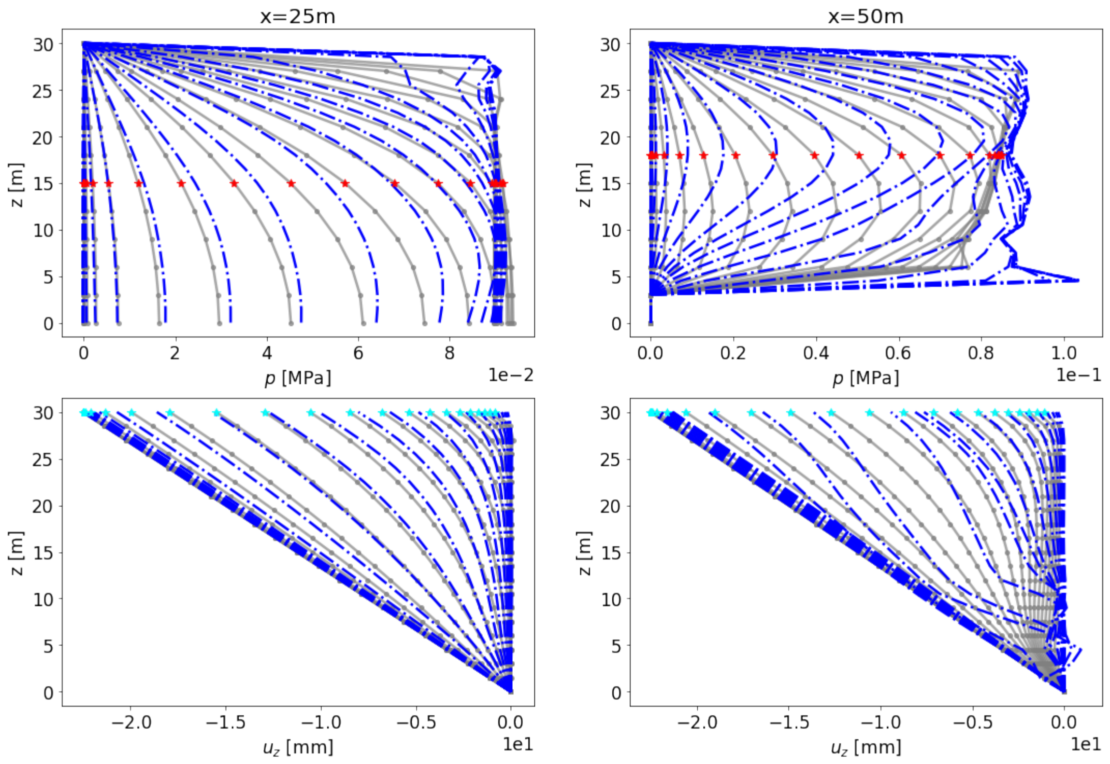


FIGURE 4.8: Case 2: Comparison between analytical and PINN solution in a two-dimensional homogeneous consolidation setting with a pumping well.

4.3 2D: heterogeneous problems

In this section, we widen the application of PINN to a more complex 2D scenario, targeting the identification of hydraulic and geomechanical parameters in Biot's system of PDEs. The study domain is characterized a heterogeneous material involving two distinct layers: one composed of sand and the other of clay. This stratified subsurface setting provides a representation of the actual geological conditions often encountered in real sedimentary basins.

4.3.1 Case 3: Heterogeneous consolidation

The first test case considers a 2D domain discretized into two layers, with different material properties, subjected to a constant load P_H. The lower layer has height H_1 and consists of sand, while the H_2-high upper layer is composed of clay. See Figure 4.9 for a sketch of the application into consideration. The challenge is to accurately identify

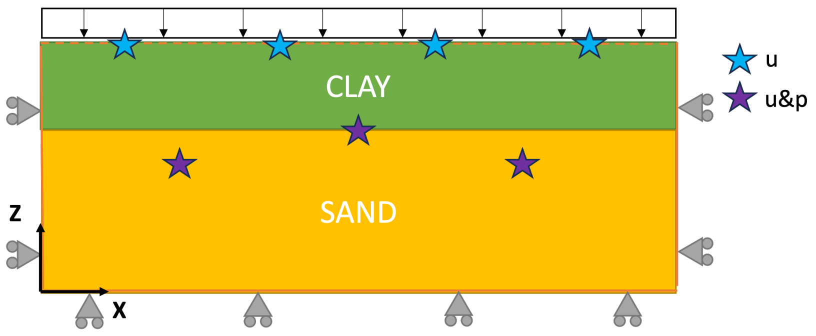


FIGURE 4.9: Case 3: Sketch of the setup for 2D consolidation in a heterogeneous setting with indication of sensor locations.

the hydraulic conductivities (k_s and k_c) and the Lamé constants (λ_s, μ_s, λ_c, and μ_c) for each layer.

The heterogeneity is modeled as follows. We marked the collocation points as belonging to the sandy or clayey layer, and the loss function contains double residual terms for the equilibrium equation (3.19) and the continuity equation (3.20): one for the lower layer, computed over the sand-collocation points, while the other is computed over the remaining clay-collocation points, corresponding to the upper layer. This is equivalent to assume to know the spatial distribution of the different layers, but not the nature of the materials which are made of, which is a quite usual condition met in reality. The boundary conditions (4.3) still hold, but the boundary condition for u_z on the top surface reads:

$$(\lambda_c + 2\mu_c)\frac{\partial u_z}{\partial z}(x, H, t) = -P_H, \qquad z = H. \tag{4.6}$$

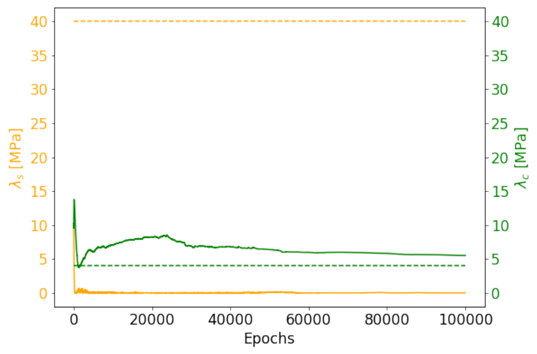
since the upper layer is now made of clay. The domain dimensions are as in the previous cases, with $H_1 = 18$ m and $H_2 = 12$ m the heights of the lower and upper layers, respectively.

The sensor locations are highlighted in Figure 4.9. We consider four top surface sensors measuring displacements in $x = 20, 40, 60, 80$ m and three sensors in $(x, z) = (25, 15)$ m, $(50, 18)$ m, $(75, 15)$ m, measuring both pressure and displacements.

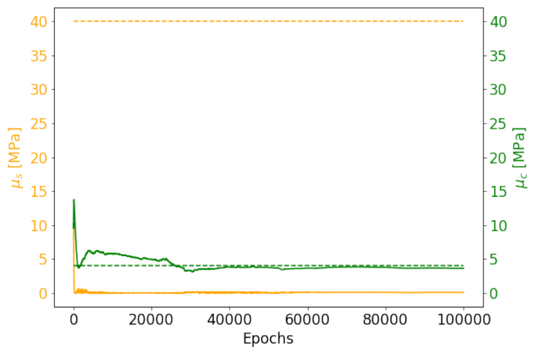
We analyze the results obtained from the inverse solution of Biot's model in a 2D heterogeneous setting, as presented in Figure 4.10. Table 4.2 displays the real values of the parameters alongside their corresponding predicted values obtained through PINN

TABLE 4.2: Errors (3.12)-(4.1) for the different model applications in a two-dimensional heterogeneous framework.

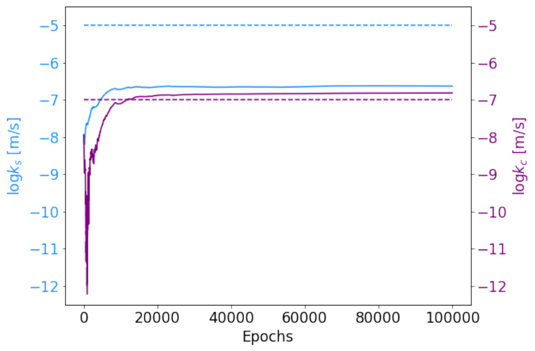
	Case 3	**Case 4**
Error p	8.621×10^{-2}	4.182×10^{-1}
Error ux	1.251×10^{-2}	6.900×10^{-4}
Error uz	1.291×10^{-2}	2.471×10^{-1}
λ_s **real** [MPa]	40	40
λ_s **pred** [MPa]	0	38.46
Error λ_s	1.000×10^{0}	3.847×10^{-2}
μ_s **real** [MPa]	40	40
μ_s **pred** [MPa]	6.14×10^{-3}	38.96
Error μ_s	9.985×10^{-1}	2.601×10^{-2}
$\log k_s$ **real** [m/s]	-5	-
$\log k_s$ **pred** [m/s]	-6.63	-
Error $\log k_s$	3.263×10^{-1}	-
λ_c **real** [MPa]	4	4
λ_c **pred** [MPa]	5.49	0.762
Error λ_c	3.744×10^{-1}	8.095×10^{-1}
μ_c **real** [MPa]	4	4
μ_c **pred** [MPa]	3.61	2.59
Error μ_c	9.686×10^{-2}	3.526×10^{-1}
$\log k_c$ **real** [m/s]	-7	-
$\log k_c$ **pred** [m/s]	-6.81	-
Error $\log k_c$	2.615×10^{-2}	-



(A)



(B)



(C)

FIGURE 4.10: Case 3: Estimation of the unknown parameters λ_s, λ_c (A), μ_s, μ_c (B), k_s, k_c (C) during the training for 2D consolidation in a heterogeneous porous medium.

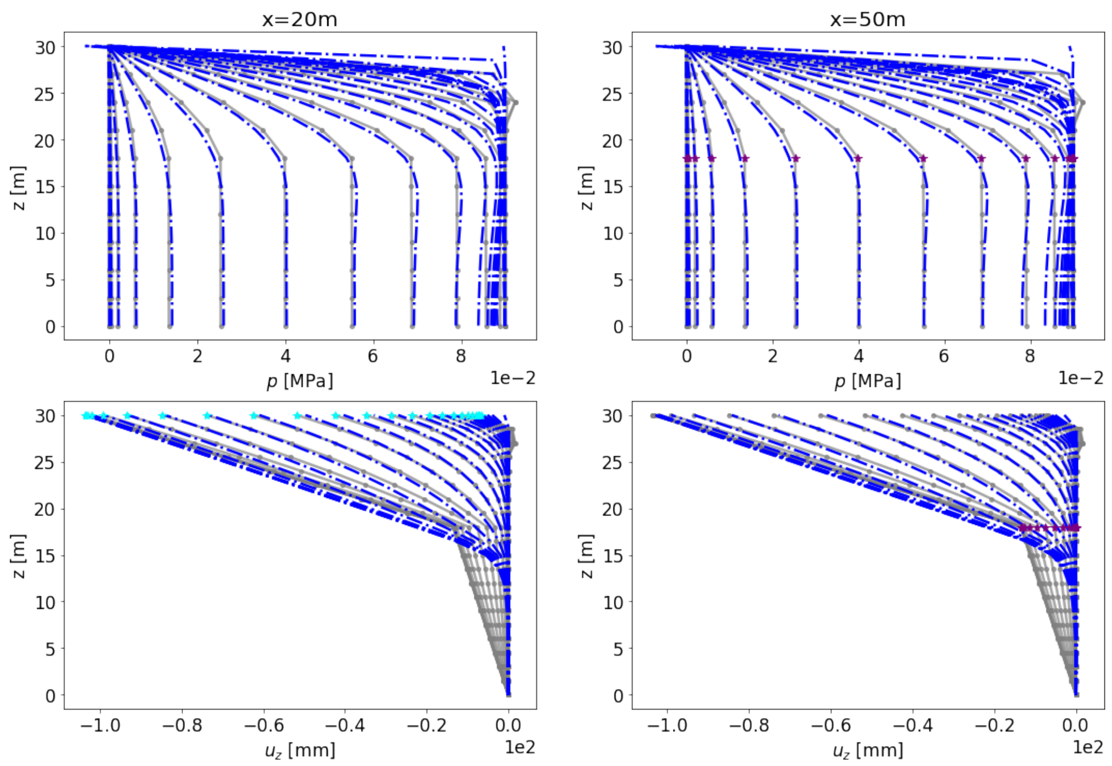


FIGURE 4.11: Case 3: Comparison between analytical and PINN solution in a two-dimensional heterogeneous consolidation setting.

and the approximation errors (3.12)-(4.1) for each of the unknowns. The Lamé constants λ_c and μ_c predictions are relatively close to their real values, suggesting that the model is able to capture the geomechanical properties of the clay layer with reasonable accuracy. The predicted hydraulic conductivity (k_c) for the clay layer is slightly higher than the real value; on the other hand, the predicted hydraulic conductivity (k_s) for the sand layer is significantly lower than the real value. This might imply a more restrictive flow within the sand layer in the model compared to the real-world conditions and that the clay layer is considered more permeable than it actually is. However, it is to be noted that the harmonic mean of the hydraulic conductivities is 1.98×10^{-7} m/s and the predicted k_c and k_s take values close to it, since they respectively equal 1.52×10^{-7} m/s and 2.33×10^{-7} m/s. This can be explained by the equivalence between the flow in an heterogeneous medium orthogonal to the spatial distribution of the materials - as in the present case - and the flow in a homogeneous domain with the hydraulic conductivity equal to the harmonic mean of the previous ones. Such result implies that the model is able to capture the physical process behind the data, but fails to catch the heterogeneity that characterize the problem. In practice, the fluid can be equivalently predicted by two heterogeneous layers or a homogeneous one with the harmonic mean of the hydraulic conductivities, and the PINN model converges to the latter solution. Another evidence of this is the prediction of the Lamé constants λ_s and μ_s, which is zero. This indicates challenges in capturing the mechanical behavior of the sand layer accurately, so the heterogeneity. The null predictions, that are not physically consistent, are a consequence of the ill-posedness of the inverse problem in the sandy layer. Indeed, in the lower part of the domain, the pressure p and the vertical displacements u_z are respectively almost constant and almost linear in space (Figure 4.11). Therefore, their first and second derivatives, respectively, are null. In particular, $\Delta \mathbf{u}$, $\nabla \text{div} \mathbf{u}$, and ∇p equals zero, so the equilibrium equation (3.1) for the sandy layer:

$$\mu_s \Delta \mathbf{u} + (\lambda_s + \mu_s) \nabla \text{div} \mathbf{u} = \alpha \nabla p, \tag{4.7}$$

is always satisfied for any value of λ_s and μ_s. The minimization performed during the training returns the lowest possible value which, with the non-negative constraint, is 0. Further investigation is needed to address this issue, for example by imposing a condition along the boundary between the two layers, which constrains the parameter values to be different from zero (as in the next test case in Section 4.3.2). Recently, a cusp-capturing PINN, able to present continuous solutions that inherently have discontinuous first derivatives on interfaces, has been proposed in Tseng et al. [2023], and could be also helpful to fix the problem.

4.3.2 Case 4: Heterogeneous consolidation with a pumping well

The fourth test case corresponds to a 2D heterogeneous framework with two layers (sand and clay) and a pumping well located in the middle of the domain. The setting is the same as Case 3, so all the hyper-parameters are set as in the previous sections. To model the presence of the well at $x = 50$ m, we impose in a strong way the boundary condition (4.5) by modifying the NN approximating p, as stated in Section 4.2.2.

The presence of the well makes the pressure no more constant in space in the lower

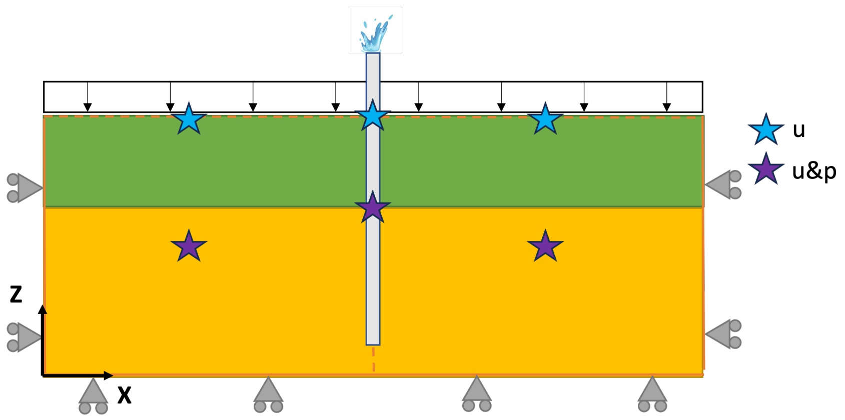


FIGURE 4.12: Case 4: Sketch of the setup for 2D consolidation in a heterogeneous setting with a pumping well. Sensor locations are marked with colored stars.

layer. However, in order to avoid the ill-posedness of the problem, we add a Neumann-type constraint on the vertical displacement along the boundary between the two layers:

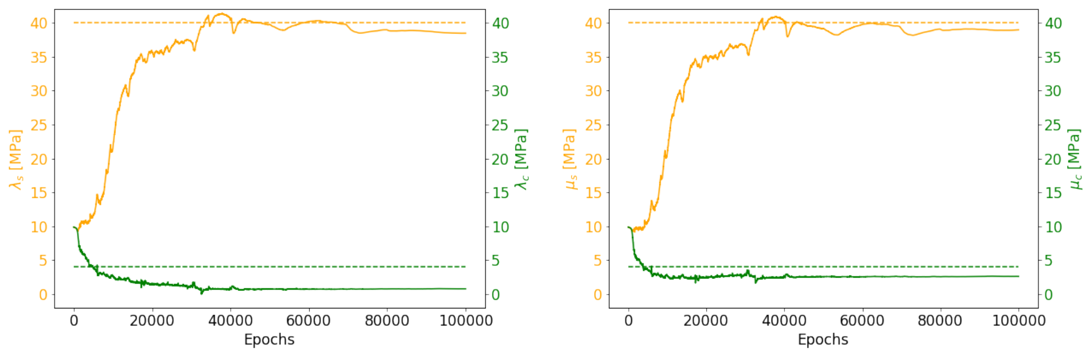
$$(\lambda_s + 2\mu_s)\frac{\partial u_z}{\partial z}^{-}(x, H_1, t) = p(x, H_1, t) - P_H, \qquad z = H_1. \tag{4.8}$$

where we use the superscript $\cdot^{-}$ to indicate the left partial derivative of u_z with respect to z. Therefore, the loss function contains an extra term to impose that the neural networks $\hat{p}$ and $\hat{u}_z$ and the estimated parameters satisfy condition (4.8) too.

The number of training data points is 150, since we consider 25 training data points per sensor. In this test case, displacement measurements are given at $(x, z) = (25, 30)$ m, $(50, 30)$ m, $(75, 30)$ m, while both pressure and displacement are supposed to be known at $(x, z) = (25, 15)$ m, $(50, 18)$ m, $(75, 15)$ m.

Table 4.2 provides various error metrics (3.12)-(4.1) and compares real values with the predicted ones for different parameters. The L^2-error for horizontal displacement (u_x) is relatively low, indicating a good agreement between the predicted and real values. However, the errors for pressure (p) and vertical displacement (u_z) are higher, suggesting potential challenges in capturing pressure and vertical deformation accurately, possibly influenced by the presence of the pumping well.

The predicted Lamé constants for the sand layer are reasonably close to their real values, with errors within acceptable ranges (2-4%). This indicates that the model is capable of capturing the geomechanical properties of the lower layer. For the clay layer,



(A) (B)

FIGURE 4.13: Case 4: Estimation of the unknown parameters λ_s, λ_c (A), and μ_s, μ_c (B) during the training for 2D consolidation in a heterogeneous porous medium.

the predicted λ_c and μ_c show discrepancies, with higher errors compared to the sand layer, suggesting that the presence of a pumping well challenges the geomechanical characterization of the upper layer.

Further refinement of the model, possibly by adjusting model architecture or training strategies, may be needed to improve accuracy in capturing the effects of the pumping well on both geomechanical parameters. Moreover, additional data points or considerations specific to the influence of pumping on material properties could be incorporated for enhanced model performance.

4.4 Discussion

In this chapter, we presented a comprehensive investigation into the application of PINNs for the hydro-poromechanical modeling of 2D consolidation in porous media. The inverse problem of parameter identification in Biot's model was approached using neural networks to estimate key geomechanical and hydraulic properties of subsurface materials. The study involved several test cases, each shedding light on the capabilities and challenges of the PINN approach. The outcomes of this 2D application hold significant implications for hydro-poromechanics, since a good characterization of subsurface materials is crucial for predicting fluid flow, deformation, and overall mechanical responses in hydro-poromechanical systems.

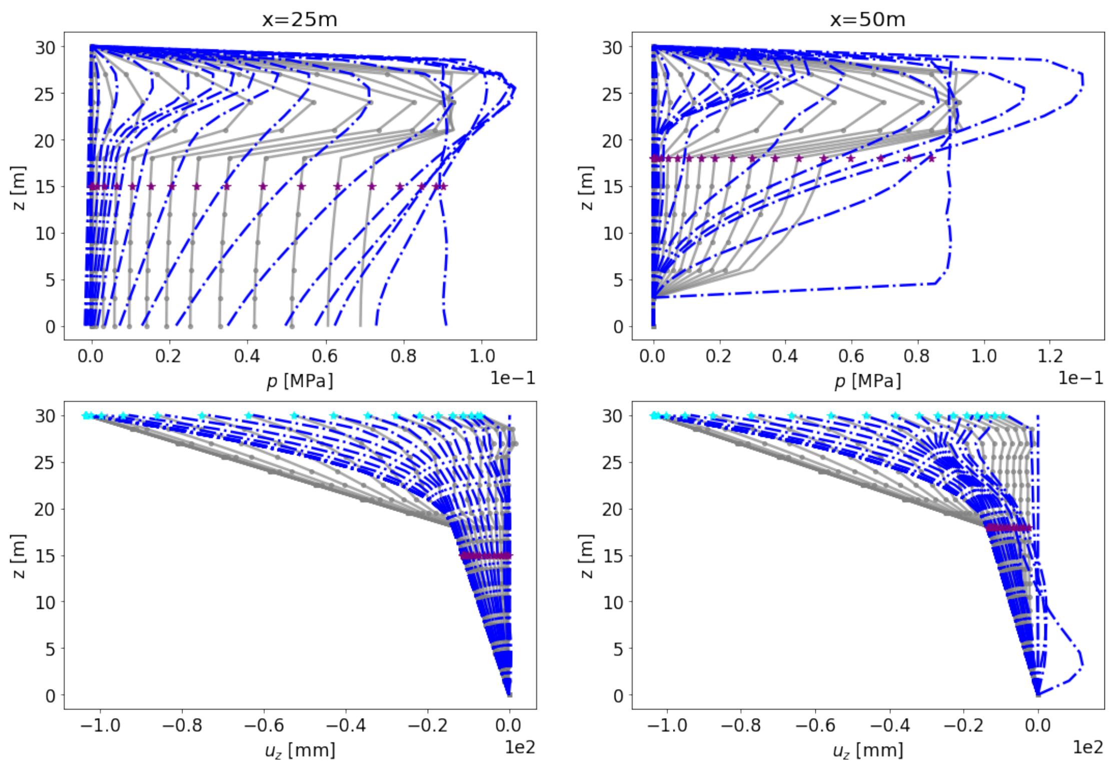


FIGURE 4.14: Case 4: Comparison between analytical and PINN solution in a two-dimensional heterogeneous consolidation setting.

The results of the PINN simulations demonstrated promising accuracy in estimating certain parameters. In the absence of a pumping well (Cases 1 and 3), the model exhibited proficiency in predicting the Lamé constants for the surface layer. In the homogenous setting, the hydraulic conductivity also showed reasonable agreement with real values, suggesting the model's ability to capture geomechanical and hydraulic properties with precision. However, in the heterogeneous application, the hydraulic conductivities were predicted equal to the harmonic mean of the reference values, since in that setting, the fluid can be equivalently predicted by two heterogeneous layers or a homogeneous one with the hydraulic mean of the conductivities. This indicates potential challenges in accurately characterizing heterogeneous conductivity and the need to properly constrain the model to the heterogeneous solution.

The ill-posed nature of the inverse problem became also evident in Case 3, where the PINN model struggled to accurately predict the geomechanical properties of the sand layer. The occurrence of zero predictions for the Lamé constants is due to the ill-posedness of the problem in regions where pressure and vertical displacements showed almost constant and linear behavior, respectively. Strategies such as incorporating boundary conditions or exploring cusp-capturing PINN models were suggested for addressing these challenges.

The introduction of a pumping well in Cases 2 and 4 added another level of complexity to the simulation. In Case 4, the Lamé constants were still accurately estimated, but the presence of the well complicated the fluid dynamics and made the hydraulic conductivities hardly identifiable. While the model of Case 4 exhibited reasonable accuracy in predicting the Lamé constants for the sand layer, discrepancies were observed in the clay layer. The pressure and vertical displacement errors increased, indicating potential challenges posed by the presence of a pumping well in accurately characterizing the subsurface.

The outcomes of this study provide valuable insights into the application of PINN for hydro-poromechanical simulations in homogeneous and heterogeneous subsurface environments. Future research directions could involve refining the model architecture, exploring advanced training strategies, and incorporating additional data points specific to the influence of pumping on material properties. Investigating cusp-capturing PINN models, as suggested in the literature, may offer a solution to the ill-posedness observed in certain regions.

In conclusion, while the PINN approach shows promise in estimating certain parameters in hydro-poromechanical simulations, ongoing research and refinement are essential to address challenges associated with ill-posedness and the influence of complex factors. These findings pave the way for advancements in the field, offering new avenues for improving the accuracy of subsurface parameter identification in real-world applications, providing a preliminary interpretation of the results and suggest avenues for further investigation of the PINN model for improved parameter identification in heterogeneous hydro-poromechanical settings and highlight potential areas for improvement and future research.

Chapter 5

PINN for time-dependent parameter estimation in epidemiology

Epidemiological models are nowadays fundamental to assist and guide policy makers in the fight against the spreading of diseases. This has been evident during the recent SARS-CoV-2 pandemic, when epidemiologists and scientists all over the world devoted their research to develop ad-hoc transmission models. Focusing, for example, on Italy, where the European outbreak started in February 2020, epidemiological models have been adopted to analyze different aspects of the epidemic: to determine the urgency to impose regional restrictions [Guzzetta et al., 2020]; to analyze the impact of the national lockdown [Marziano et al., 2021b; Gatto et al., 2020]; to explore the results of transmission scenarios after the release of the restrictions [Bertuzzo et al., 2020]; to study the impact of the different variants and the vaccination campaign [Marziano et al., 2021a; Gozzi et al., 2022; Parolini et al., 2022]; and to compute optimal strategies for the vaccine deployment in order to minimize the number of cases or deaths [Lemaitre et al., 2022; Ziarelli et al., 2023]. Most of these studies describe the SARS-CoV-2 transmission using different variations of compartmental models. The basic SIR model is at the core of those more-complex epidemiological models. It subdivides the population of interest into compartments indicating the infectious status of each individual (i.e. susceptible, infected, or recovered individuals). The dynamic describes the mean contacts between susceptible and infected individuals, and thus, the average rate at which susceptible individuals transit to the infected compartment. The main model parameter is the rate of transmission of the infection, β. This is strictly related to the well known basic reproduction number, $\mathcal{R}_0$, representing the average number of secondary infections generated by one infected individual in a totally susceptible population. The value of this quantity changes during an outbreak due to the temporal variations in human behavior (caused, for example, by changes in individual awareness or social distancing policies) and in the infectiousness of the virus. The effective reproduction number, $\mathcal{R}_t$, aims at describing the ongoing transmission in a changing system.

Data-driven methods provide effective estimates of $\mathcal{R}_t$ based on the renewal equation [Cori et al., 2013; Pasetto et al., 2021; Trevisin et al., 2023], i.e., a convolution on the reported cases having as kernel the serial interval (the time interval between the symptom onset of an individual and its secondary infections). These data-driven estimates do not explicitly provide a relationship between the changes in $\mathcal{R}_t$ and its possible causes, such as the implemented non pharmaceutical interventions or the vaccination campaigns. Compartmental models give a deeper understanding of the ongoing spreading of the disease and, at the same time, allow the computation of $\mathcal{R}_t$ using the spectral radius of the next generation matrix [Mari et al., 2021; van den Driessche and Watmough, 2002; Diekmann et al., 1990]. However, they require the assessment and calibration of time-dependent parameters.

Tracking the temporal variations in the model parameters is an essential but complex problem to follow and predict the spreading of a disease. Many studies tackle this problem using Bayesian inference, i.e., searching for the posterior distribution of the unknown parameters based on the available reported cases and the prior distribution. Among these approaches, we recall the iterative particle filter [Ionides et al., 2015], sequential data-assimilation schemes [Pasetto et al., 2017], or the use of subsequent *Markov Chain Monte Carlo* (MCMC) [Parolini et al., 2022; Bertuzzo et al., 2020]. Being based on random sampling, these approaches might result in low quality results and large computational times, due to the slow Monte Carlo convergence.

Here, we propose to adopt a deterministic approach based on *Physics-Informed Neural Network* (PINN). In practice, this is done by describing the state variables and, in case, the time-dependent parameters using NNs. The parameters of the NNs are trained by seeking the minimum of a loss function based on both the misfit on the available data, and the residual of the differential equations governing the problem at hand, i.e., the SIR model equations in our case.

The application of PINNs to epidemiological models became particularly relevant during the COVID-19 pandemic. Many studies used PINNs as an inverse-problem solvers, to calibrate the parameters of epidemiological compartmental models. However, the model parameters has frequently been considered constant in time, e.g, [Torku et al., 2021; Malinzi et al., 2022], or with particular periodic dependencies on time [Berkhahn and Ehrhardt, 2022]. Schiassi et al. [2021] showed the computational efficacy of using PINNs to estimate constant parameters of different basic compartmental models under increasing levels of noise in the data. Long et al. [2021] considered a more realistic scenario, and used PINNs to accurately identify the time-varying transmission parameter in a *Susceptible-Infectious-Recovered-Deceased* (SIRD) model of COVID-19 when assimilating the reported infected cases in three USA States. Feng et al. [2022] proposed a similar approach to predict the number of active cases and removed cases in the US. Olumoyin et al. [2021] used PINNs to track the changes in transmission rate and the number of asymptomatic individuals for COVID-19, while Ning et al. [2023] presented a similar application to a *Susceptible-Exposed-Infectious-Recovered-Deceased* (SEIRD) model during the first months of the Italian outbreak. Bertaglia et al. [2022] constrained PINNs to satisfy an asymptotic-preservation property to avoid poor results caused by the multiscale nature of the residual terms in the loss function. Building on top of these examples, we aim to deeper explore the properties of PINNs as an inverse solver for the estimation of a time-dependent transmission rate in SIR models. In particular, we propose two modifications of the PINNs algorithm that grant faster convergence and more stable results.

The first proposed modification splits the PINN implementation in two steps. The motivation for this approach is that in the common PINN implementation for SIR-like models, the NNs representing the model state variables and, if present, the time-dependent parameters are calibrated together through the minimization of the loss function on the data and the model residual. This inverse problem is particularly complex and many epochs might be required to achieve convergence. Starting from the idea that the available epidemiological data, which are typically the daily or weekly reported infections, is directly associated to a model state variable, the split PINN approach is based on the following two steps: as first, construct the NN of the state variable associated to the data, e.g., the infected compartment, by minimizing the loss function based on the data; as second, calibrate the other NNs for the remaining state variables and parameters based on the NN computed in the first step and the minimization of the residuals of the governing equations. We will refer to the traditional PINN approach as *joint approach*, in contrast to the described *split approach*.

The second proposed modification reduces the number of NNs considered in the PINN approximation and, consequently, simplifies the structure of the loss function. This simplification is possible because, in simple SIR-based models, the transmission parameter and the infected compartment control the system dynamic. In fact, these functions allow to directly evaluate the other state variables, which are then redundant in the formulation of the loss function.

Our analysis compares the joint, split and reduced approaches in a sequence of synthetic test cases where we progressively challenge the structure of the transmission rate from constant, to a sinusoidal-like dependence on time, to a real scenario, and increase the noise on the synthetic reported data. The proposed test cases assume model parameters that are inspired by the first months of the COVID-19 outbreak in Italy. As a final example of application, the proposed PINN strategies are adapted in order to fit the real epidemiological data recorded in Italy. Due to the large uncertainties that characterize the real data on the reported infections, in this last setting we propose to include in the loss function also the data on the daily hospitalizations, which are a more reliable representation of the number of individuals with severe symptoms.

5.1 PINN solution to the SIR model

The well-known SIR model is largely adopted for the theoretical analysis of epidemics, and lies at the core of several more complex epidemiological models for real applications. At a given time t [T], the individuals in a population of dimension N [–] are subdivided into compartments on the basis of their epidemiological status, in this case the susceptible (S), the infected (I), and the recovered (R) individuals. The number of individuals in the three compartments changes in time under the assumption that,

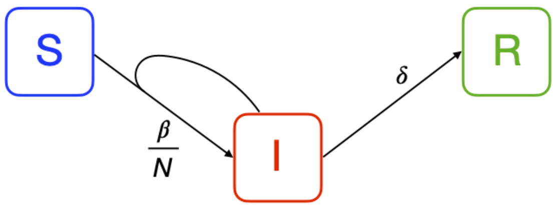


FIGURE 5.1: Scheme of the SIR compartmental model.

in a well mixed population, any susceptible individual can enter in contact with any infected individual, thus possibly becoming infected itself (Figure 5.1).

From a mathematical point of view, the strong form of the ordinary differential problem governing these dynamics can be stated as follows. Let $\mathcal{T} = [t_0, t_f] \subseteq \mathbb{R}^+ \cup \{0\}$ be the time domain of interest, with t_0 and t_f [T] the initial and final times of the simulation, respectively. Given the continuous functions $\beta(t) : \mathcal{T} \to \mathbb{R}^+$ and $\delta(t) : \mathcal{T} \to \mathbb{R}^+$, find $S(t) : \mathcal{T} \to [0, N]$, $I(t) : \mathcal{T} \to [0, N]$, and $R(t) : \mathcal{T} \to [0, N]$ such that:

$$\begin{cases} \dot{S}(t) = -\dfrac{\beta}{N} I(t)S(t) \\ \dot{I}(t) = \dfrac{\beta}{N} I(t)S(t) - \delta I(t) \\ \dot{R}(t) = \delta I(t) \end{cases} \quad , \quad \forall\, t \in \mathcal{T}, \tag{5.1}$$

and satisfying the initial conditions:

$$\begin{cases} S(t_0) = N - I_0 \\ I(t_0) = I_0 \\ R(t_0) = 0 \end{cases} \tag{5.2}$$

In Eqs. (5.1)-(5.2) β [T^{-1}] is the transmission rate controlling the average rate of the infection, δ [T^{-1}] is the mean rate of removal of the infected individuals that become recovered, and I_0 is the number of infected individuals (typically 1, but not necessarily) at the initial instant. Another relevant quantity used to set up the model is $D = \delta^{-1}$, i.e., the mean reproduction period [T] representing the average time spent by an individual in compartment I. Initial conditions for the spreading of a new disease assume that the population at the initial time is completely susceptible besides a small number I_0 of infected individuals.

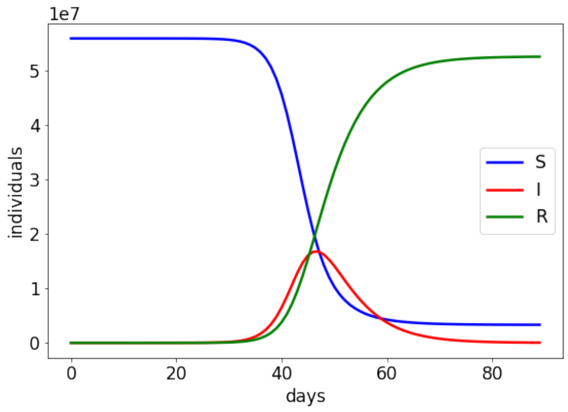


FIGURE 5.2: Solution of the SIR model (5.1) with $\mathcal{R}_0 = 3$ and constant parameters.

The basic reproduction number $\mathcal{R}_0$ [-] associated to this model reads $\mathcal{R}_0 = \beta(t_0)/\delta(t_0)$ and provides an estimate of the number of secondary infections generated by one infectious individual in a susceptible population, i.e. at the beginning of the epidemic. The threshold $\mathcal{R}_0 > 1$ indicates the occurrence of an outbreak, while $\mathcal{R}_0 < 1$ indicates that the number of infected individuals is rapidly decreasing. Note that, in a real population, the number of individuals in each compartment is a discrete variable, whose dynamic can be described by stochastic approaches, e.g., the Gillespie method or discrete Markov chains. Hence, the continuous deterministic model in Equations (5.1)-(5.2) is a valuable representation of the mean process in large populations.

Standard numerical *Ordinary Differential Equation* (ODE) solvers, such as Runge-Kutta-based methods, can provide an accurate solution to the differential problem (5.1)-(5.2). For $\mathcal{R}_0 > 1$ and constant parameters, the solution depicts an initial exponential-like increase in the number of infections up to a peak, and then a fast decrease due to the depletion of susceptible individuals (see Figure 5.2). However, it is clear that this dynamic does not correspond to what happens during an outbreak. The main challenge when using a model based on (5.1) to describe a real epidemic is that the transmission rate β and the mean reproduction period δ^{-1} can change in time because of many factors: social behaviors (individual awareness, increase or decrease of gatherings, mobility, social distancing), non-pharmaceutical interventions (use of devices that reduce transmission - such as masks, introduction of lock-downs), changes in the pathogen infectiousness due to new variants, reduction of the susceptibility of the population due to vaccination campaigns. In this evolving scenario, the effective reproduction number $\mathcal{R}_t$ [-] is the critical quantity that controls the spreading of the disease. $\mathcal{R}_t$ is the equivalent of $\mathcal{R}_0$ in time, i.e., $\mathcal{R}_t = \beta(t)/\delta(t) \cdot S(t)/N$, taking into account that the number of susceptible individuals decreases and the main parameters controlling the spreading of the disease generally change. An essential element for a reliable simulation is therefore the assessment of $\mathcal{R}_t$, hence $\beta(t)$ and $\delta(t)$ along with the compartment $S(t)$, from the available epidemiological data.

We develop and analyze a PINN-based approach to simultaneously solve the problem (5.1)-(5.2) and estimate the temporal values of the reproduction number by using a time series of infectious individuals as basic epidemiological information.

For the SIR model (5.1)-(5.2), we assume that the training data points for the fitting are the reported infections. Let $\tilde{I}_j$ be the number of reported infected individuals at times $\tilde{t}_j$, $j = 1, \ldots, N_d$. This might be subject to reporting errors, thus, in general $\tilde{I}_j \neq I(\tilde{t}_j)$. The residual of the governing equations is computed over N_c collocation points.

We aim at finding an approximate NN representation for the susceptible, infected and recovered individuals, $\hat{S}(t)$, $\hat{I}(t)$, $\hat{R}(t)$, respectively, along with the transmission rate $\hat{\beta}(t)$. The rate of removal from the infectious class δ is assumed constant in time now on. Since the state variables S, I, R span an extremely wide range of values (from zero to the population size $N > 10^6$), the functional search is optimized by a proper scaling:

$$S(t) = CS_s(t_s), \quad I(t) = CI_s(t_s), \quad R(t) = CR_s(t_s), \tag{5.3}$$

where C [-] is an appropriate constant and t_s is the dimensionless scaled temporal variable, $t_s = (t - t_0)/(t_f - t_0)$. The system of ODEs (5.1) for the scaled variables becomes:

$$\begin{cases} \dot{S}_s(t_s) = -C_1\beta_s(t_s)I_s(t_s)S_s(t_s) \\ \dot{I}_s(t_s) = C_1\beta_s(t_s)I_s(t_s)S_s(t_s) - C_2 I_s(t_s) \\ \dot{R}_s(t_s) = C_2 I_s(t_s) \end{cases}, \quad t_s \in [0, 1], \tag{5.4}$$

where $\beta_s(t_s) : [0, 1] \to \mathbb{R}^+$, $C_1 = (t_f - t_0)C/N$ and $C_2 = (t_f - t_0)\delta C$. The initial conditions (5.2) are correspondingly scaled as well as the infectious data $\tilde{I}_j = C\tilde{I}_{s,j}$ at times $\tilde{t}_{s,j} = (\tilde{t}_j - t_0)/(t_f - t_0)$.

The SIR model (5.4) does not consider death and birth processes and assumes a negligible mortality rate of the disease. Thus, the total population N is constant in time and equal to $N = S + I + R$. Under these hypotheses, the PINN model needs only two NNs representing the behavior of the population: one for the state variable of the susceptible individuals $\hat{S}_s$, and one for the infected individuals $\hat{I}_s$. The number of recovered individuals is computed as $\hat{R}_s = \frac{N}{C} - \hat{I}_s - \hat{S}_s$. A third NN is included for the estimation of the transmission rate $\hat{\beta}_s$. This approach consistently reduces the number of parameters to be tuned during the training.

It is important to underline that the state variables represent the number of individuals in a compartment, thus they all have positive outputs. Training the model without imposing this condition could lead to nonphysical negative NNs outputs. The non-negative constraint can be imposed in the NN in two alternative ways: inserting a penalty term for the negative values of th NNs (weak constraint) or building the NN architecture so as to allow for positive values only (hard constraint). The latter prescription can be met by setting the output activation function, i.e., the one related the last layer, equal for example to the square function. An experimental comparison between the two approaches shows that the latter is generally more effective and provides more robust results. The numerical outcomes that follow are therefore obtained by using the hard constraint prescription for the non-negativity of the solution. The same constraint is adopted to entail a positive value for β_s.

The selection of the loss function is one of the most sensitive steps in the PINN approach, given the multi-objective nature of the method. Using the MSE as loss measure, the objective is to minimize the mismatch on the N_d data:

$$\mathcal{L}_d(\hat{I}_s) = \omega_d \frac{1}{N_d} \sum_{j=1}^{N_d} \left[\hat{I}_s(\tilde{t}_{s,j}) - \tilde{I}_{s,j} \right]^2, \tag{5.5}$$

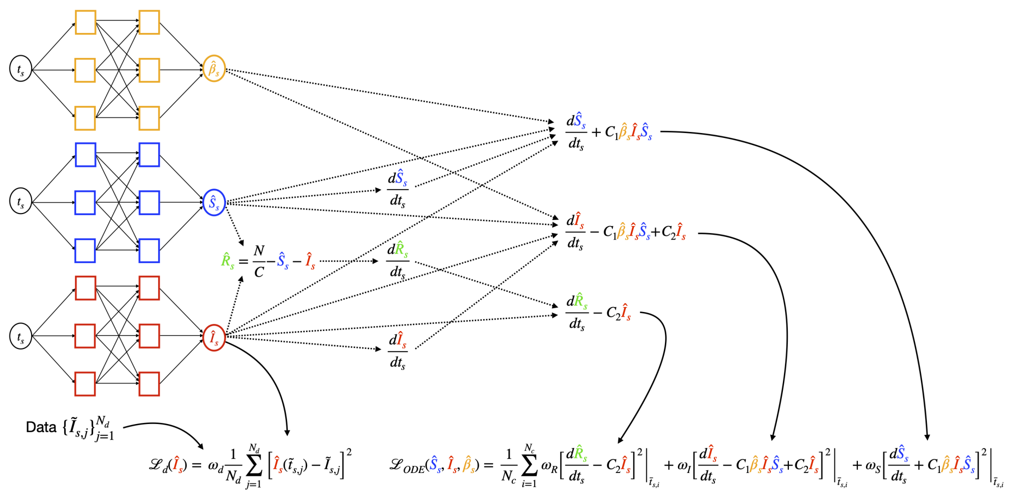

t_s
$\hat\beta_s$
$\hat S_s$
$\hat I_s$
$\hat R_s = \frac{N}{C} - \hat S_s - \hat I_s$
$\frac{d\hat S_s}{dt_s}$
$\frac{d\hat R_s}{dt_s}$
$\frac{d\hat I_s}{dt_s}$
$\frac{d\hat S_s}{dt_s} + C_1\hat\beta_s\hat I_s\hat S_s$
$\frac{d\hat I_s}{dt_s} - C_1\hat\beta_s\hat I_s\hat S_s + C_2\hat I_s$
$\frac{d\hat R_s}{dt_s} - C_2\hat I_s$
Data $\{\tilde I_{s,j}\}_{j=1}^{N_d}$
$\mathcal{L}_d(\hat I_s) = \omega_d \frac{1}{N_d}\sum_{j=1}^{N_d}\left[\hat I_s(\tilde t_{s,j}) - \tilde I_{s,j}\right]^2$
$\mathcal{L}_{ODE}(\hat S_s, \hat I_s, \hat\beta_s) = \frac{1}{N_c}\sum_{i=1}^{N_c} \omega_R\left[\frac{d\hat R_s}{dt_s} - C_2\hat I_s\right]^2\Big|_{\bar t_{s,i}} + \omega_I\left[\frac{d\hat I_s}{dt_s} - C_1\hat\beta_s\hat I_s\hat S_s + C_2\hat I_s\right]^2\Big|_{\bar t_{s,i}} + \omega_S\left[\frac{d\hat S_s}{dt_s} + C_1\hat\beta_s\hat I_s\hat S_s\right]^2\Big|_{\bar t_{s,i}}$

FIGURE 5.3: Diagram of the PINN model for the SIR equations with unknown $\beta(t)$. The parameters of the NNs for β, S, I are obtained by minimizing the loss functions on the infectious data, and on the residual and initial conditions of the model equations.

the squared norm of the residual of Equations (5.4) evaluated on N_c collocation points $\{\bar t_{s,i}\}_{i=1}^{N_c}$:

$$\begin{aligned}\mathcal{L}_{ODE}(\hat S_s, \hat I_s, \hat\beta_s) \quad = \quad & \frac{1}{N_c}\sum_{i=1}^{N_c}\left\{\omega_S\left[\frac{d\hat S_s}{dt_s} + C_1\hat\beta_s\hat I_s\hat S_s\right]^2\Big|_{\bar t_{s,i}}\right. \\ & \left. + \omega_I\left[\frac{d\hat I_s}{dt_s} - C_1\hat\beta_s\hat I_s\hat S_s + C_2\hat I_s\right]^2\Big|_{\bar t_{s,i}} + \omega_R\left[\frac{d\hat R_s}{dt_s} - C_2\hat I_s\right]^2\Big|_{\bar t_{s,i}}\right\},\end{aligned} \tag{5.6}$$

and the misfit on the initial conditions:

$$\mathcal{L}_{IC}(\hat S_s, \hat I_s) = \omega_{S_0}\left[\hat S_s(0) - \frac{N - I_0}{C}\right]^2 + \omega_{I_0}\left[\hat I_s(0) - \frac{I_0}{C}\right]^2 + \omega_{R_0}\hat R_s^2(0)\,, \tag{5.7}$$

where ω_* are proper weights needed to balance the relative importance of the entries arising from each contribution to the global MSE value. Figure 5.3 shows a diagram of the PINN implementation for the solution of the scaled SIR model (5.4).

We explore two possible approaches for the construction of the PINN model, indicated as *joint* or *split*. The joint approach aims to simultaneously calibrate $\hat S_s$, $\hat I_s$, and $\hat\beta_s$ by minimizing the joint loss function corresponding to the sum of $\mathcal{L}_d$, $\mathcal{L}_{ODE}$ and $\mathcal{L}_{IC}$:

$$\mathcal{L}_{\text{joint}}(\hat S_s, \hat I_s, \hat\beta_s) = \mathcal{L}_d(\hat I_s) + \mathcal{L}_{ODE}(\hat S_s, \hat I_s, \hat\beta_s) + \mathcal{L}_{IC}(\hat S_s, \hat I_s). \tag{5.8}$$

By distinction, the split approach subdivides the overall problem. First, $\hat{I}_s$ is independently calibrated on the data error $\mathcal{L}_d$ (5.5) only. In this case, a standard NN is used with weight $\omega_d = 1$, thus obtaining a differentiable regression function for the data. The only-data regression is followed by a fully-physics-informed regression, where the parameters defining $\hat{S}_s$ and $\hat{\beta}_s$ are trained by minimizing:

$$\mathcal{L}_{\text{split}}(\hat{S}_s, \hat{\beta}_s) = \mathcal{L}_{ODE}(\hat{S}_s, \hat{\beta}_s) + \mathcal{L}_{IC}(\hat{S}_s). \tag{5.9}$$

5.1.1 Reduced SIR model

The system of ODEs in (5.1) can be further reduced by directly considering the definition of the effective reproduction number $\mathcal{R}_t$. By easy developments, the model (5.1) becomes:

$$\begin{cases} \dot{I}(t) = \delta(\mathcal{R}_t - 1)I(t) \\ \dot{S}(t) = -\delta\mathcal{R}_t I(t) \end{cases}, \quad t \in [t_0, t_f]. \tag{5.10}$$

where the unknown functions are $I(t)$ and $S(t)$, and the state variable $R(t)$ is simply obtained from the consistency relationship $R(t) = N - S(t) - I(t)$. The initial conditions (5.2) still hold. The new system (5.10) can be solved sequentially by integrating the upper equation first and then computing $S(t)$ from the second equation.

This approach reduces the number of functions that are approximated by NNs to two, i.e., I and $\mathcal{R}_t$, and eliminates any redundant term in the loss function minimized in the PINN approach. The same scaling as in Equation (5.3) is used for the state variable I, so that the upper equation in (5.10) reads:

$$\dot{I}_s(t_s) = \delta(t_f - t_0)(\mathcal{R}_t - 1)I_s(t_s), \qquad t_s \in [0, 1]. \tag{5.11}$$

The NNs approximating the variables of interest, i.e., $\hat{I}_s$ and $\hat{\mathcal{R}}_t$, can be obtained by minimizing the mismatch on data $\mathcal{L}_d$ (5.5) and the squared norm of the residual of Equation (5.11) on N_c collocation points:

$$\mathcal{L}_{r,ODE}(\hat{I}_s, \hat{\mathcal{R}}_t) = \frac{1}{N_c} \sum_{i=1}^{N_c} \left[\frac{d\hat{I}_s}{dt_s} - \delta(t_f - t_0)(\hat{\mathcal{R}}_t - 1)\hat{I}_s \right]^2 \Big|_{\bar{t}_{s,i}} \tag{5.12}$$

Notice that in this case the contributions in $\mathcal{L}_d$ and $\mathcal{L}_{r,ODE}$ have a consistent size, hence there is no need for introducing the weight parameters ω_* to balance the loss function terms. For this reason, we simply set $\omega_d = 1$ in the expression (5.5).

The joint and split approaches can be formulated for this PINN-based model as well. The joint approach consists in training simultaneously the NNs $\hat{I}_s$ and $\hat{\mathcal{R}}_t$ by minimizing the total loss function:

$$\mathcal{L}_{r,\text{joint}}(\hat{I}_s, \hat{\mathcal{R}}_t) = \mathcal{L}_d(\hat{I}_s) + \mathcal{L}_{r,ODE}(\hat{I}_s, \hat{\mathcal{R}}_t). \tag{5.13}$$

By distinction, the split approach implies training $\hat{I}_s$ on the data only by the minimization of $\mathcal{L}_d$ in Equation (5.5). Then, the time-dependent parameter $\hat{\mathcal{R}}_t$ is obtained by minimizing:

$$\mathcal{L}_{r,\text{split}}(\hat{\mathcal{R}}_t) = \mathcal{L}_{r,ODE}(\hat{\mathcal{R}}_t). \tag{5.14}$$

Notice that in the reduced PINN model no initial condition is set, but we let the model deduce it from the data. From a theoretical viewpoint, initial conditions are not necessary because $\hat{I}_s$ is obtained from the data, while the governing differential equation (5.11) is used to calibrate $\hat{\mathcal{R}}_t$. This outcome is relevant because it replicates what typically happens in a real-case scenario, where there is no actual knowledge about the instant of beginning of the epidemic outbreak. In fact, the case 0 in most outbreaks is unknown and the conventional start of the epidemic has a number of infected individuals that is usually largely underestimated. The use of the reduced modeling approach makes it possible to remove the term related to the initial condition from the loss function.

5.1.2 SIR model with the hospitalization compartment

The reported infections can be often affected by large uncertainties. Especially at the beginning of an epidemic outbreak, the disease cannot be easily recognized, either because of the difficulty of correctly identifying the symptoms, or the absence of well-established detection and surveillance procedures, or the impossibility of reaching and testing all the people infected by the disease. Moreover, these data can be strongly affected by territorial peculiarities and the logistic of testing facilities. Hence, founding an epidemiological model on these pieces of information can undermine its reliability. A much less uncertain epidemiological datum is the daily number of individuals that require to be hospitalized. This is fraction of the overall number of infected individuals, however such a value can be representative of the entire I compartment by assuming that hospitalization is needed over a certain common threshold level of symptoms in the population.

We introduce a new variable, H, defined as:

$$H(t) = \delta \sigma I(t), \tag{5.15}$$

where σ represents the fraction of infected individuals moving to the hospitalized compartment. Note that also parameter σ might change in time, for example because of the insurgence of more aggressive variants or the improvement of home treatment. A more convenient formulation uses the cumulative number Σ_H of hospitalized individuals:

$$\Sigma_H(t) = \int_{t_0}^{t} H(z)\, \mathrm{d}z. \tag{5.16}$$

The new formulation of the updated SIR model can be therefore stated as follows. Given $\mathcal{R}_t : \mathcal{T} \to \mathbb{R}^+$, $\sigma(t) : \mathcal{T} \to [0,1]$, and $\delta(t) : \mathcal{T} \to \mathbb{R}^+$, find $\Sigma_H(t) : \mathcal{T} \to [0,N]$, $I(t) : \mathcal{T} \to [0,N]$, and $S(t) : \mathcal{T} \to [0,N]$ such that:

$$\begin{cases} \dot{\Sigma}_H(t) = \delta \sigma I(t) \\ \dot{I}(t) = \delta(\mathcal{R}_t - 1) I(t) \\ \dot{S}(t) = -\mathcal{R}_t \delta I(t) \end{cases} \quad , \quad \forall\, t \in \mathcal{T}, \tag{5.17}$$

with $R(t) = N - I(t) - S(t)$, the initial conditions (5.2) and $\Sigma_H(t_0) = 0$. The available information from the actual epidemiological data is the daily variation Δ_H of the cumulative number of hospitalized individuals:

$$\Delta_H(t) = \Sigma_H(t) - \Sigma_H(t-1) \simeq \dot{\Sigma}_H(t), \tag{5.18}$$

whose values represents the training dataset for the PINN approximation of system (5.17).

As previously done, the functional search of the approximating NNs is carried out on the properly scaled quantities $I(t) = CI_s(t_s)$ (see Equation (5.3)) and:

$$\Delta_H(t) = C_H \Delta_{H,s}(t_s), \tag{5.19}$$

with C_H the scaling factor. The upper equation in system (5.17) with the scaled quantities reads:

$$C_H \Delta_{H,s}(t_s) = \delta C \sigma_s(t_s) I_s(t_s), \tag{5.20}$$

with $\sigma_s : \mathcal{T} \to [0,1]$, while the second scaled equation is the same as in (5.11). Hence, the NNs needed to solve the SIR model with hospitalization data are $\hat{\Delta}_{H,s}$, $\hat{I}_s$, $\hat{\sigma}_s$, and $\hat{\mathcal{R}}_t$. The training data points for the fitting are both the scaled reported infections $\tilde{I}_{s,j}$ and the hospitalizations $\tilde{\Delta}_{H,s,j}$ at the scaled times $\tilde{t}_{s,j}$, $j = 1, \dots, N_d$. The NNs can be obtained by minimizing the mismatch (5.5) on the infection data and on the hospitalization data:

$$\mathcal{L}_H = \frac{1}{N_d} \sum_{j=1}^{N_d} \left[\hat{\Delta}_{H,s}(\tilde{t}_{s,j}) - \tilde{\Delta}_{H,s,j} \right]^2, \tag{5.21}$$

and the squared norm of the residuals of Equations (5.11) and (5.20) on N_c collocation points:

$$\begin{aligned} \mathcal{L}_{H,ODE}(\hat{\Delta}_{H,s}, \hat{I}_s, \hat{\sigma}_s, \hat{\mathcal{R}}_t) \quad &= \quad \frac{1}{N_c} \sum_{i=1}^{N_c} \left\{ \left[\frac{d\hat{I}_s}{dt_s} - \delta(t_f - t_0)(\hat{\mathcal{R}}_t - 1)\hat{I}_s \right]^2 \Big|_{\bar{t}_{s,i}} \right. \\ & \qquad \left. + \left[\hat{\Delta}_{H,s} - \frac{\delta C \hat{\sigma}_s}{C_H} \hat{I}_s \right]^2 \Big|_{\bar{t}_{s,i}} \right\}. \end{aligned} \tag{5.22}$$

The joint approach consists in the simultaneous estimate of $\hat{\Delta}_{H,s}$, $\hat{I}_s$, $\hat{\sigma}_s$, and $\hat{\mathcal{R}}_t$ by finding the minimum to the functional:

$$\mathcal{L}_{H,\text{joint}}(\hat{\Delta}_{H,s}, \hat{I}_s, \hat{\sigma}_s, \hat{\mathcal{R}}_t) = \mathcal{L}_d(\hat{I}_s) + \mathcal{L}_H(\hat{\Delta}_{H,s}) + \mathcal{L}_{H,ODE}(\hat{\Delta}_{H,s}, \hat{I}_s, \hat{\sigma}_s, \hat{\mathcal{R}}_t). \tag{5.23}$$

As for the PINN solution to the reduced SIR model, it is not necessary to include the mismatch on the initial conditions into the global loss function (5.23) because they are met through the available training data. Moreover, also the use of non-unitary weights ω_* for the different contributions to $\mathcal{L}_{H,\text{joint}}$ is not required since all terms are likely to have a similar magnitude.

In the split approach, $\hat{\Delta}_{H,s}$ is directly trained with the hospitalization data only by minimizing $\mathcal{L}_H$ in Equation (5.21). Then, $\hat{I}_s$ is computed from (5.20) as:

$$\hat{I}_s = \frac{C_H}{\delta C}\frac{\hat{\Delta}_{H,s}}{\hat{\sigma}_s}, \tag{5.24}$$

and $\hat{\sigma}_s$ and $\hat{\mathcal{R}}_t$ are trained by minimizing:

$$\begin{aligned}\mathcal{L}_{H,\text{split}}(\hat{\sigma}_s, \hat{\mathcal{R}}_t) \quad = \quad & \frac{1}{N_d}\sum_{j=1}^{N_d}\Big[\frac{C_H}{\delta C}\frac{\hat{\Delta}_{H,s}(\tilde{t}_{s,j})}{\hat{\sigma}_s(\tilde{t}_{s,j})} - \tilde{I}_{s,j}\Big]^2 + \\ & \frac{1}{N_c}\sum_{i=1}^{N_c}\Big\{\frac{C_H}{C}\Big[\frac{d}{dt_s}\Big(\frac{\hat{\Delta}_{H,s}}{\delta\hat{\sigma}_s}\Big) - (t_f - t_0)(\hat{\mathcal{R}}_t - 1)\frac{\hat{\Delta}_{H,s}}{\hat{\sigma}_s}\Big]\Big\}^2\Big|_{\bar{t}_{s,i}}.\end{aligned} \tag{5.25}$$

In real-world scenarios, the new daily infections is a more common piece of information than the total number of infected individuals. In order to include these data in the PINN model, we introduce the cumulative number Σ_I of infected individuals:

$$\Sigma_I(t) = \int_{t_0}^{t} I(z)\,\mathrm{d}z. \tag{5.26}$$

The variation of Σ_I in time coincides with negative variation of the class of susceptible individuals $S(t)$, so we can simply update the SIR model with hospitalization data (5.17) by replacing the last equation with:

$$\dot{\Sigma}_I(t) = \delta\mathcal{R}_t I(t) \tag{5.27}$$

Since the available information is the daily variation Δ_I of the cumulative number of infected individuals:

$$\Delta_I(t) = \Sigma_I(t) - \Sigma_I(t-1) \simeq \dot{\Sigma}_I(t), \tag{5.28}$$

we use these values as training data set. As usual, scaled values are considered such as $\Delta_I = C\Delta_{I,s}$ and we assume that the set of scaled values $\tilde{\Delta}_{I,s,j}$ is available at the training scaled times $\tilde{t}_{s,j}$, $j = 1, \ldots, N_d$, instead of $\tilde{I}_{s,j}$. The mismatch of $\hat{\Delta}_{I,s}$, i.e., the NN approximating $\Delta_{I,s}$, with the data is measured by:

$$\mathcal{L}_I = \frac{1}{N_d}\sum_{j=1}^{N_d}\Big[\hat{\Delta}_{I,s}(\tilde{t}_{s,j}) - \tilde{\Delta}_{I,s,j}\Big]^2, \tag{5.29}$$

while the squared norm of the residual reads:

$$\begin{aligned}\mathcal{L}_{HI,ODE}(\hat{\Delta}_{H,s},\hat{\Delta}_{I,s},\hat{I}_s,\hat{\sigma}_s,\hat{\mathcal{R}}_t) &= \frac{1}{N_c}\sum_{i=1}^{N_c}\Big\{\Big[\frac{d\hat{I}_s}{dt_s} - \delta(t_f-t_0)(\hat{\mathcal{R}}_t-1)\hat{I}_s\Big]^2\Big|_{\bar{t}_{s,i}} \\ &+\Big[\hat{\Delta}_{H,s} - \frac{\delta C\hat{\sigma}_s}{C_H}\hat{I}_s\Big]^2\Big|_{\bar{t}_{s,i}} + \Big[\hat{\Delta}_{I,s} - \delta\hat{\mathcal{R}}_t\hat{I}_s\Big]^2\Big|_{\bar{t}_{s,i}}\Big\}.\end{aligned} \tag{5.30}$$

Hence, with the joint approach we aim at minimizing the functional:

$$\mathcal{L}_{HI,\text{joint}}(\hat{\Delta}_{H,s},\hat{\Delta}_{I,s},\hat{I}_s,\hat{\sigma}_s,\hat{\mathcal{R}}_t) = \mathcal{L}_I(\hat{\Delta}_{I,s})+\mathcal{L}_H(\hat{\Delta}_{H,s})+\mathcal{L}_{HI,ODE}(\hat{\Delta}_{H,s},\hat{\Delta}_{I,s},\hat{I}_s,\hat{\sigma}_s,\hat{\mathcal{R}}_t). \tag{5.31}$$

By distinction, with the split approach we first train $\hat{\Delta}_{H,s}$ by the available data (see Eqiuation (5.21)). Then, we use Equation (5.24) for $\hat{I}_s$ and:

$$\hat{\Delta}_{I,s} = \frac{C_H}{C}\frac{\hat{\mathcal{R}}_t\hat{\Delta}_{H,s}}{\hat{\sigma}_s} \tag{5.32}$$

for $\hat{\Delta}_{I,s}$, and minimize the functional:

$$\begin{aligned}\mathcal{L}_{HI,\text{split}}(\hat{\sigma}_s,\hat{\mathcal{R}}_t) &= \frac{1}{N_d}\sum_{j=1}^{N_d}\Big[\frac{C_H}{C}\frac{\hat{\mathcal{R}}_t\hat{\Delta}_{H,s}(\tilde{t}_{s,j})}{\hat{\sigma}_s(\tilde{t}_{s,j})} - \tilde{\Delta}_{I,s,j}\Big]^2 \\ &+\frac{1}{N_c}\sum_{i=1}^{N_c}\Big\{\frac{C_H}{C}\Big[\frac{d}{dt_s}\Big(\frac{\hat{\Delta}_{H,s}}{\delta\hat{\sigma}_s}\Big) - (t_f-t_0)(\hat{\mathcal{R}}_t-1)\frac{\hat{\Delta}_{H,s}}{\hat{\sigma}_s}\Big]\Big\}^2\Big|_{\bar{t}_{s,i}}.\end{aligned} \tag{5.33}$$

This choice for the split approach is based on the fact that hospitalization data are usually more reliable than infected individuals, hence they are more appropriate for an only-data regression training.

5.2 Comparison of the joint and split approaches

We analyze the performance of the PINN-based approaches to estimate the state variables and identify the governing parameters of an epidemiological model mimicking

	State variables	Estimated parameters	Reference values	Training data
Case 1	S,I,R	β (constant)	$\beta_0 = 0.6\,\mathrm{d}^{-1}$	$\tilde{I}_j$ (Poisson error)
Case 2	S,I,R	β (time-dependent)	Synthetic β	$\tilde{I}_j$ (Poisson error)
Case 3	S,I,R	β (time-dependent)	COVID-19 $\mathcal{R}_t$	$\tilde{I}_j$ (Poisson error)
Case 4	I	$\mathcal{R}_t$	Synthetic β	$\tilde{I}_j$ (40% Gaussian error)
Case 5	I,Δ_H	$\mathcal{R}_t$,σ (time-dependent)	Synthetic β,σ	$\tilde{I}_j$ (40% Gaussian error), $\tilde{\Delta}_{H,j}$ (Poisson error)
Case 6	Δ_I,Δ_H	$\mathcal{R}_t$,σ (constant)	COVID-19 $\mathcal{R}_t$	$\tilde{\Delta}_{I,j}$, $\tilde{\Delta}_{H,j}$ (COVID-19 dataset)
Case 7	Δ_I,Δ_H	$\mathcal{R}_t$,σ (time-dependent)	COVID-19 $\mathcal{R}_t$	$\tilde{\Delta}_{I,j}$, $\tilde{\Delta}_{H,j}$ (COVID-19 dataset)

TABLE 5.1: Different scenarios adopted to analyze the performance of the proposed PINN-based approaches.

the setup of the first 90 days of a COVID-like disease outbreak in Italy. The total population is set to $N = 56 \times 10^6$ and the mean infectious period to $D = 5$ days, which is an estimate used for COVID-19 [Bertuzzo et al., 2020]. Therefore, in the following we will assume that $\delta = D^{-1}$ is constant in time. The initial value of infectious individuals I_0 is set to 1. The accuracy of the trained NNs is evaluated by the 2-norm of the error with respect to the 2-norm of the reference solution:

$$e_r = \frac{\|\hat{y} - y_{ref}\|_2}{\|y_{ref}\|_2}, \tag{5.34}$$

where y can be either one of the state variables, or a time-dependent parameter. The relative error (5.34) is numerically computed by using 90 points equally spaced in the domain. Our PINN-based approach is implemented by making use of the SciANN software library [Haghighat and Juanes, 2021].

We consider a number of scenarios, summarized in Table 5.1, differing for the reference SIR model and state variables of interest, the selection of the estimated governing parameters, and the available training data.

For the estimation of the transmission rate $\beta(t)$ in the basic SIR model (5.1)-(5.2), we consider three different scenarios:

- Case 1: constant β. We use this scenario to compare the efficiency of the joint and split approaches (5.8) and (5.9), respectively;
- Case 2: synthetic time-dependent $\beta(t)$, where the reference values are provided as an analytical function;
- Case 3: an application to a real-case scenario where the reference $\beta(t)$ is obtained from the estimates of $\mathcal{R}_t$ in the first months of the COVID-19 epidemic outbreak

in Italy.

In each case, the training data are the number of infectious individuals per day. These are synthetically generated by numerically integrating the system (5.1) using the selected reference function for $\beta(t)$. In particular, we used $N_d = 90$ training data points (one value per day). To take into account possible reporting errors, the data $\tilde{I}_j$ for each time $\tilde{t}_j$ are obtained by sampling from a Poisson distribution having as mean $I(\tilde{t}_j)$. This kind of Poisson error is frequently assumed on data arising from a counting process. The infectious data and the epidemiological model are scaled by a factor $C = 10^5$ (see Eq. (5.3)). The NNs for $\hat{S}_s$ and $\hat{I}_s$ are built with 4 hidden layers, 50 neurons for each and $\tanh$ as activation function, while $\hat{\beta}_s$ has 4 hidden layers with 100 neurons. In Case 1 the constant transmission rate is treated as a single parameter in the training. The output activation is set to x^2, as stated in Section 5.1. We consider $N_c = 6000$ collocation points randomly sampled in $[t_0, t_f]$ from a uniform distribution, Adam optimization algorithm [Kingma and Ba, 2015] with a learning rate equal to 0.001, and Glorot initialization [Glorot and Bengio, 2010] of the NNs. In the joint approach we trained the NNs for 5000 epochs with batches containing 100 training points, while in the split approach we set the number of epochs to 3000 for the data fitting training and to 1000 for the fully-physics-informed regression, with a batch size equal to 10 and 100, respectively. The weights ω_* are calibrated in the fitting process using the eigenvalues of the NTK [Wang et al., 2021b].

The reduced SIR model (5.10) is used to explore a more realistic scenario with strongly perturbed data. The motivation is that the data collected during the beginning of an outbreak are particularly subject to reporting errors. For example, the lack of a proper surveillance system and the low capacity in testing the presence of an infection may lead to an underestimation of the ongoing transmission. The joint and split approaches (5.13) and (5.14), respectively, are used to estimate the governing parameter $\mathcal{R}_t$ in the following inverse problem:

- Case 4: synthetic time-dependent $\beta(t)$ (as in Case 2), subject to a larger error noise on the infectious data.

The same NN architectures as described before for $\hat{\beta}_s$ is used for $\hat{\mathcal{R}}_t$ as well.

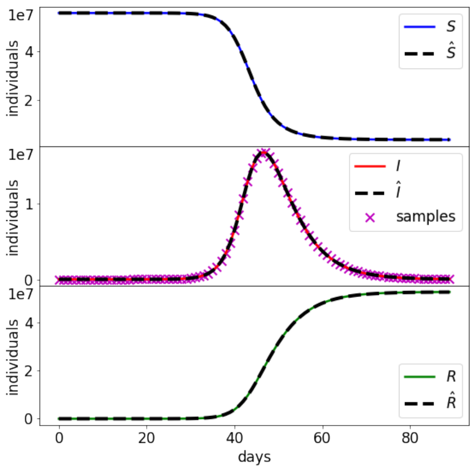
Finally, the performances of the joint and split PINN approaches in the SIR model with hospitalization datafor the simultaneous calibration of parameters σ and $\mathcal{R}_t$ are compared in the following realistic case scenarios:

- Case 5: an adaptation of the synthetic scenario described in Case 4, considering both infections data, hospitalization data, and a time-dependent σ;
- Case 6: an application to the real data collected in Italy during the first months of the COVID-19 pandemic, considering as unknowns the value of $\mathcal{R}_t$ (time-dependent) and σ (constant).
- Case 7: the same of Case 6, but having both $\mathcal{R}_t$ and σ as functions of time.

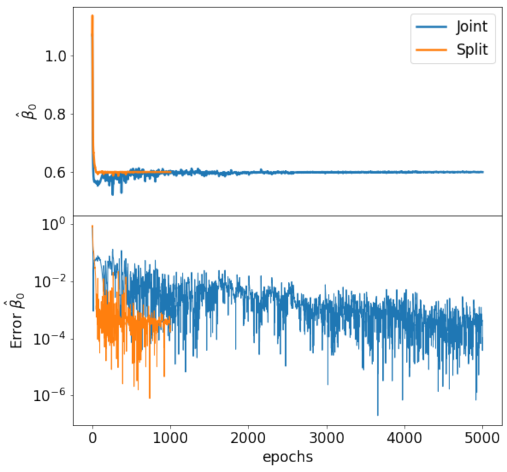
The NN for $\hat{\mathcal{R}}_t$ has 4 hidden layers with 100 neurons each, $\hat{\sigma}_s$ has 10 hidden layers with 5 neurons each. In Case 5, the synthetic data of the daily hospitalizations, $\{\tilde{\Delta}_{H,j}\}_{j=1}^{N_d}$, are obtained by sampling from a Poisson distribution having as mean value the reference solution. The scaled values are obtained by setting $C_H = 10^3$. In Cases 6 and 7, we consider the epidemiological data provided by the Italian surveillance system EpiCentro [2020] from February 21st, 2020 to May 20th, 2020. The period coincides with the advent of the disease and its initial spread. The vaccination campaign was not started yet and possible reinfections are negligible. The Italian dataset contains the number of new daily hospitalizations and reported infections, $\{\tilde{\Delta}_{H,j}\}_{j=1}^{N_d}$ and $\{\tilde{\Delta}_{I,j}\}_{j=1}^{N_d}$, respectively, and supplies an estimate of the COVID-19 reproduction number $\mathcal{R}_t$ based on Cori et al. [2013]. The scaled values are obtained by setting C and C_H equal to the maximum experimented values for Δ_I and Δ_H, respectively, in the 90 days taken into consideration. New infections are multiplied by a reporting ratio $\alpha_r = 6$, following the estimate from Italian Institute of Statistic based on the sierological data [ISTAT, 2020].

5.2.1 Case 1: constant parameter

The first scenario assumes a constant transmission rate during the simulation. The reference value for the parameter is fixed to $\beta = \beta_0 = 0.6\ \mathrm{d}^{-1}$, corresponding to a basic reproduction number $\mathcal{R}_0 = 3$, which is an estimate of the basic reproduction number in the COVID-19 epidemic in Italy. The resulting system dynamic is shown in Figure 5.4a. The evaluation of a constant parameter does not require an additional NN for β,



(A)



(B)

FIGURE 5.4: Case 1: Comparison between (A) reference solutions of the SIR model (5.1) and PINN approximations in the split approach; (B) β_0 identification during training of the joint and the split methods.

	Daily data		Weekly data	
	Joint	Split	Joint	Split
Training time [s]	2223	719	1977	493
Error S	2.763×10^{-3}	6.221×10^{-4}	2.833×10^{-2}	2.247×10^{-2}
Error I	3.846×10^{-3}	8.444×10^{-4}	4.544×10^{-2}	4.245×10^{-2}
Error R	3.258×10^{-3}	7.434×10^{-4}	3.276×10^{-2}	2.491×10^{-2}
Error β	5.086×10^{-3}	3.587×10^{-4}	4.693×10^{-2}	5.187×10^{-2}
PINN $\hat{\beta}$	0.59738	0.60007	0.57323	0.56888

TABLE 5.2: Case 1: Training time, approximation errors, and estimations of the joint and split methods for a constant transmission rate $\beta = \beta_0 = 0.6\ \text{d}^{-1}$. The daily and weekly infection data two correspond to $N_d = 90$ and $N_d = 13$, respectively.

and it is implemented through an object of the SciANN parameter class with the additional non-negative constraint. Both the joint and split approaches obtain solutions that match well the reference dynamic (see Figure 5.4a for the results of the split approach). Figure 5.4b shows that the convergence of the joint approach to the reference β value is slower than the split approach. In fact, there are strong oscillations during training. The split approach mitigates these oscillations and reaches a faster convergence to the reference value.

Table 5.2 provides a quantitative comparison of the resulting errors for each state variable along with the training times. Note that for the split approach the sum of the only-data and physics-informed training is reported. In both approaches the PINN

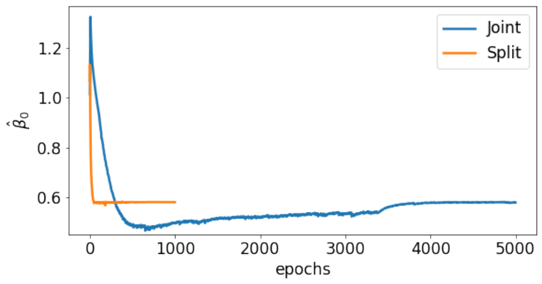


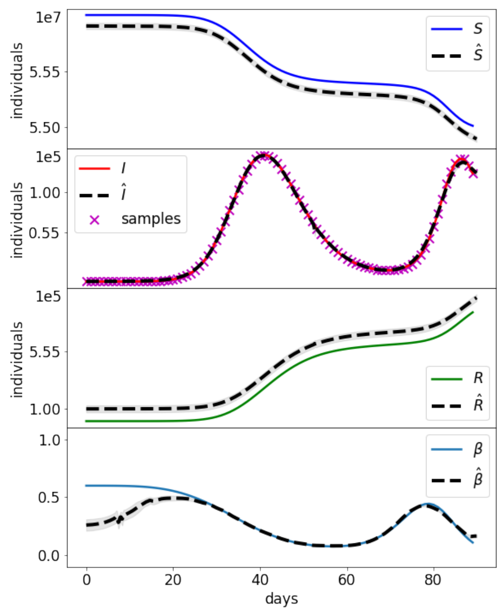
FIGURE 5.5: Case 1. Comparison between β_0 identification during the training of the joint and split methods for the weekly infection data ($N_d = 13$).

solution reaches an acceptable accuracy, with a relative error on the order of 10^{-3} for all state variables, resulting in a good estimate of the unknown parameter β as well. The split approach reduces the total training time about by a factor 3, and improves the accuracy with respect to the joint approach by one order of magnitude on average, hence it appears to be in this case largely preferable.

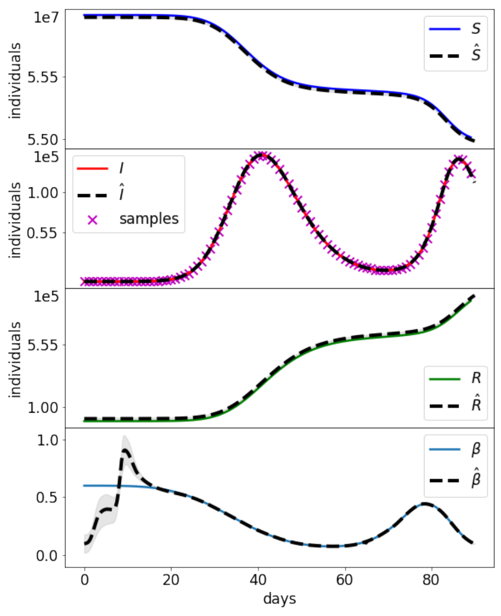
A second test is carried out by changing the amount of available data for the training. For instance, we consider only weekly values for the infection data, which are more likely to compensate the errors and oscillations of daily data. This reduces the number of training points to $N_d = 13$. The model is built and trained as stated in Section 5.2, with the difference that the training of $\hat{I}_s$ in the split method is performed at each epoch on the whole data set and the mini-batches are not needed. Moreover, given the overall small size of the training data set, we can decrease the maximum number of epochs in the data regression from 3000 to 1000. Training times and approximation errors reported in Table 5.2 show an equivalent outcome in terms of accuracy, but with a strong gain in training time for the split approach. By reducing the number of samples the amount of information is smaller, hence the accuracy decreases with respect to a daily updated information. The convergence of the parameter β has a similar behavior to the one shown previously (Figure 5.5), with the split method reducing both the oscillations and the convergence time.

5.2.2 Cases 2 and 3: Time-dependent transmission rate $\beta(t)$

Case 2 consists of a simulation of disease spread according to the time-dependent transmission rate plotted in Figure 5.6. This $\beta(t)$ behavior implies two waves of infection, with a maximum number of infectious individuals two orders of magnitude lower than in Case 1. As in Case 1, the numerical integration of the SIR model (5.1)-(5.2) provides the reference solution, from which we take the perturbed synthetic samples on the infectious individuals to train the neural networks. The transmission rate is here approximated by a NN, $\hat{\beta}_s$, since it is a function of time. Notice that even $\hat{\beta}_s$ has a quadratic output activation to ensure its non-negativity.



(A) Joint



(B) Split

FIGURE 5.6: Case 2: Comparison between the reference solution of the SIR model and the PINN approximations with the joint (A) and split (B) approach. Grey bands provide the confidence interval for one standard deviation.

Figure 5.6 shows the reference values of the unknowns and the corresponding PINN approximations. The dashed lines correspond to the mean outcome from 10 different runs, while grey bands provide the confidence interval for one standard deviation. Usually, the split method provides more stable and accurate results, with smaller variations from one run to another. The higher stability and speed of convergence of the split approach can be also appreciated from the error behavior on $\hat{\beta}_s$ during the training (Figure 5.7). The overall performance of the PINN approaches is summarized in Table 5.3.

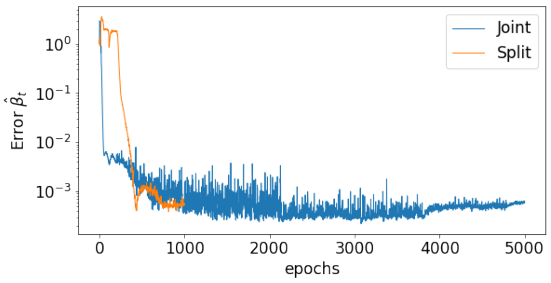


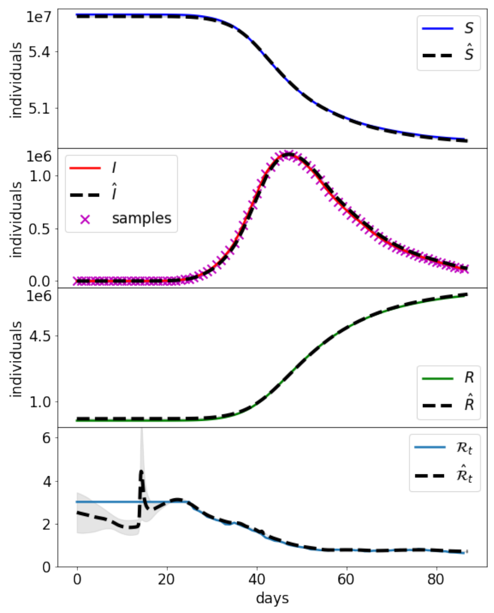
FIGURE 5.7: Case 2: Comparison between the errors on $\hat{\beta}_s(t)$ during the training with the joint and split approach.

	Joint	**Split**
Case 2		
Training time [s]	1936	717
Error S	1.814×10^{-3}	3.381×10^{-4}
Error I	2.501×10^{-2}	7.754×10^{-3}
Error R	2.288×10^{-1}	4.251×10^{-2}
Error β	2.678×10^{-1}	4.127×10^{-1}
Error β (last 70d)	5.979×10^{-2}	1.563×10^{-2}
Case 3		
Training time [s]	1906	726
Error S	1.935×10^{-3}	5.031×10^{-4}
Error I	4.729×10^{-3}	6.006×10^{-3}
Error R	2.805×10^{-2}	7.101×10^{-3}
Error $\mathcal{R}_t$	4.692×10^{-1}	6.988×10^{-1}
Error $\mathcal{R}_t$ (last 70d)	5.341×10^{-2}	6.701×10^{-2}

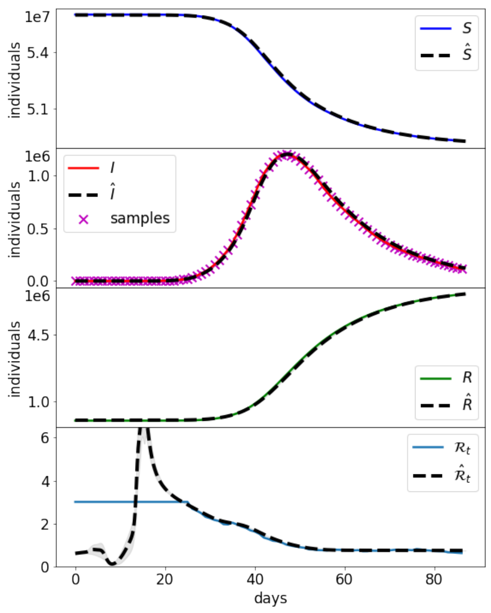
TABLE 5.3: Cases 2-3: Training time and approximation errors for the state variables and the estimated parameters with the joint and split approach.

Both approaches, however, fail to estimate the initial values of β (Figure 5.6). The low number of infected individuals and the presence of perturbed data produce an initial dynamic that the NNs erroneously learn by considering a larger number of initial infected individuals and a lower initial transmission rate. These kinds of errors represent a common hurdle in epidemiology, as the evolution of a disease is extremely difficult to be identified at the beginning of the epidemic outbreak under the assumption of a temporal-depending transmission rate. For this reason, Table 5.3 provides also the error for β during the last 70 days of simulation. These small errors confirm the accuracy of the PINN estimates when the data provide a clear signal.

Case 3 has the same setting as Case 2 but considers as reference values for the effective reproduction number, and consequently the transmission rate in the model, the estimates supplied by the Italian Institute of Health ISS [EpiCentro, 2020], depicted in Figure 5.8. For the first 20 days, the values have been kept equal to 3.012, in order to



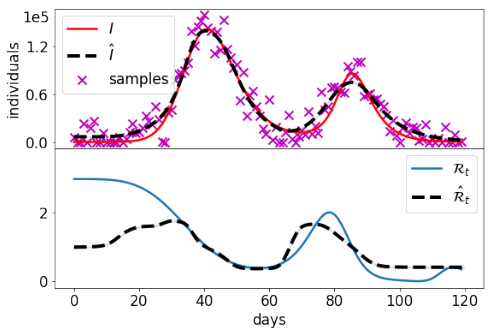
(A) Joint



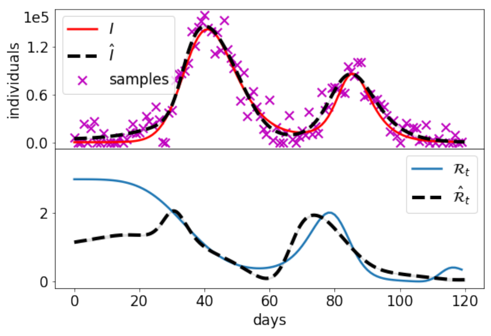
(B) Split

FIGURE 5.8: Case 3: Comparison between the reference solution of the SIR model and the PINN approximations with the joint (A) and split (B) approach. Grey bands provide the confidence interval for one standard deviation.

simulate the free transmission of the pathogen in a completely susceptible population without restrictions in contacts. This application to a realistic scenario reproduces the first three months of the Italian COVID-19 epidemic, from February 21^{st} to May 20^{th}, 2020. The epidemiological data for training the PINN approaches are synthetically generated as in Case 2. Figure 5.8 and Table 5.3 contain the outcome of the joint and split methods. Similarly to Case 2, the estimation of the temporal evolution of the transmission rate is hard for the initial times. Both methods can achieve a good accuracy after the first 20 days, as it can be deduced from Figure 5.8 and from the errors associated to the last 70 days of the simulation. The matching is quite accurate, with relative errors on the order of 10^{-3}, and the variance from 10 different runs is also limited. While the gain in accuracy of the split method is not so clear in this case, the benefit in the training duration is still relevant. Similar results in terms of accuracy has been obtained on these test cases when using the reduced implementation (5.10) of the split and joint approaches.



(A) Joint



(B) Split

FIGURE 5.9: Case 4: Reference solution and PINN approximation with the joint (A) and split (B) approach in the reduced SIR model (5.10) and noisy data.

5.2.3 Cases 4 and 5: Reduced model and hospitalization data

Case 4 is used to investigate the performance of the joint and split approaches for the reduced model (5.10) in a synthetic scenario where the data are subject to large errors. The unknown time-dependent transmission rate is the same as the one of Case 2, whose corresponding reproduction number is shown in Figure 5.9.

The time domain is extended to $t_f = 120$ days, which implies two complete waves of infection. To simulate the typically large uncertainties on the reported daily infections, we generate synthetic noisy data by perturbing the numerical solution for $I(\tilde{t}_j)$ with a Gaussian error having zero mean and coefficient of variation of 40%. The data are then rounded to the closest integer with the negative values set to 0. Finally, the scaling factor C is set to 10^5 with $N_d = 120$.

The outcome of the PINN approximations are provided in Figure 5.9 and Table 5.4. The larger errors on the data reduce the accuracy in the estimate of the reproduction number. The two approaches are almost equivalent in terms of accuracy, but the split one has faster training times. Clearly, a large uncertainty in the training data reduces the reliability of the split approach, which mostly relies on this piece of information, while the joint approach compensate possible errors on data with the residuals of the governing equations. Nevertheless, the accuracy in the reproduction of the $\mathcal{R}_t$ appears to be in any case fairly satisfactory, given the large reported errors.

In Case 5 we try to improve the $\mathcal{R}_t$ estimates obtained in Case 4 by introducing the

	Joint	**Split**
Case 4		
Training time [s]	1658	1053
Error I	1.411×10^{-1}	1.331×10^{-1}
Error $\mathcal{R}_t$	4.961×10^{-1}	4.744×10^{-1}
Error $\mathcal{R}_t$ (last 100d)	3.141×10^{-1}	3.336×10^{-1}
Case 5		
Training time [s]	1829	1445
Error Δ_H	8.783×10^{-3}	4.722×10^{-3}
Error I	1.209×10^{-1}	7.434×10^{-2}
Error $\mathcal{R}_t$	4.407×10^{-1}	4.992×10^{-1}
Error $\mathcal{R}_t$ (last 100d)	2.410×10^{-1}	1.240×10^{-1}
Error σ	3.858×10^{-1}	1.417×10^{-1}
Error σ (last 100d)	2.626×10^{-1}	1.145×10^{-1}

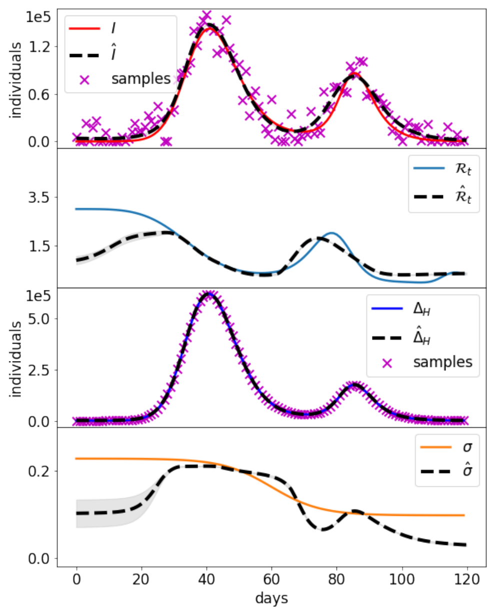
TABLE 5.4: Cases 4-5: Training time and approximation errors for the state variables and the estimated parameters with the joint and split approach.

information on the daily hospitalizations. This entails that PINN trains also the NN for parameter σ, that describes the fraction of infected individuals that become hospitalized. The reference σ for this case is shown in Figure 5.10. The results of the joint and split approaches are summarized in Figure 5.10 and Table 5.4. The inclusion of more reliable data implies a better estimate of $\mathcal{R}_t$ with respect to Case 4. This is particularly evident for the split approach, which is able to provide good estimates of both temporal depending parameters ($\mathcal{R}_t$ and σ), and thus, lower errors (Table 5.4).

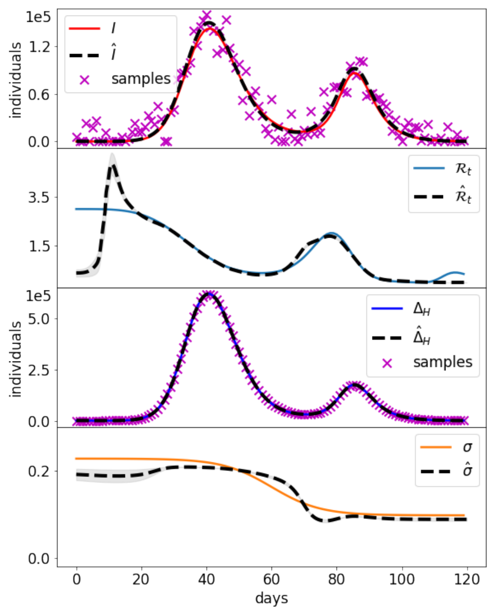
5.2.4 Cases 6 and 7: Application to real data

Cases 6 and 7 use the same setup as Case 5, but with the real epidemiological data collected during the Italian COVID-19 epidemic outbreak. The fraction of the number of infected individuals that becomes hospitalized, σ, is assumed to be constant in time in Case 6 and time-dependent in Case 7. The goal is to estimate $\mathcal{R}_t$ and σ by means of the presented PINN approaches. We use the reproduction number evaluated by ISS (Figure 5.11) as a reference value for the comparison.

The results of the joint and the split approaches are shown in Figures 5.11 and 5.12 for Cases 6 and 7, respectively. In Case 6 the joint approach provides an acceptable accuracy on the new infection data, while the peak of the daily hospitalizations is underestimated (Figure 5.11a). The split approach, instead, is firstly trained on the daily hospitalization well retrieving these high-fidelity data. The second part of the training, considering both the daily infections and the reduced model equations, does



(A) Joint



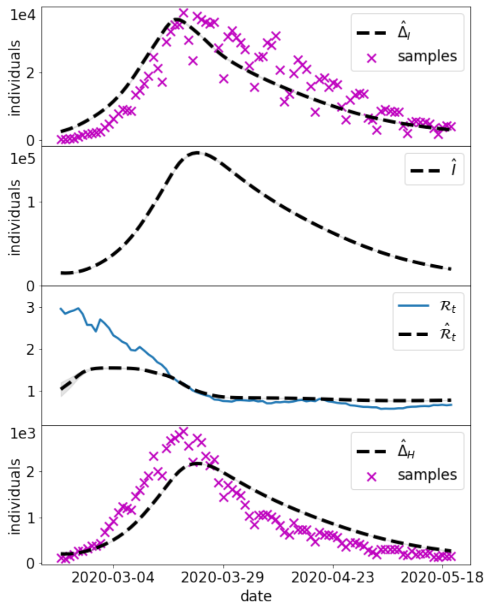
(B) Split

FIGURE 5.10: Case 5: Reference solution and PINN approximation in the joint (A) and split (B) approaches when considering the daily data on both hospitalizations and infections.

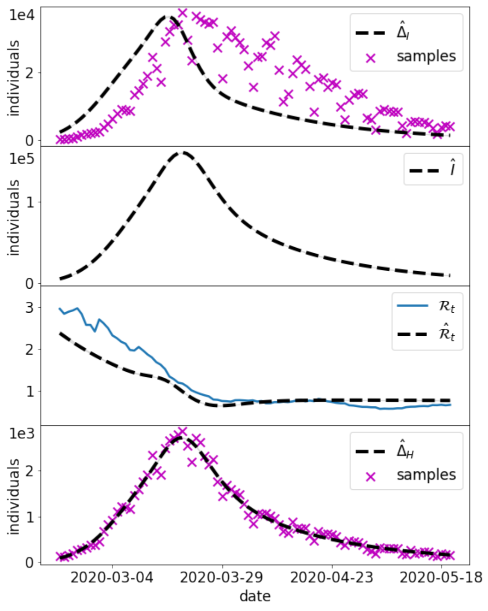
not achieve the same level of accuracy for the infection data with respect to the joint approach (Figure 5.11b). For what concerns the estimation of $\mathcal{R}_t$, the joint approach strongly underestimates its value at the beginning of the epidemic. By distinction, the split approach provides a trend of $\mathcal{R}_t$ that is fully consistent with the ISS estimates, i.e., a decrease during the first weeks of the epidemic, followed by an almost stationary value around 1 during the recession phase. The overall performance of the PINN approaches are summarized in Table 5.5.

	Joint	**Split**
Case 6		
Training time [s]	1655	1042
Error $\mathcal{R}_t$	3.881×10^{-1}	2.495×10^{-1}
Error $\mathcal{R}_t$ (last 100d)	1.474×10^{-1}	1.821×10^{-1}
Estimated $\hat{\sigma}$	0.0686	0.0849
Case 7		
Training time [s]	1778	1101
Error $\mathcal{R}_t$	3.646×10^{-1}	2.236×10^{-1}
Error $\mathcal{R}_t$ (last 100d)	1.682×10^{-1}	1.742×10^{-1}

TABLE 5.5: Cases 6-7: Training time and approximation errors for the state variables and the estimated parameters with the joint and split approach.



(A) Joint



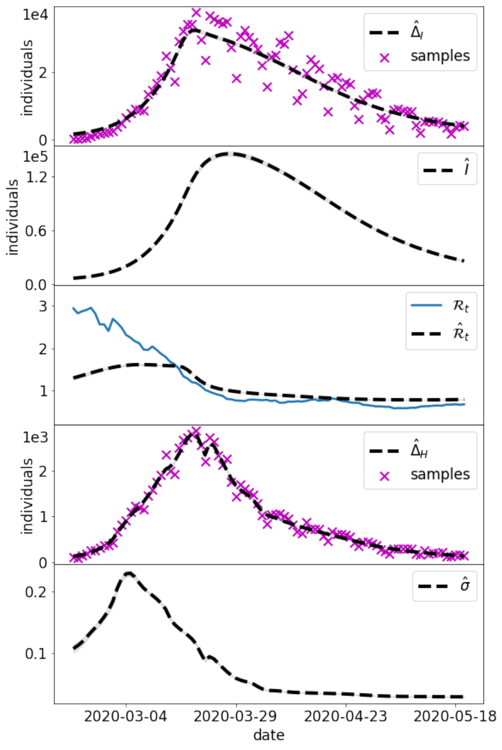
(B) Split

FIGURE 5.11: Case 6: PINN approximations with the joint (A) and split (B) methods for the real Italian COVID-19 outbreak with σ as a constant parameter. The blue trajectory of $\mathcal{R}_t$ is the official estimate by ISS.

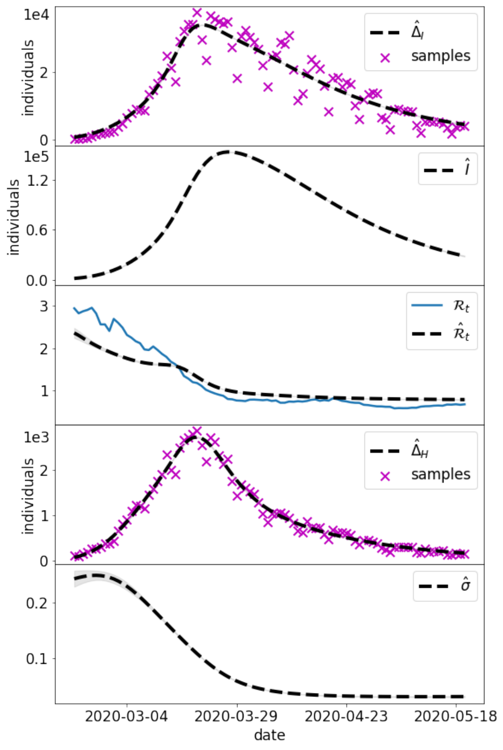
Finally, considering a time-dependent σ (Case 7) helped both approaches to improve the estimate of both hospitalized and infectious data. Both approaches depict a similar trend for σ, showing that the fraction of infected individuals that became hospitalized decreased during March 2020 from a peak of about 23% to about 3%, which is a realistic outcome in the framework of Italian COVID-19 outbreak. The main differences among the two approaches are still at the beginning of the outbreak, where the joint approach suggests a lower value of σ, while underestimating the values of $\mathcal{R}_t$. The training time required by the split approach was about 40% of the of time for the joint approach (Table 5.5).

5.3 Discussion

Synthetic test Cases 1-3 showed that, when infectious data are subject to small errors, both the split and joint PINN approaches are able to retrieve with high accuracy the system dynamics. The initial training on the data of the split approach provides a



(A) Joint



(B) Split

FIGURE 5.12: Case 7: PINN approximations with the joint (A) and split (B) methods for the real Italian COVID-19 outbreak with σ as a time-dependent parameter. The blue trajectory of $\mathcal{R}_t$ is the official estimate by ISS.

clear advantage when minimizing the residual on the model equations and estimating the reproduction number. In fact, it achieves lower errors (up to an order of magnitude) with faster computational times (speed up larger than 60% in Cases 2 and 3). This is probably due to more stable results during the training epochs, as depicted in Figure 5.4b for a constant transmission (Case 1), and Figure 5.7 for a time-dependent transmission (Case 2). However, the simultaneous estimate of the initial conditions and the initial transmission proved to be particularly challenging for both PINN approaches (Figures 5.6 and 5.8). In fact, even small errors on the data become particularly relevant when there are low number of infections, such as at the beginning of the epidemic. Model results could improve by assuming a constant initial transmission rate, which is generally in agreement with the free circulation of the pathogen in absence of interventions.

The large errors that typically characterize the data on reported daily infections might deteriorate the retrieval of the temporal changes of the transmission rate and the associated effective reproduction number (Case 4, Figure 5.9). For this reason, numerous epidemiological analysis are based on data that are not biased by the surveillance system, such as daily hospitalization data, e.g., Bertuzzo et al. [2020]. The synthetic test Case 5 shows that the use of daily hospitalizations and infections into the reduced PINN model allowed to improve the accuracy in the estimation of the uncertain time-varying parameters, in particular both the effective reproduction number $\mathcal{R}_t$ and the fraction of infected individuals requiring hospitalization σ. The split approach still outperforms the joint counterpart with 20% savings in the training cost, however both approaches still present limitations at the beginning of the outbreak.

The application to the Italian COVID-19 epidemic (Cases 6 and 7) emphasizes the importance of considering the fraction of hospitalized individuals, σ, as a temporal-dependent parameter. In fact, the PINN approximations were not able to accurately follow both time series of daily hospitalized and infectious data when considering a constant σ (Figure 5.11). Results notably improve in the case of a temporal-dependent σ (Figure 5.12). Many modeling studies assume a constant σ, with possible temporal variations assigned only on the arrival of new disease variants or after the deployment of vaccines. Other processes that might directly impact this parameter are typically neglected. For example, in the early stages of the outbreak, the fear of the new disease might prompt many symptomatic infected individuals to seek health care at the hospital (thus generating a large value of σ). The subsequent overcrowding of the hospitals and improvement of treatment at home might reduce the value of σ in time. The time-dependent σ values estimated by the PINN approaches (Figure 5.12) show exactly such a dynamic, with small differences at the beginning. We argue that also in this application the split approach outperforms the joint one. Besides the advantages in the computational times (40% faster), the effective reproduction number obtained with the split approach depicts a closer trajectory to the reference $\mathcal{R}_t$ estimated by the Italian Institute of Health ISS (Figure 5.12).

This study focuses on the standard SIR model. While the split approach can be easily adapted to more complex compartmental models (which, for example, consider an exposed compartment, deaths, re-infections, and vaccinations), the reduced equations described in (5.10) will require ad-hod formulations depending on the model.

In conclusion, the proposed split PINN-based approach is a robust and easy-to-implement tool to monitor the initial spreading of a disease. It provides estimates of the temporal changes in the model parameters, which is essential to produce more accurate short-term forecasts.

Chapter 6

Conclusions

In this thesis, we focussed on exploring of the application of *Physics-Informed Neural Network* (PINN) across diverse domains, spanning hydro-poromechanics and epidemiology. The investigation encompassed various test cases and applications, providing insights into the potential of PINNs in solving complex problems and contributing to advancements in both fields. The examination included employing PINNs for forward solution modeling in hydro-poromechanics, leveraging these NNs for parameter estimation in the same context, and extending their application to epidemiological modeling.

The motivation behind this study stemmed from the need for robust, computationally efficient tools capable of handling intricate phenomena in hydro-poromechanics and epidemiology. Traditional approaches faced challenges in capturing the complex dynamics of these systems, dealing with high computational times and resources, the need of almost real-time data assimilation, and the problem specificity of the models. In this context, PINNs, with their ability to incorporate physics-based constraints into machine learning frameworks, can present a promising avenue for addressing these challenges.

PINNs were employed not only for generating forward solutions but also for estimating the governing problem parameters within the respective domains. Enhancing the assimilation of real-world data, particularly in scenarios with limited or noisy information, remains a pivotal area for improvement, and the development of robust data assimilation techniques could further enhance the accuracy of parameter estimations.

- In hydro-poromechanical applications in the geosciences, the accurate modeling of subsurface processes is paramount for mitigating geological hazards, optimizing resource extraction, and ensuring the sustainability of civil engineering projects.

- In epidemiology, the refined models bear implications for public health strategies, offering a more precise understanding of disease spread and aiding in the formulation of effective intervention measures.

The interdisciplinary nature of this work showcases the versatility of PINNs as a powerful tool in bridging the gap between traditional physics-based modeling and modern machine learning approaches.

PINN in hydro-poromechanics

The implementation of PINN models for problems governed by Biot's equations in hydro-poromechanics has been investigated.

- Since the effectiveness of neural networks in representing physical processes depends on hyper-parameter selection, we performed an extensive experimentation to analyze effective architectures for accurate PINN training in poromechanics, revealing the most influential hyper-parameters. Results of the analysis include insights into significant hyper-parameters, network complexity requirements, and the robustness of the PINN approach.

- PINN implementation on coupled problems faces other challenges like high computational cost and difficulties in multi-objective loss minimization. Despite these limitations, PINNs have an advantage in allowing automatic data integration in the PDE solution stage. A sensor-driven framework was introduced to accelerate convergence and improve accuracy by automatically integrating the available field data. The sensor-driven approach demonstrated improvements in computational training time and model accuracy.

Application of the analysis to practical cases, such as limited observations, provided promising results, offering a starting point for data assimilation in real-world problems.

A comprehensive exploration of PINN applications to inverse solution in porous media was presented too.

- The study considered multiple test cases, revealing insights into the capabilities and challenges of the PINN approach. The PINN simulations demonstrated promising accuracy in estimating parameters such as Lamé constants and hydraulic conductivity in certain scenarios.
- In cases without a pumping well, the model proficiently predicted Lamé constants for the surface layer. In a homogeneous setting, hydraulic conductivity showed reasonable agreement with real values. However, challenges were observed in accurately characterizing heterogeneity, indicating the need for proper constraints.
- The introduction of a pumping well added complexity, challenging accurate characterization of the fluid dynamics. Therefore, while PINN approach tuned out to be promising in estimating certain parameters, ongoing research and refinement are essential to address challenges associated with ill-posedness and the influence of complex factors.

The findings pave the way for advancements in subsurface parameter identification, suggesting directions for improvement and future research in heterogeneous hydro-poromechanical settings.

PINN in epidemiology

In the context of epidemiology, we proposed two innovative approaches to improve the application of PINNs for the solution of SIR-based epidemiological models, and to estimate the time-dependent transmission rate, or the effective reproduction number, of an epidemic.

1. The first idea consists in splitting the training of the NNs in two steps: the first step provides the fit on the epidemiological data, while the second step minimizes the residual on the model equations. The performance of the split approach has been compared to a standard PINN application, which trains simultaneously the NNs on the joint loss function on data and residual.

2. The second idea consists in implementing a modification to the basic model equations, possibly removing the state variables that are not directly related in the disease transmission and the associated redundant terms in the loss function. This reduced PINN model has been extended to include both infection cases and hospitalization data, which are usually more reliable pieces of information.

The model has been applied on several synthetic test cases revealing that both split and joint PINN approaches accurately capture system dynamics when infectious data have small errors.

- The split approach, with preliminary training on data, showed advantages in minimizing residuals and estimating the reproduction number.
- Daily hospitalization data integration into the reduced PINN model enhanced accuracy in estimating time-varying parameters, including the effective reproduction number and the fraction of infected individuals requiring hospitalization.
- An application with real-world data, coming from the Italian COVID-19 epidemic has been presented. PINN approximations showed good results, aligning well with real-world reference data.

PINNs showcased their adaptability by incorporating epidemic surveillance data, enabling improved estimates of critical time-dependent parameters such as the effective reproduction number and the fraction of infected individuals requiring hospitalization. The split approach, involving a two-step training process, emerged as a computationally efficient alternative, outperforming the joint approach in terms of both accuracy and speed. In conclusion, the study proposes a reliable split-PINN-based approach for disease spread monitoring, providing accurate parameter estimates crucial for improved short-term forecasting.

Future research should delve into methodologies for quantifying and propagating uncertainties within the PINN-based models. This would provide a more comprehensive understanding of the reliability of model predictions and parameter estimations. The contributions of this work could be further extended and enhanced in several ways. The integration of additional data sources and more complex models to further improve the accuracy and robustness of the proposed methodologies should be explored. Furthermore, the application of PINNs in addressing real-world challenges and the incorporation of additional features into the modeling frameworks present possible directions for future investigations. The proposed techniques could be further exploited to improve the assessment of quantities of interest and effectively manage the tasks of uncertainty quantification and sensitivity analysis. Advancements in the integration of machine learning techniques with physics-based models can hold the potential to improve our understanding and predictive capabilities in the domains of hydro-poromechanics and epidemiology.

Acronyms

Adam	*Adaptive Moment Estimation*
AI	*Artificial Intelligence*
API	*Application Programming Interface*
CNN	*Convolutional Neural Network*
DL	*Deep Learning*
FD	*Finite Difference*
FE	*Finite Element*
FNN	*Feedforward Neural Network*
GD	*Gradient Descent*
GPU	*Graphics Processing Unit*
ISS	*Istituto Superiore della Sanità*
LSTM	*Long Short-Term Memory*
MAE	*Mean Absolute Error*
MCMC	*Markov Chain Monte Carlo*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
MSE	*Mean Squared Error*
NLP	*Natural Language Processing*
NN	*Neural Network*

NTK	*Neural Tangent Kernel*
ODE	*Ordinary Differential Equation*
PDE	*Partial Differential Equation*
PINN	*Physics-Informed Neural Network*
ReLU	*Rectified Linear Unit*
RNN	*Recurrent Neural Networks*
SEIRD	*Susceptible-Exposed-Infectious-Recovered-Deceased*
SIR	*Susceptible-Infectious-Recovered*
SIRD	*Susceptible-Infectious-Recovered-Deceased*
SGD	*Stochastic Gradient Descent*
TPU	*Tensor Processing Unit*

Bibliography

Albi, G., Pareschi, L., and Zanella, M. Modelling lockdown measures in epidemic outbreaks using selective socio-economic containment with uncertainty. *Mathematical Biosciences and Engineering*, 18(6):7161–7190, 2021. ISSN 1551-0018. doi: 10.3934/mbe.2021355. URL https://www.aimspress.com/article/doi/10.3934/mbe.2021355.

Almajid, M. M. and Abu-Al-Saud, M. O. Prediction of porous media fluid flow using physics informed neural networks. *Journal of Petroleum Science and Engineering*, 208:109205, 2022.

Amari, S. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, (3):299–307, 1967.

Amini, D., Haghighat, E., and Juanes, R. Physics-informed neural network solution of thermo–hydro–mechanical processes in porous media. *Journal of Engineering Mechanics*, 148, 11 2022. doi: 10.1061/(ASCE)EM.1943-7889.0002156.

Asrav, T. and Aydin, E. Physics-informed recurrent neural networks and hyper-parameter optimization for dynamic process systems. *Computers & Chemical Engineering*, 173:108195, 2023. doi: https://doi.org/10.1016/j.compchemeng.2023.108195. URL https://www.sciencedirect.com/science/article/pii/S0098135423000649.

Baydin, A. G., Pearlmutter, B. A., and Radul, A. A. Automatic differentiation in machine learning: a survey. *CoRR*, abs/1502.05767, 2015. URL http://arxiv.org/abs/1502.05767.

Bekele, Y. W. Physics-informed deep learning for flow and deformation in poroelastic media. *arXiv preprint arXiv:2010.15426*, 2020.

Bekele, Y. W. Physics-informed deep learning for one-dimensional consolidation. *Journal of Rock Mechanics and Geotechnical Engineering*, 13(2):420–430, 2021. ISSN 1674-7755. doi: https://doi.org/10.1016/j.jrmge.2020.09.005. URL https://www.sciencedirect.com/science/article/pii/S1674775520301384.

Berkhahn, S. and Ehrhardt, M. A physics-informed neural network to model COVID-19 infection and hospitalization scenarios. *Advances in Continuous and Discrete Models*, 2022(1):61, October 2022. ISSN 2731-4235. doi: 10.1186/s13662-022-03733-5. URL https://doi.org/10.1186/s13662-022-03733-5.

Berrone, S., Canuto, C., Pintore, M., and Sukumar, N. Enforcing Dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks. *Heliyon*, 9:e18820, 08 2023. doi: 10.1016/j.heliyon.2023.e18820.

Bertaglia, G., Lu, C., Pareschi, L., and Zhu, X. Asymptotic-preserving neural networks for multiscale hyperbolic models of epidemic spread. *Mathematical Models and Methods in Applied Sciences*, 32(10):1949–1985, sep 2022. doi: 10.1142/s0218202522500452.

Bertuzzo, E., Mari, L., Pasetto, D., Miccoli, S., Casagrandi, R., Gatto, M., and Rinaldo, A. The geography of COVID-19 spread in Italy and implications for the relaxation of confinement measures. *Nature Communications*, 11(1), 2020. doi: 10.1038/s41467-020-18050-2.

Biot, M. A. General Theory of Three-Dimensional Consolidation. *Journal of Applied Physics*, 12:155–164, 1941. doi: https://doi.org/10.1063/1.1712886.

Bose, N. K. and Liang, P. *Neural Network Fundamentals with Graphs, Algorithms, and Applications*. McGraw-Hill, Inc., USA, 1996. ISBN 0070066183.

Bottazzi, F. and Rossa, E. A Functional Data Analysis Approach to Surrogate Modeling in Reservoir and Geomechanics Uncertainty Quantification. *Mathematical Geosciences*, 49:1–24, 04 2017. doi: 10.1007/s11004-017-9685-y.

Box, G. E. P. and Wilson, K. B. On the Experimental Attainment of Optimum Conditions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(1):1–38, 01 1951. doi: 10.1111/j.2517-6161.1951.tb00067.x.

Brownlee, J. *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery, 2016.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6), 2021.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, pages 1–12, 2022.

Castelletto, N., Ferronato, M., Gambolati, G., Janna, C., and Teatini, P. Numerical modeling of rock/casing interaction in radioactive-marker boreholes of the Northern Adriatic basin, Italy. *SPE Reservoir Evaluation & Engineering*, 13(06):906–913, 2010.

Castelletto, N., White, J., and Tchelepi, H. Accuracy and convergence properties of the fixed-stress iterative solution of two-way coupled poromechanics. *International Journal for Numerical and Analytical Methods in Geomechanics*, 39:1593–1618, 06 2015. doi: 10.1002/nag.2400.

Chen, F., Sondak, D., Protopapas, P., Mattheakis, M., Liu, S., Agarwal, D., and Giovanni, M. D. NeuroDiffEq: A Python package for solving differential equations with neural networks. *Journal of Open Source Software*, 5(46):1931, 2020a. doi: 10.21105/joss.01931. URL https://doi.org/10.21105/joss.01931.

Chen, Y., Lu, L., Karniadakis, G. E., and Negro, L. D. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Opt. Express*, 28(8):11618–11633, Apr 2020b. doi: 10.1364/OE.384875. URL https://opg.optica.org/oe/abstract.cfm?URI=oe-28-8-11618.

Cheng, A. *Poroelasticity*. Theory and Applications of Transport in Porous Media. Springer International Publishing, 2016. ISBN 9783319252025. URL https://books.google.it/books?id=SZkFDAAAQBAJ.

Chowdhury, G. G. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, jan 2003. ISSN 0066-4200. doi: 10.1002/aris.1440370103.

Cori, A., Ferguson, N. M., Fraser, C., and Cauchemez, S. A New Framework and Software to Estimate Time-Varying Reproduction Numbers During Epidemics. *American Journal of Epidemiology*, 178(9):1505–1512, November 2013. ISSN 0002-9262. doi: 10.1093/aje/kwt133. URL https://doi.org/10.1093/aje/kwt133.

Coussy, O. *Mechanics of porous continua*. Wiley, Chichester, 1995.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, 92:88, 2022. doi: 10.1007/s10915-022-01939-z. URL https://doi.org/10.1007/s10915-022-01939-z.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Deng, L., Hinton, G., and Kingsbury, B. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8599–8603. IEEE, 2013.

Detournay, E. and Cheng, A. Fundamentals of poroelasticity. *Journal of Fluid Mechanics*, pages 113–171, 1993. ISSN 0022-1120.

Diekmann, O., Heesterbeek, J. A. P., and Metz, J. A. J. On the definition and the computation of the basic reproduction ratio R0 in models for infectious diseases in heterogeneous populations. *Journal of Mathematical Biology*, 28(4):365–382, June 1990. ISSN 1432-1416. doi: 10.1007/BF00178324. URL https://doi.org/10.1007/BF00178324.

Dourado, A. and Viana, F. A. Physics-informed neural networks for missing physics estimation in cumulative damage models: a case study in corrosion fatigue. *Journal of Computing and Information Science in Engineering*, 20(6), 2020.

EpiCentro. COVID-19 integrated surveillance: key national data, 2020. URL https://www.epicentro.iss.it/en/coronavirus/sars-cov-2-integrated-surveillance-data.

Feng, L., Chen, Z., Jr, H. A. L., Furati, K., Khaliq, A., Feng, L., Chen, Z., Jr, H. A. L., Furati, K., and Khaliq, A. Data driven time-varying SEIR-LSTM/GRU algorithms to track the spread of COVID-19. *Mathematical Biosciences and Engineering*, 19(9):8935–8962, 2022. ISSN 1551-0018. doi: 10.3934/mbe.2022415. URL http://www.aimspress.com/rticle/doi/10.3934/mbe.2022415.

Ferronato, M., Gambolati, G., Janna, C., and Teatini, P. Numerical modelling of regional faults in land subsidence prediction above gas/oil reservoirs. *International journal for numerical and analytical methods in geomechanics*, 32(6):633–657, 2008.

Ferronato, M., Castelletto, N., and Gambolati, G. A fully coupled 3-D mixed finite element model of Biot consolidation. *Journal of Computational Physics*, 229:4813–4830, 06 2010. doi: 10.1016/j.jcp.2010.03.018.

Ferronato, M., Gambolati, G., Teatini, P., Castelletto, N., and Janna, C. Ii cycle compressibility from satellite measurements. *Géotechnique*, 63:479–486, 05 2013. doi: 10.1680/geot.11.P.149.

Fokker, P., Wassing, B., Van Leijen, F., Hanssen, R., and Nieuwland, D. Application of an ensemble smoother with multiple data assimilation to the bergermeer gas field, using ps-insar. *Geomechanics for Energy and the Environment*, 5, 11 2015. doi: 10.1016/j.gete.2015.11.003.

Fraces, C. G. and Tchelepi, H. Physics Informed Deep Learning for Flow and Transport in Porous Media. 2021.

Fuks, O. and Tchelepi, H. A. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1), 2020.

Galloway, D. L. and Burbey, T. J. Review: regional land subsidence accompanying groundwater extraction. *Hydrogeology Journal*, 19(8):1459–1486, 2011.

Gambolati, G. and Freeze, R. A. Mathematical simulation of the subsidence of Venice: 1. Theory. *Water Resources Research*, 9(3):721–733, 1973.

Gambolati, G. and Teatini, P. Geomechanics of subsurface water withdrawal and injection. *Water Resources Research*, 51(6):3922–3955, 2015.

Gambolati, G., Teatini, P., Baú, D., and Ferronato, M. Importance of poroelastic coupling in dynamically active aquifers of the Po river basin, Italy. *Water resources research*, 36(9):2443–2459, 2000.

Gatto, M., Bertuzzo, E., Mari, L., Miccoli, S., Carraro, L., Casagrandi, R., and Rinaldo, A. Spread and dynamics of the COVID-19 epidemic in Italy: Effects of emergency containment measures. *Proceedings of the National Academy of Sciences*, 117(19): 10484–10491, May 2020. doi: 10.1073/pnas.2004978117. URL https://www.pnas.org/doi/10.1073/pnas.2004978117. Publisher: Proceedings of the National Academy of Sciences.

Giordano, G., Colaneri, M., Di Filippo, A., Blanchini, F., Bolzern, P., De Nicolao, G., Sacchi, P., Colaneri, P., and Bruno, R. Modeling vaccination rollouts, SARS-CoV-2 variants and the requirement for non-pharmaceutical interventions in Italy. *Nature Medicine*, 27(6):993–998, 2021. doi: 10.1038/s41591-021-01334-5. URL https://doi.org/10.1038/s41591-021-01334-5.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Gozzi, N., Chinazzi, M., Davis, J. T., Mu, K., Piontti, A. P. y., Ajelli, M., Perra, N., and Vespignani, A. Anatomy of the first six months of COVID-19 vaccination campaign in Italy. *PLOS Computational Biology*, 18(5):e1010146, 2022. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1010146. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1010146. Publisher: Public Library of Science.

Guzman, J., Babaei, M., Shi, J.-Q., Korre, A., and Durucan, S. Coupled flow-geomechanical performance assessment of co2 storage sites using the ensemble kalman filter. *Energy Procedia*, 63:3475–3482, 12 2014. doi: 10.1016/j.egypro.2014.11.376.

Guzzetta, G., Poletti, P., Ajelli, M., Trentini, F., Marziano, V., Cereda, D., Tirani, M., Diurno, G., Bodina, A., Barone, A., Crottogini, L., Gramegna, M., Melegaro, A., and Merler, S. Potential short-term outcome of an uncontrolled COVID-19 epidemic in Lombardy, Italy, February to March 2020. *Eurosurveillance*, 25 (12):2000293, March 2020. ISSN 1560-7917. doi: 10.2807/1560-7917.ES.2020.25.12.2000293. URL https://www.eurosurveillance.org/content/10.2807/1560-7917.ES.2020.25.12.2000293. Publisher: European Centre for Disease Prevention and Control.

Guéguen, Y., Dormieux, L., and Boutéca, M. Chapter 1 - fundamentals of poromechanics. In Guéguen, Y. and Boutéca, M., editors, *Mechanics of Fluid-Saturated Rocks*, volume 89 of *International Geophysics*, pages 1–54. Academic Press, 2004. doi: https://doi.org/10.1016/S0074-6142(03)80017-7. URL http://www.sciencedirect.com/science/article/pii/S0074614203800177.

Haghighat, E. and Juanes, R. SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. *Computer Methods in Applied Mechanics and Engineering*, 373:113552, 2021. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113552. URL https://www.sciencedirect.com/science/article/pii/S0045782520307374.

Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2021.113741. URL https://www.sciencedirect.com/science/article/pii/S0045782521000773.

Haghighat, E., Amini, D., and Juanes, R. Physics-informed neural network simulation of multiphase poroelasticity using stress-split sequential training. *Computer Methods in Applied Mechanics and Engineering*, 397:115141, 2022. ISSN 0045-7825. doi: https://

doi.org/10.1016/j.cma.2022.115141. URL https://www.sciencedirect.com/science/article/pii/S0045782522003152.

Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ, 1999. 2nd edition.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

He, Q., Barajas-Solano, D., Tartakovsky, G., and Tartakovsky, A. M. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources*, 141:103610, 2020.

Hennigh, O., Narasimhan, S., Nabian, M. A., Subramaniam, A., Tangsali, K., Fang, Z., Rietmann, M., Byeon, W., and Choudhry, S. NVIDIA SimNet™: An AI-Accelerated Multi-Physics Simulation Framework. In Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V. V., Dongarra, J. J., and Sloot, P. M., editors, *Computational Science – ICCS 2021*, pages 447–461, Cham, 2021. Springer International Publishing.

Hochreiter, S. Untersuchungen zu dynamischen neuronalen Netzen [Ph. D. dissertation]. *Technische Universitt München, München, Germany*, 1991.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

Ionides, E. L., Nguyen, D., Atchadé, Y., Stoev, S., and King, A. A. Inference for dynamic and latent variable models via iterated, perturbed Bayes maps. *Proceedings of the National Academy of Sciences*, 112(3):719–724, January 2015. doi: 10.1073/pnas.1410597112. URL https://www.pnas.org/doi/10.1073/pnas.1410597112. Publisher: Proceedings of the National Academy of Sciences.

Islam, M., Thakur, M. S. H., Mojumder, S., and Hasan, M. N. Extraction of material properties through multi-fidelity deep learning from molecular dynamics simulation. *Computational Materials Science*, 188:110187, 2021. ISSN 0927-0256. doi: https://doi.org/10.1016/j.commatsci.2020.110187. URL https://www.sciencedirect.com/science/article/pii/S0927025620306789.

ISTAT. Primi risultati dell'indagine di sieroprevalenza sul SARS-CoV-2. Technical report, Istituto Nazionale di Statistica, Ministero della Salute (Italia), aug 2020. URL https://www.istat.it/it/files//2020/08/ReportPrimiRisultatiIndagineSiero.pdf.

Ivakhnenko, A. G. and Lapa, V. G. Cybernetic predicting devices. 1966.

Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, Mar 2020. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.109136. URL http://dx.doi.org/10.1016/j.jcp.2019.109136.

Jahandideh, A., Hakim-Elahi, S., and Jafarpour, B. Inference of rock flow and mechanical properties from injection-induced microseismic events during geologic co2 storage. *International Journal of Greenhouse Gas Control*, 105:103206, 2021. ISSN 1750-5836. doi: https://doi.org/10.1016/j.ijggc.2020.103206. URL https://www.sciencedirect.com/science/article/pii/S1750583620306319.

Kadeethum, T., Jørgensen, T. M., and Hamidreza, H. M. N. Physics-informed neural networks for solving nonlinear diffusivity and Biot's equations. *PLOS ONE*, 15(5):1–28, 05 2020a. doi: 10.1371/journal.pone.0232683. URL https://doi.org/10.1371/journal.pone.0232683.

Kadeethum, T., Jørgensen, T., and Nick, H. Physics-informed Neural Networks for Solving Inverse Problems of Nonlinear Biot's Equations: Batch Training. In *Proceedings of 54th ARMA US Rock Mechanics / Geomechanics Symposium*. American Rock Mechanics Association (ARMA), 2020b.

Karniadakis, G., Y-Kevrekidis, L-Lu, Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nat Rev Phys*, pages 1–19, 05 2021. doi: 10.1038/s42254-021-00314-5.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Kissas, G., Yang, Y., Hwuang, E., Witschey, W. R., Detre, J. A., and Perdikaris, P. 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.

Koryagin, A., Khudorozkov, R., and Tsimfer, S. PyDEns: a Python Framework for Solving Differential Equations with Neural Networks. 2019.

Lemaitre, J., Pasetto, D., Zanon, M., Bertuzzo, E., Mari, L., Miccoli, S., Casagrandi, R., Gatto, M., and Rinaldo, A. Optimal control of the spatial allocation of COVID-19 vaccines: Italy as a case study. *PLoS Computational Biology*, 18(7), 2022. doi: 10.1371/journal.pcbi.1010237.

Lewis, R. W. and Schrefler, B. A. *The Finite Element Method in the Static and Dynamic Deformation and Consolidation of Porous Media*. Wiley, Chichester, UK, 2nd edition, 1998.

Li, W., Bazant, M. Z., and Zhu, J. A physics-guided neural network framework for elastic plates: Comparison of governing equations-based and energy-based approaches. *Computer Methods in Applied Mechanics and Engineering*, 383:113933, 2021. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2021.113933. URL https://www.sciencedirect.com/science/article/pii/S004578252100270X.

Long, J., Khaliq, A. Q. M., and Furati, K. M. Identification and prediction of time-varying parameters of COVID-19 model: a data-driven deep learning approach. *International Journal of Computer Mathematics*, 98(8):1617–1632, August 2021. ISSN 0020-7160. doi: 10.1080/00207160.2021.1929942. URL https://doi.org/10.1080/00207160.2021.1929942. Publisher: Taylor & Francis_eprint: https://doi.org/10.1080/00207160.2021.1929942.

Lou, Q., Meng, X., and Karniadakis, G. E. Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation. *Journal of Computational Physics*, 447:110676, 2021.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. DeepXDE: A Deep Learning Library for Solving Differential Equations. *SIAM Review*, 63(1):208–228, Jan 2021. ISSN 1095-7200. doi: 10.1137/19m1274067. URL http://dx.doi.org/10.1137/19M1274067.

Malinzi, J., Gwebu, S., and Motsa, S. Determining COVID-19 Dynamics Using Physics Informed Neural Networks. *Axioms*, 11(3):121, March 2022. ISSN 2075-1680. doi: 10.3390/axioms11030121. URL https://www.mdpi.com/2075-1680/11/3/121. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

Mantica, S., Nguyen, S.-K., G, V., Musso, G., M, B., and F, G. Implementation of an elasto-viscoplastic constitutive law in Abaqus/Standard for an improved characterization of rock materials. In *Science in the age of experience*, Boston, 05 2016.

Marca, R. D., Loy, N., and Tosin, A. An SIR–like kinetic model tracking individuals' viral load. *Networks and Heterogeneous Media*, 17(3):467–494, 2022. ISSN 1556-1801. doi: 10.3934/nhm.2022017. URL https://www.aimsciences.org/article/id/62450e5d2d80b73510fdd1b9.

Mari, L., Casagrandi, R., Bertuzzo, E., Pasetto, D., Miccoli, S., Rinaldo, A., and Gatto, M. The epidemicity index of recurrent SARS-CoV-2 infections. *Nature Communications*, 12(1), 2021. doi: 10.1038/s41467-021-22878-7.

Marziano, V., Guzzetta, G., Mammone, A., Riccardo, F., Poletti, P., Trentini, F., Manica, M., Siddu, A., Bella, A., Stefanelli, P., Pezzotti, P., Ajelli, M., Brusaferro, S., Rezza, G., and Merler, S. The effect of COVID-19 vaccination in Italy and perspectives for living with the virus. *Nature Communications*, 12(1):7272, December 2021a. ISSN 2041-1723. doi: 10.1038/s41467-021-27532-w.

Marziano, V., Guzzetta, G., Rondinone, B. M., Boccuni, F., Riccardo, F., Bella, A., Poletti, P., Trentini, F., Pezzotti, P., Brusaferro, S., Rezza, G., Iavicoli, S., Ajelli, M., and Merler,

S. Retrospective analysis of the Italian exit strategy from COVID-19 lockdown. *Proceedings of the National Academy of Sciences*, 118(4):e2019617118, January 2021b. doi: 10.1073/pnas.2019617118. URL https://www.pnas.org/doi/10.1073/pnas.2019617118. Publisher: Proceedings of the National Academy of Sciences.

Mathews, A., Francisquez, M., Hughes, J. W., Hatch, D. R., Zhu, B., and Rogers, B. N. Uncovering turbulent plasma dynamics via deep learning from partial observations. *Physical Review. E*, 104(2), 8 2021. doi: 10.1103/physreve.104.025205.

McCulloch, W. S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

Millevoi, C., Spiezia, N., and Ferronato, M. On Physics-Informed Neural Networks Architecture for Coupled Hydro-Poromechanical Problems. *Submitted to Journal of Computational Physics*, 2021. doi: 10.2139/ssrn.4074416. URL https://ssrn.com/abstract=4074416.

Millevoi, C., Pasetto, D., and Ferronato, M. A Physics-Informed Neural Network approach for compartmental epidemiological models. *Submitted to PLOS Computational Biology*, 2023a. doi: 10.48550/arXiv.2311.09944. URL https://doi.org/10.48550/arXiv.2311.09944.

Millevoi, C., Spiezia, N., and Ferronato, M. How to deal with heterogeneous domains in porous media with PINNs. *Submitted to Engineering with Computers*, 2023b.

Mitchell, T. *Machine Learning*. McGraw-Hill Education, New York, NY, 1997. ISBN 9780070428072.

Mow, V. C., Kuei, S. C., Lai, W. M., and Armstrong, C. G. Biphasic Creep and Stress Relaxation of Articular Cartilage in Compression: Theory and Experiments. *Journal of Biomechanical Engineering*, 102(1):73–84, 02 1980. ISSN 0148-0731. doi: 10.1115/1.3138202. URL https://doi.org/10.1115/1.3138202.

Murphy, K. P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020.

Ning, X., Guan, J., Li, X.-A., Wei, Y., and Chen, F. Physics-Informed Neural Networks Integrating Compartmental Model for Analyzing COVID-19 Transmission Dynamics. *Viruses*, 15(8):1749, August 2023. ISSN 1999-4915. doi: 10.3390/v15081749. URL https://www.mdpi.com/1999-4915/15/8/1749. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.

Ogilvy, K. W. and G., M. A. A contribution to the mathematical theory of epidemics. In *Proceedings of the Royal Society of London*, volume 115 of *A*, pages 1–19, 1927. doi: 10.1038/s42254-021-00314-5. URL http://doi.org/10.1098/rspa.1927.0118.

Olumoyin, K. D., Khaliq, A. Q. M., and Furati, K. M. Data-Driven Deep-Learning Algorithm for Asymptomatic COVID-19 Model with Varying Mitigation Measures and Transmission Rate. *Epidemiologia*, 2(4):471–489, December 2021. ISSN 2673-3986. doi: 10.3390/epidemiologia2040033. URL https://www.mdpi.com/2673-3986/2/4/33. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

Parolini, N., Dede', L., Ardenghi, G., and Quarteroni, A. Modelling the COVID-19 epidemic and the vaccination campaign in Italy by the SUIHTER model. *Infectious Disease Modelling*, 7(2):45–63, June 2022. ISSN 2468-0427. doi: 10.1016/j.idm.2022.03.002. URL https://www.sciencedirect.com/science/article/pii/S2468042722000100.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

Pasetto, D., Finger, F., Rinaldo, A., and Bertuzzo, E. Real-time projections of cholera outbreaks through data assimilation and rainfall forecasting. *Advances in Water Resources*, 108:345–356, 2017. doi: 10.1016/j.advwatres.2016.10.004.

Pasetto, D., Lemaitre, J., Bertuzzo, E., Gatto, M., and Rinaldo, A. Range of reproduction number estimates for COVID-19 spread. *Biochemical and Biophysical Research Communications*, 538:253–258, 2021. doi: 10.1016/j.bbrc.2020.12.003.

Pomerat, J., Segev, A., and Datta, R. On neural network activation functions and optimizers in relation to polynomial regression. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 6183–6185. IEEE, 2019.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. 2017a.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. 2017b.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Reddi, S. J., Kale, S., and Kumar, S. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryQu7f-RZ.

Reichstein, M., Camps-Valls, G., Stevens, B., and et al. Deep learning and process understanding for data-driven Earth system science. *Nature*, 566:195–204, 2019. doi: https://doi.org/10.1038/s41586-019-0912-1.

Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115:211–252, 2015. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

Russel, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, Harlow, UK, 4th edition, 2022. ISBN 9781292401133.

Sahli Costabal, F., Yang, Y., Perdikaris, P., Hurtado, D. E., and Kuhl, E. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42, 2020.

Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

Schiassi, E., De Florio, M., D'Ambrosio, A., Mortari, D., and Furfaro, R. Physics-Informed Neural Networks and Functional Interpolation for Data-Driven Parameters Discovery of Epidemiological Compartmental Models. *Mathematics*, 9(17): 2069, August 2021. ISSN 2227-7390. doi: 10.3390/math9172069. URL https://www.mdpi.com/2227-7390/9/17/2069.

Sejnowski, T. *The Deep Learning Revolution*. MIT Press, 2018. ISBN 9780262038034. URL https://books.google.it/books?id=9xZxDwAAQBAJ.

Shin, Y., Darbon, J., and Karniadakis, G. E. On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs. *Communications in Computational Physics*, 28(5):2042–2074, Jun 2020. ISSN 1991-7120. doi: 10.4208/cicp.oa-2020-0193. URL http://dx.doi.org/10.4208/cicp.OA-2020-0193.

Smith, J. D., Ross, Z. E., Azizzadenesheli, K., and Muir, J. B. HypoSVI: Hypocentre inversion with Stein variational inference and physics informed neural networks. *Geophysical Journal International*, 228(1):698–710, 08 2021. ISSN 0956-540X. doi: 10.1093/gji/ggab309. URL https://doi.org/10.1093/gji/ggab309.

Song, C., Alkhalifah, T., and Waheed, U. B. Solving the frequency-domain acoustic VTI wave equation using physics-informed neural networks. *Geophysical Journal International*, 225(2):846–859, 2021.

Stielow, T. and Scheel, S. Reconstruction of nanoscale particles from single-shot wide-angle free-electron-laser diffraction patterns with physics-informed neural networks. *Phys. Rev. E*, 103:053312, May 2021. doi: 10.1103/PhysRevE.103.053312. URL https://link.aps.org/doi/10.1103/PhysRevE.103.053312.

Sukumar, N. and Srivastava, A. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, February 2022. ISSN 0045-7825. doi: 10.1016/j.cma.2021.114333. URL http://dx.doi.org/10.1016/j.cma.2021.114333.

Sun, L., Gao, H., Pan, S., and Wang, J.-X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in*

Applied Mechanics and Engineering, 361:112732, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2019.112732. URL https://www.sciencedirect.com/science/article/pii/S004578251930622X.

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

Terzaghi, K. *Erdbaumechanik auf bodenphysikalischer grundlage*. F. Deuticke, Leipzig u. Wien, 1925.

Torku, T. K., Khaliq, A. Q. M., and Furati, K. M. Deep-Data-Driven Neural Networks for COVID-19 Vaccine Efficacy. *Epidemiologia*, 2(4):564–586, December 2021. ISSN 2673-3986. doi: 10.3390/epidemiologia2040039. URL https://www.mdpi.com/2673-3986/2/4/39. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

Trevisin, C., Bertuzzo, E., Pasetto, D., Mari, L., Miccoli, S., Casagrandi, R., Gatto, M., and Rinaldo, A. Spatially explicit effective reproduction numbers from incidence and mobility data. *Proceedings of the National Academy of Sciences of the United States of America*, 120(20):e2219816120, May 2023. ISSN 1091-6490. doi: 10.1073/pnas.2219816120.

Tseng, Y.-H., Lin, T.-S., Hu, W.-F., and Lai, M.-C. A cusp-capturing PINN for elliptic interface problems. *Journal of Computational Physics*, 491:112359, 2023. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2023.112359. URL https://www.sciencedirect.com/science/article/pii/S0021999123004540.

van den Driessche, P. and Watmough, J. Reproduction numbers and sub-threshold endemic equilibria for compartmental models of disease transmission. *Mathematical Biosciences*, 180(1):29–48, November 2002. ISSN 0025-5564. doi: 10.1016/S0025-5564(02)00108-6. URL https://www.sciencedirect.com/science/article/pii/S0025556402001086.

Verruijt, A. Elastic storage of aquifers. *Flow through porous media*, pages 331–376, 1969.

Verruijt, A. Numerical and analytical solutions of poroelastic problems. *Geotechnical Research*, 5(1):39–50, 2018. doi: 10.1680/jgere.15.00006. URL https://doi.org/10.1680/jgere.15.00006.

Wang, H. F. *Theory of Linear Poroelasticity with Applications to Geomechanics and Hydrogeology*. Princeton University Press, 2000. URL http://www.jstor.org/stable/j.ctt1jktrr4.

Wang, K., Chen, Y., Mehana, M., Lubbers, N., Bennett, K. C., Kang, Q., Viswanathan, H. S., and Germann, T. C. A physics-informed and hierarchically regularized data-driven model for predicting fluid flow through porous media. *Journal of Computational Physics*, 443:110526, 2021a.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient pathologies in physics-informed neural networks. 2020.

Wang, S., Yu, X., and Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, page 110768, 2021b. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110768. URL https://www.sciencedirect.com/science/article/pii/S002199912100663X.

Wang, S. J. and Hsu, K. C. The application of the first-order second-moment method to analyze poroelastic problems in heterogeneous porous media. *Journal of Hydrology*, 369(1):209–221, 2009. doi: https://doi.org/10.1016/j.jhydrol.2009.02.049.

Wiecha, P. R., Arbouet, A., Girard, C., and Muskens, O. L. Deep learning in nano-photonics: inverse design and beyond. *Photon. Res.*, 9(5):B182–B200, May 2021. doi: 10.1364/PRJ.415960. URL https://opg.optica.org/prj/abstract.cfm?URI=prj-9-5-B182.

Wilschut, F., Peters, E., Thienen-Visser, K., Fokker, P., and Hooff, P. Joint History Matching of Well Data and Surface Subsidence Observations Using the Ensemble Kalman Filter: A Field Study. *Proceedings of SPE Reservoir Simulation Symposium*, 2, 02 2011. doi: 10.2118/141690-MS.

Xiao, H., Wu, J.-L., Laizet, S., and Duan, L. Flows over periodic hills of parameterized geometries: A dataset for data-driven turbulence modeling from direct simulations. *Computers & Fluids*, 200:104431, 2020. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2020.104431. URL https://www.sciencedirect.com/science/article/pii/S0045793020300074.

Yang, M. and Foster, J. T. hp-Variational Physics-Informed Neural Networks for Nonlinear Two-Phase Transport in Porous Media. *Journal of Machine Learning for Modeling and Computing*, 2(2), 2021.

Yang, X., Zafar, S., Wang, J.-X., and Xiao, H. Predictive large-eddy-simulation wall modeling via hysics-informed neural networks. *Physical Review Fluids*, 4(3):034602, 2019.

Yin, M., Zheng, X., Humphrey, J. D., and Karniadakis, G. E. Non-invasive inference of thrombus material properties with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 375:113603, 2021.

Young, T., Hazarika, D., Poria, S., and Cambria, E. Recent Trends in Deep Learning Based Natural Language Processing. *CoRR*, 2017. URL http://arxiv.org/abs/1708.02709.

Yucesan, Y. A. and Viana, F. A. Hybrid physics-informed neural networks for main bearing fatigue prognosis with visual grease inspection. *Computers in Industry*, 125:103386, 2021. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2020.103386. URL https://www.sciencedirect.com/science/article/pii/S0166361520306205.

Zhang, D., Lu, L., Guo, L., and Karniadakis, G. E. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

Zhang, Z. A physics-informed deep convolutional neural network for simulating and predicting transient Darcy flows in heterogeneous reservoirs without labeled data. *Journal of Petroleum Science and Engineering*, page 110179, 2022.

Zhu, Q., Liu, Z., and Yan, J. Machine learning for metal additive manufacturing: Predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Computational Mechanics*, 67(2):619–635, 2021.

Ziarelli, G., Dede', L., Parolini, N., Verani, M., and Quarteroni, A. Optimized numerical solutions of SIRDVW multiage model controlling SARS-CoV-2 vaccine roll out: An application to the Italian scenario. *Infectious Disease Modelling*, 8(3):672–703, June 2023. ISSN 2468-2152. doi: 10.1016/j.idm.2023.05.012. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10240908/.

Zoccarato, C., Baù, D., Ferronato, M., Gambolati, G., Alzraiee, A., and Teatini, P. Data assimilation of surface displacements to improve geomechanical parameters of gas storage reservoirs. *Journal of Geophysical Research: Solid Earth*, 121(3):1441–1461, 2016. doi: https://doi.org/10.1002/2015JB012090. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015JB012090.

Zoccarato, C., Ferronato, M., Franceschini, A., Janna, C., and Teatini, P. Modeling fault activation due to fluid production: Bayesian update by seismic data. *Computational Geosciences*, 23:705–722, 2019. doi: https://doi.org/10.1007/s10596-019-9815-3.