UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Eric Mochiutti

Physics-Informed Echo State Networks for Modeling Controllable Dynamical Systems

Florianópolis
2023

Eric Mochiutti

Physics-Informed Echo State Networks for Modeling Controllable Dynamical Systems

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Engenharia de Sistemas e Automação.
Orientador: Prof. Eric Aislan Antonelo, Dr.
Co-supervisor:: Prof. Eduardo Camponogara, Dr.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Mochiutti, Eric
Physics-Informed Echo State Networks for Modeling Controllable Dynamical Systems / Eric Mochiutti ; orientador, Eric Aislan Antonelo, coorientador, Eduardo Camponogara, 2024.
85 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2024.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Inteligência Artificial. 3. Redes Neurais. 4. Redes de Estado de Eco. 5. Identificação de Sistemas. I. Antonelo, Eric Aislan. II. Camponogara, Eduardo. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Eric Mochiutti

Physics-Informed Echo State Networks for Modeling Controllable Dynamical Systems

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Levy Boccato, Dr.
Departamento de Engenharia de Computação e Automação
Universidade Estadual de Campinas

Prof. Gustavo Artur de Andrade, Dr.
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Engenharia de Sistemas e Automação.

ICP-Edu
Documento assinado digitalmente
Julio Elias Normey Rico
Data: 24/01/2024 21:17:10-0300
CPF: ***.840.859-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Julio Elias Normey Rico, Dr.
Coordenador do Programa

ICP-Edu
Documento assinado digitalmente
Eric Aislan Antonelo
Data: 21/01/2024 22:43:44-0300
CPF: ***.163.409-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Eric Aislan Antonelo, Dr.
Orientador

ICP-Edu
Documento assinado digitalmente
Eduardo Camponogara
Data: 24/01/2024 14:50:55-0300
CPF: ***.988.180-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Eduardo Camponogara, Dr.
Co-supervisor:
Universidade Federal de Santa Catarina – UFSC

Florianópolis, 2023.

Dedico este trabalho à minha mãe Marilene, que me apoiou incondicionalmente durante essa jornada, e ao meu pai Silas (in memoriam), que sempre incentivou os meus estudos.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Eric Aislan Antonelo, and co-advisor, Eduardo Camponogara, for their support and guidance during my research. I'm especially grateful for the opportunity they provided for the development of this research.

Thanks to my mom and dad, whose encouragement and inspiration have been a constant source of motivation. My brother, my girlfriend, dear friends, and everyone who supported me during this research journey, I am also deeply grateful for their patience and understanding during the moments of my absence. I am grateful for your invaluable help that was crucial to my success.

I would like to express my gratitude to the Human Resources Program of the National Agency of Petroleum, Natural Gas, and Biofuels (PRH-ANP) for financial support. Additionally, I extend my thanks to Petrobras, and FEESC for funding the research that made this work possible.

“Even the smallest person can change the course of the future.”
(J.R.R. Tolkien)

RESUMO

Redes Neurais com Estados de Eco (ESNs) são redes neurais recorrentes em que apenas a camada de saída linear instantânea é treinada puramente a partir de dados. Portanto, proporcionam facilidade no treinamento para modelar sistemas dinâmicos não lineares, o que pode ser usado em aplicações de controle, por exemplo. Ao incorporar leis físicas no treinamento das ESNs, as Redes de Estados de Eco Informadas por Física (PI-ESNs) foram propostas na literatura, inicialmente para modelar sistemas dinâmicos caóticos sem entradas externas. Isso resulta em uma rede que requer menos dados para treinamento, ao ser regularizada pelas informações das Equações Diferenciais Ordinárias (ODEs) durante o treinamento. Nesta dissertação, é proposta uma PI-ESN com entradas externas, que pode ser usada para modelar sistemas dinâmicos não lineares controláveis. Adicionalmente, um método autoadaptativo que equilibra as funções de custo da literatura é implementado e integrado à PI-ESN proposta (PI-ESN-a), servindo para balancear as contribuições do termo de regressão residual e do termo de perda informada por física. O desempenho dessa arquitetura foi avaliado em dois sistemas não lineares modelados por ODEs: o oscilador de Van der Pol, conhecido por seu comportamento auto-oscilatório, e o sistema de quatro tanques, comumente usado em testes de sistemas de controle multivariável. Também foi conduzida uma análise para um sistema modelado por Equações Diferenciais Algébricas (DAE), representando uma Bomba Elétrica Submersível (ESP) usada para elevação artificial em poços de petróleo e gás. Uma análise comparativa entre a PI-ESN-a e uma ESN convencional revelou que o erro de previsão diminui para a primeira, especialmente em cenários com disponibilidade limitada de dados. Além disso, constatou-se que o tempo de execução da PI-ESN é ordens de magnitude menor do que os métodos numéricos tradicionais para resolver DAEs.

Palavras-chave: Aprendizado de Máquina. Redes Neurais. Redes de Estados de Eco. Aprendizado de Máquina Informada pela Física. Identificação de Sistemas.

ABSTRACT

Echo State Networks (ESNs) are recurrent neural networks where only a linear instantaneous readout output layer is trained purely from data. Thus, they provide ease of training for modeling nonlinear dynamical systems, which can be used for control applications, for instance. By incorporating physical laws into ESN's training, Physics-Informed Echo State Networks (PI-ESNs) were proposed in the literature, initially for modeling chaotic dynamic systems without external inputs. This results in a network which needs less data to train, as it is regularized by the information from Ordinary Differential Equations (ODEs) during its training. In this dissertation, a PI-ESN with external inputs is proposed, which can be used to model controllable nonlinear dynamical systems. In addition, a self-adaptive weighting loss method from the literature is implemented and integrated into the proposed PI-ESN, serving to balance the contributions of the residual regression term and the physics-informed loss term (PI-ESN-a). The performance of this architecture was evaluated for two nonlinear systems modeled by ODEs: the Van der Pol oscillator, known for its self-oscillatory behavior, and the four-tank system, standard in multivariable control system tests. Additionally, a system modeled by Differential Algebraic Equations (DAE), representing a Submersible Electric Pump (ESP) used for artificial lift in oil and gas wells, was assessed. Comparative analysis between the proposed PI-ESN and a conventional ESN showed that the prediction error decreases for the former, especially in scenarios with limited data availability. Furthermore, the runtime of the PI-ESN was found to be orders of magnitude lower than traditional numerical methods for solving DAEs.

Keywords: Machine Learning. Neural Networks. Echo States Network. Physics Informed Machine Learning. System Identification.

RESUMO EXPANDIDO

Redes Neurais de Estados de Eco (ESNs) são redes neurais recorrentes, na qual a camada de saída é treinada exclusivamente com dados, sendo essa arquitetura usada para modelagem de sistemas dinâmicos não lineares. Esse modelo pode ser aplicado em sistemas de controle, proporcionando uma abordagem eficaz para lidar com complexidades e não linearidades. As Redes de Estados de Eco Informadas por Física (PI-ESNs) foram propostas para incorporar leis físicas no treinamento, inicialmente para sistemas caóticos sem entradas externas, reduzindo a quantidade de dados ao serem regularizadas por Equações Diferenciais Ordinárias (ODEs) que regem a física do sistema. Neste estudo, uma PI-ESN com entradas externas é proposta para modelar sistemas dinâmicos não lineares controláveis. Introduz-se um método autoadaptativo (PI-ESN-a) para balancear as funções de custo das PI-ESNs, buscando equilibrar as contribuições do termo de regressão residual e do termo de perda informada por física. O desempenho é avaliada em sistemas não lineares modelados por ODEs, como o oscilador de Van der Pol e o sistema de quatro tanques, e para um sistema modelado por Equações Diferenciais Algébricas (DAE), representando uma Bomba Elétrica Submersível (ESP) para elevação artificial em poços de petróleo e gás.

O capítulo 1 apresenta a motivação, objetivos e contribuições da dissertação, destacando o uso de técnicas de aprendizado de máquina informadas por física para modelagem de sistemas dinâmicos controláveis. O capítulo 2 realiza uma revisão da literatura relevante, explorando redes neurais usadas para sistemas de controle e Redes de Estados de Eco Informadas por Física usadas para modelagem de sistemas caóticos que serviram como base para o desenvolvimento da pesquisa. O capítulo 3 fornece uma base teórica, incluindo os conceitos fundamentais de sistemas dinâmicos, identificação de sistemas, e redes neurais artificiais.

O capítulo 4 explora a arquitetura das Redes de Estados de Eco, abordando os pesos, parâmetros da rede, otimização e a integração das leis físicas na função de custo. Nesse capítulo também é discutido a implementação da PI-ESN-a, destacando o balanceamento dinâmico entre a função custo de dados e a função de custo física para melhor desempenho em modelar sistemas físicos.

O capítulo 5 descreve as experiências conduzidas usando a ESN convencional, PI-ESN e PI-ESN-a em três sistemas diferentes: oscilador de Van der Pol, sistema de quatro tanques e uma Bomba Elétrica Submersível. A análise comparativa entre a PI-ESN-a e a ESN convencional mostra uma redução significativa no erro de previsão, principalmente em contextos com dados limitados disponíveis. Além disso, destaca-se uma notável diminuição no tempo de execução quando comparada a métodos numéricos tradicionais para a resolução de Equações Diferenciais Algébricas (DAEs).

O capítulo 6 apresenta a conclusão da dissertação, destacando os resultados obtidos, destacando a superioridade da PI-ESN-a em cenários com disponibilidade limitada de dados, e ressalta a eficiência computacional da abordagem em comparação com métodos numéricos tradicionais para resolver DAEs.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

Physical laws expressed through Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDEs) are fundamental in industry, enabling the modeling and understanding of complex phenomena. They are essential for optimizing processes, conserving resources, and enhancing industrial efficiency. These equations translate natural laws into mathematical formulations for understanding their behavior under varying conditions that aid in predicting and controlling complex industrial systems (Zheng; Wu, 2023).

The simulation of real-world systems employ computer models to predict the states of a dynamical system. This technique has a wide range of practical applications and is crucial for making informed and efficient decisions. Typically, these systems are solved numerically, but simulating them using traditional approaches can be computationally expensive, especially if the system is complex and has many states.

Furthermore, physical models do not always contain all the information about the phenomenon, often involving various approximations that can deviate from the real scenario. In this context, the use of a data-driven approach can be employed as proxy models of the phenomenological models to accelerate the simulation (Brunton; Kutz, 2022). Among the data-driven approaches, one prominent method involves the utilization of neural networks for system identification. These approaches depend significantly on the availability of a substantial dataset to effectively train the neural network in learning the desired physical processes. An approach known as physics-informed machine learning proposes using physical knowledge to assist in the training of the neural network, integrating insights from historical data with a priori knowledge available in the form of ODEs and PDEs, which approximate the system (Edwards, 2022).

Recent approaches that use deep learning to achieve this combination of data and physics are called Physics-Informed Neural Networks (PINNs) (Raissi; Perdikaris; Karniadakis, George E, 2019; Karniadakis, George Em et al., 2021). PINNs employ a loss function with two components: data loss and physics loss. While the first term corresponds to the empirical error given the training samples, the latter expresses the adherence to physical laws and constraints. By minimizing both terms, PINNs are capable of predicting the system's behavior while significantly reducing computational costs compared to a numerical simulation of the differential equation system's (Edwards, 2022).

Integrating a physics-based understanding of the system with a data-driven methodology reduces the need for data to achieve accurate results. This integration is particularly advantageous in cases where data acquisition is limited, as often encountered in various real-world applications (small data regime). Furthermore, incorporating physics knowledge can help regularize the neural network, ensuring compliance with the laws of physics and reducing overfitting to the available data (Raissi; Perdikaris; Karniadakis, George E, 2019).

An option for predicting dynamical systems is the Recurrent Neural Networks (RNNs) architecture. RNNs offer an effective approach to model dynamic systems by leveraging their internal memory to capture temporal dependencies. An example of RNNs is the LSTM (Long Short-Term Memory) network (Hochreiter; Schmidhuber, 1997). LSTMs are trained using iterative gradient descent-like methods and feature a specialized architecture designed to address the instabilities and convergence issues frequently encountered in traditional RNNs.

Echo State Networks (ESNs) offer a compelling advantage over Long Short-Term Memory (LSTM) networks when it comes to predicting chaotic dynamical systems. While both ESNs and LSTMs fall under the category of Recurrent Neural Networks (RNNs), they differ significantly in their approaches (Jaeger, 2001).

An ESN is divided into a reservoir network and a subsequent readout output layer. The reservoir is a randomly generated RNN with random fixed weights, which is responsible to map the input signals into a high-dimensional nonlinear space with memory. The linear readout output, the sole trainable part of the ESN, maps instantaneous readings of the reservoir state into the desired output. This training is typically achieved through a least-squares analytic solution, eliminating the need for numerical least-squares solvers (Verstraeten et al., 2007).

ESNs have outperformed LSTMs due to their rapid training capabilities and their ability to achieve state-of-the-art performance in modeling chaotic dynamical systems (Shahi; Fenton; Cherry, 2022). ESNs attains this performance advantage without the need for the intricate architectural design and training complexity from LSTMs. This makes ESNs a preferable choice when dealing with time series prediction.

Due to the characteristics of the ESN architecture, hybrid methods for integrating physical information have been introduced by Doan, Polifke, and Magri (2019b), requiring only minor modifications to the neural network's structure and altering the loss function to integrate the physics information. Furthermore, ESNs are highly promising candidates for control applications, as demonstrated in the works of Waegeman, Wyffels, and Schrauwen (2012) and Jean P. Jordanou et al. (2018), as they do not require substantial alterations to their architecture, unlike traditional PINNs (Antonelo, E. A. et al., 2021).

However, the original PI-ESN has been designed for systems without external control input. Therefore, to make PI-ESNs work for modeling controlled systems described by ODEs, a control input must be added to the network. Another application of interest involves oil production platforms employing artificial lift mechanisms described by DAEs, which pose greater numerical complexity compared to systems described by ODEs.

Time series predictions are essential for optimizing the control and management of complex systems, improving operational efficiency. Oil production plants entail intricate phenomena that are difficult to be represented using models. Therefore, the modification and improvement of techniques, such as PI-ESNs, are crucial for addressing challenges within datasets of limited size, including those encountered in oil production platforms and various engineering sectors. In this context, applying PI-ESNs in scenarios with limited data avail-

ability offers a hybrid approach that combines insights from physical models with data-driven techniques to enhance prediction performance.

1.2 OBJECTIVES

The main objective of this dissertation is to extend the Physics-Informed Echo State Network (PI-ESN) by including external inputs for modeling controllable dynamical systems described by ODEs and DAEs.

The specific objectives are:

- To implement a PI-ESN capable of incorporating external input information to address dynamical systems that respond to control signals.
- To apply the PI-ESN to benchmark dynamic systems described by ODEs: the Van der Pol oscillator, and the four-tank problem.
- To apply the PI-ESN to an electric submersible pump (ESP) used for artificial lift in oil wells, described by DAEs.
- To perform experiments on the proposed PI-ESN method in different settings with limited data, comparing to its ESN conventional counterpart.

1.3 CONTRIBUTIONS

This dissertation presents the following contributions:

- A PI-ESN architecture to work with external inputs to enable the modeling of controllable dynamical systems. This feature allows the PI-ESN to naturally accept control inputs, in contrast to conventional PINNs which can only have fixed inputs or initial conditions.
- Self-adaptive weights were introduced into the PI-ESN training, dynamically balancing each term (data loss and physics-based loss) in the loss function, helping the training convergence for a class of systems.
- A comprehensive set of experiments was conducted, demonstrating the improved performance and predictive capabilities of PI-ESN over the conventional ESN in low-data scenarios for three different dynamic systems.
- An experiment that demonstrates that the execution time of PI-ESN is significantly smaller than the numerical solution of the DAE.

1.4 THESIS ORGANIZATION

The remainder of this dissertation is organized as follows:

- **RELATED WORKS**: In this chapter, the focus is on conducting a comprehensive literature review of relevant works concerning PINNs/ESN for Control and PI-ESN. This review highlights the works used as the foundation for this dissertation.
- **THEORETICAL FOUNDATIONS**: This chapter is dedicated to establishing the necessary theoretical foundations to comprehend this work. Key concepts such as system identification, mathematical models, and neural networks are addressed. This theoretical groundwork will serve as the basis for the subsequent discussions and implementations.
- **ECHO STATE NETWORKS**: This chapter provides an exploration of the Echo State Network architecture, specifically focusing on the necessary adaptations for the implementation of the PI-ESN. It delves into these adaptations, emphasizing how they can enhance ESN's performance.
- **EXPERIMENTS**: This chapter presents the experimental results focusing on the application of PI-ESN to systems with external inputs. It also shows the following: 1) the impact of training data size vs. number of collocation points for physics-informed training; 2) robustness to uncertainties in model parameters; 3) PI-ESN applied to a system described by Algebraic Differential Equations; and 4) a thorough examination of the limitations encountered in the proposed technique.
- **CONCLUSION**: This chapter concludes the dissertation, providing a comprehensive overview of the achievements and future prospects.

2 RELATED WORKS

This chapter provides a comprehensive literature review of relevant works that are correlated with this dissertation and have served as the foundation for its development. The chapter is structured as follows:

- **PINNs/Echo State Networks for Control (Section 2.1):** This section presents an exploration of neural networks applied for system control, with a focus on their applications and contributions.
- **Physics-Informed Echo State Networks (Section 2.2):** This section delves into the concept of Physics-Informed Echo State Networks, providing a comprehensive examination of the state-of-the-art and highlighting the research that serves as the foundation for this dissertation.

2.1 PINNS/ECHO STATE NETWORKS FOR CONTROL

In the architecture of PINNs, the inclusion of the time as an input to the neural network poses challenges in generalization during the test phase, especially when time exceeds the values observed in the training data. Physics-Informed Neural Nets for Control (PINC) are proposed by Eric Aislan Antonelo et al. (2021) to extend the PINNs to deal with variable longer time horizons that are not fixed in the training phase, by adding the initial state and control signal as additional input variables to the PINNs.

In multi-body dynamics, PINC have been used for Model Predictive Control (MPC). This approach streamlines controller design, reducing the need for finite difference methods. In fact, automatic differentiation is employed to compute sensitivities efficiently, leading to gradient-based algorithms for optimal control problems (Nicodemus et al., 2022).

Another viable option is the utilization of Echo State Networks (ESNs) in control problems, as this neural network architecture requires minimal adaptation to be applied in MPC-type controllers. For instance, the Practical Nonlinear Model Predictive Controller (PNMPC) partially linearizes the ESN model, eliminating the need for finite difference methods. This approach, known as PNMPC-ESN, has proven successful in controlling a gas-lifted oil well model, effectively adhering to constraints and achieving set point tracking (Jordanou, Jean P. et al., 2022). ESNs can also be employed for online control, where they were effectively applied to oil well problems (Jordanou, J.; Antonelo, E.; Camponogara, 2019).

Furthermore, ESNs find application in the control of various systems. In the case of the Twin Rotor Aero-Dynamical System, ESNs are used with robust control to estimate and mitigate disturbances, resulting in improved control quality (Czajkowski, 2014). For motor speed control for differential drive robots, ESN controllers have been shown to outperform PID controllers, demonstrating their versatility and effectiveness across different control scenarios (Salmen; Ploger, 2005).

2.2 PHYSICS-INFORMED ECHO STATE NETWORKS

The ESN has proven to be an architecture with significant potential for applying physics-informed machine learning techniques to time series prediction tasks relevant to the industry. The literature on implementing hybrid echo state networks begins with the work of Pathak et al. (2018). This work focuses on hybrid forecasting of chaotic processes, combining an approximate knowledge-based model with an ESN. Their hybrid approach demonstrates improved state prediction performance in two applications: forecasting the Lorenz system and the Kuramoto-Sivashinsky equations. Essentially, their method expands the input to the reservoir network to include predictions from the knowledge-based model.

Another method for incorporating physics into ESN's was proposed by Doan, Polifke, and Magri (2019b). In the proposed technique, the architecture of the network is minimally altered, with physics being implemented in the loss function used to train the output layer of the ESN. The proposed Physics-Informed Echo State Network (PI-ESN's) is trained to solve supervised learning tasks while ensuring adherence to physics laws. By incorporating an additional loss function based on the system's governing equations during training, non-physics predictions are penalized without requiring extra training data. The proposed approach demonstrates a significant improvement in the predictability horizon of chaotic systems (chaotic Lorenz system and Charney-DeVore system), outperforming conventional ESN's. It is worth noting that Pathak et al. (2018) do not incorporate physics laws into the training of ESNs as PI-ESNs do and requires the simulation of the model.

In the study conducted by Doan, Polifke, and Magri (2019a), PI-ESNs were employed for predicting extreme events in turbulence. This research showcases the integration of empirical and physics-based modeling methods, facilitating precise forecasts of extreme events and abrupt transitions in self-sustaining turbulence processes. Another application of PI-ESNs involved reconstructing the evolution of unmeasured states in chaotic systems, as demonstrated by Doan, Polifke, and Magri (2020). Through training the PI-ESN using data lacking information on unmeasured states alongside the physics equations of a chaotic system, accurate reconstruction of these unmeasured states was achieved. Additionally, the robustness of the reconstruction to noisy data highlights the PI-ESN's denoising capabilities, demonstrating the potential of integrating physics knowledge and machine learning to enhance the prediction and reconstruction of unmeasured states in chaotic dynamical systems. In this case, the prediction horizon of the physics error is estimated by discretizing the equations using an explicit Euler time-integration scheme.

The method proposed by Racca and Magri (2021) represents an improvement over the previous approach by utilizing automatic differentiation for physics error computation, instead of relying on an explicit Euler integration scheme. Through the application of automatic differentiation, the method achieves a more precise estimation of the inherent physical error within the chaotic system. This improved accuracy can be attributed to the meticulous treatment of gradients and derivatives, enabling a finer adjustment of model parameters during the training

process.

Another proposal by Oh (2020) introduces the concept of ESNs that rely solely on physics, acting as an approximation of ODEs. Physics-informed neural network models learn solutions under specific initial or boundary conditions, requiring retraining if those conditions change. This limitation makes them less suitable for replicating a target system's behavior across various practical scenarios. In response, an ODE approximator is designed as a straightforward solution to replicate the solution sequentially, guided by the fundamental invariance of differential equations. In this method, the input sequence is unknown and must be determined as an accurate solution. To overcome this challenge, the idea is to split the training into two stages. Initially, a test solution is used. Once an accurate approximation is achieved in the first stage, it becomes possible to prepare the input sequence for the next stage. The goal is to generate the solution in a recurrent manner, but this requires the two-stage approach for effective operation. This method was applied to the unforced Van der Pol oscillator and the Lorenz system, both of which are chaotic systems.

In Na et al. (2023), the Pi-HESN architecture (Physics-Informed Hierarchical Echo State Networks) is proposed to predict the dynamics of chaotic systems. This architecture is based on multiple reservoirs connected sequentially. These reservoirs capture the dynamics of the system by processing experimental data, using the internal state signal of each layer to collect states. Then, Pi-HESN integrates data and physical laws, similar to a regular PI-ESN, incorporating prior physical knowledge into the objective function to maintain fundamental physical principles. This combination of data-based and knowledge-based approaches in Pi-HESN enhances model generalization, reduces the need for extensive training data, and ensures the consistency of physics in the results for four classic chaotic systems.

However, the aforementioned studies focus on predictions for systems without external control input. For control purposes, it is necessary to adapt the network architecture to accept an external input. Therefore, the motivation behind this work is to modify this architecture to incorporate an external input. To achieve this, three different systems were chosen for this application: the Van der Pol Oscillator, four-tank system, and Electric Submersible Pump (ESP). The latter being a system of particular interest, given its practical use in the oil and gas industry.

3 THEORETICAL FOUNDATIONS

This chapter presents the fundamental concepts that forms the basis of this work. The chapter is organized as follows:

- **Dynamic Systems and System Identification (Section 3.1):** In this section, the fundamental principles governing dynamic systems are explored, accompanied by an explanation of the system identification process.
- **Neural Networks (Section 3.2):** This section provides a brief review of Artificial Neural Networks, focusing on the data-driven approach for system identification.

3.1 DYNAMIC SYSTEMS AND SYSTEM IDENTIFICATION

3.1.1 Dynamic Systems

Mathematical equations in dynamical systems describe the natural world, facilitating the representation of interactions among quantities. A dynamical system can be defined as a relationship between an input and an output that depends not only on the current input but also on past inputs (Chen, 2012). In essence, these systems involve the analysis, prediction, and understanding of systems defined by collections of differential equations. Real-world systems can be mathematically represented using Ordinary Differential Equations (ODE), partial Differential Equations (PDE) or Differential-Algebraic system of Equations (DAE), formulated based on established physical laws. These equations find application in various domains such as weather prediction, fluid dynamics, turbulence modeling, and nearly all systems exhibiting temporal evolution (Brunton; Kutz, 2022).

Dynamical systems can be classified as discrete versus continuous systems, linear versus nonlinear behaviors and time-variant versus time-invariant dynamics (Chen, 2012). These classifications offer direct insights into the nature of dynamic processes:

- **Continuous-time or Discrete-time**: Discrete dynamical systems involve variables that change in specific and separate steps or increments, often described by a finite difference equation. These systems are well-suited for modeling processes that evolve in discrete intervals, such as population growth or algorithm iterations. On the other hand, continuous-time dynamical systems involve variables that change continuously as time progresses, typically described by differential equations. These systems find applications in modeling continuous phenomena like fluid flow, continuous stirred-tank reactor, or electrical circuits.
- **Linear or non-linear**: For linear systems, the system's behavior can be described using linear equations that adhere to the principle of superposition, making them analytically tractable. Many engineering and physical systems can be approximated as linear within certain operating ranges. However, for nonlinear systems, the principles of proportionality and superposition are not applicable. Their behavior is more complex and can give rise to phenomena such as chaos and bifurcations. Nonlinear systems are prevalent across

a wide range of natural and engineered systems, from biological networks to intricate physical processes.

  - Non-Chaotic System: exhibit stable and predictable behaviors over time;
  - Chaotic System: Sometimes a nonlinear system has a complex steady-state behavior, deviating from equilibrium or periodic oscillation. This behavior is commonly known as chaos. Some of these chaotic motions exhibit randomness, despite the deterministic nature of the system. The chaotic system's presents high sensitivity with respect to initial conditions, resulting in unpredictable and seemingly random behaviors. This phenomenon is famously illustrated by the butterfly effect (Devaney, 2021; Khalil, 2001; Brunton; Kutz, 2022).

- **Time-variant or Time-invariant**: Time-variant systems exhibit behaviors that change over time. The system's parameters, equations, or relationships evolve with time, leading to varying dynamics. Analyzing time-variant systems often requires techniques that account for changing conditions. For control purposes, a slowly varying parameter, such as the pressure in an oil and gas reservoir, can be considered time-invariant (Jordanou, Jean Panaioti, 2019). On the other hand, time-invariant systems maintain consistent behaviors regardless of time. The system's parameters and equations remain constant, allowing for simpler analysis and prediction of long-term behaviors.

Equation (1) represents a generic dynamical system, with $x(t)$ denoting the system's state and $u(t)$ serving as an input, acting as an independent variable.

$$\dot{x}(t) = g(t,x(t),u(t);\theta), \tag{1}$$

where g represents a vector field that possibly depends on the system state $x(t)$, the time t, and a parameter set θ. States vectors, inputs vectors, and functions are presented in Equation (2):

$$\mathbf{x}(t) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{u}(t) = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}, \quad \mathbf{g}(t,x(t),u(t);\theta) = \begin{bmatrix} g_1(t,x(t),u(t);\theta) \\ g_2(t,x(t),u(t);\theta) \\ \vdots \\ g_n(t,x(t),u(t);\theta) \end{bmatrix} \tag{2}$$

When Equation (1) does not require the explicit presence of an input $\mathbf{u}$, it is referred to as the unforced state equation, which is given as follows:

$$\dot{\mathbf{x}}(t) = g(t,\mathbf{x}(t);\theta) \tag{3}$$

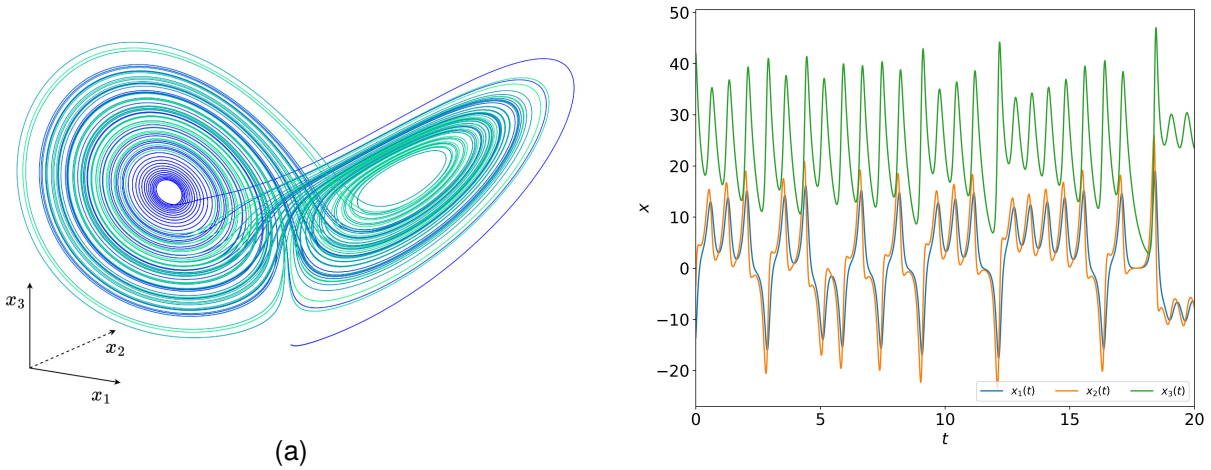
The unforced state equation does not mean that the input to the system is zero. It could be that the input has been specified as a given function of time, $u = g(t)$, a given

feedback function of the state, $u = g(x)$ (Khalil, 2001). An example of a dynamical system is the chaotic Lorenz system in Equation (4), each described by its respective set of equations (Lorenz, 1963):

$$\begin{aligned} \dot{x}_1 &= \sigma(x_2 - x_1), \\ \dot{x}_2 &= x_1(\rho - x_3) - x_2, \\ \dot{x}_3 &= x_1 x_2 - \beta x_3, \end{aligned} \tag{4}$$

where $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$. In this case, the state vector is $\mathbf{x_d} = [x_1\ x_2\ x_3]^T$ and the parameter vector is $\theta = [\sigma\ \rho\ \beta]^T$. The Lorenz system, as shown in Figure 1, stands as an example of a chaotic dynamical system that has captured considerable attention in the field of nonlinear dynamics. This recognition is attributed to its different traits, including sensitivity to initial conditions and the presence of attractors (Brunton; Kutz, 2022). In the Figure 1 (a), it is possible to observe the evolution of the Lorenz dynamical equations and their respective attractor. Meanwhile, Figure 1 (b) illustrates the temporal evolution of the dynamical states.

Figure 1 – The Lorenz Attractor



(a) (b)

Source: Author

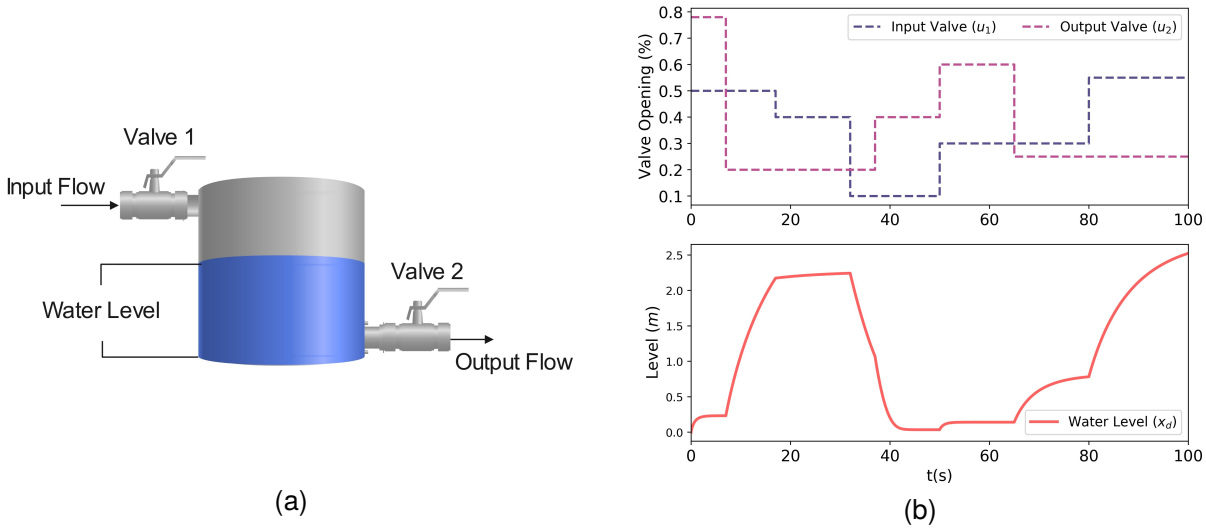
Another example of a dynamic system is a valve-controlled system designed to regulate fluid flow in response to changing conditions, as shown in Figure 2. The system involves the manipulation of valve openings, where two valves, denoted as u_1 and u_2, control the flow of fluid. The system aims to maintain a stable flow rate within a specified range by adjusting these valve openings. The dynamics of this system are described by the following differential equation:

$$A\frac{dx_d}{dt} = K_1 u_1 - K_2 u_2 \sqrt{2 g x_d}, \tag{5}$$

where A represents the cross-sectional area of the tank's water, K_1 and K_2 are constants

associated with the valves, x_d is the fluid level within the system, and g is the acceleration due to gravity. Given specific values of $K_1 = 0.05\,\text{m}^3/\text{s}$, $K_2 = 0.015\,\text{m}^3/\text{s}$, $A = 0.05\,\text{m}^2$, and $g = 9.8\,\text{m/s}^2$, the equation captures how changes in the valve openings affect the rate of fluid flow and, consequently, the fluid level within the system.

Figure 2 – Water Level System



(a) (b)

Source: Author

This valve-controlled system showcases the principles of dynamic control, where precise adjustments of valve openings enable the regulation of fluid flow and level, contributing to the stability and efficiency of industrial processes involving fluid transport and management.

The Lorenz system and the water level system demonstrate contrasting behaviors with respect to their interactions with external factors. The Lorenz system operates autonomously, driven solely by its internal state variables. It does not require any external input to evolve over time. In other words, the Lorenz system does not respond to external signals or influences. As a result, the vector $\mathbf{u}(t)$ representing external inputs is zero ($u = 0$), or, it can be said that the system's input consists of its own output values (output feedback), characterizing it as an unforced steady-state system. Its behavior is self-contained, determined solely by its initial conditions and the equations governing its dynamics.

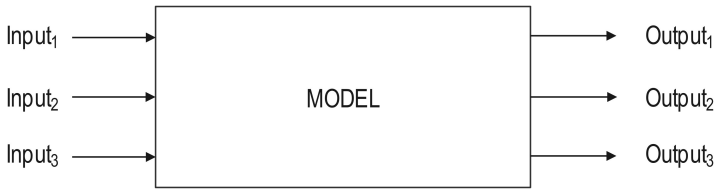
In contrast, the water level system is influenced by external factors, specifically the valve actions. Changing the valve openings affects the system's behavior by altering the flow rate and, consequently, the water level. In this case, the vector $\mathbf{u}(t)$ captures the effects of valve adjustments. Unlike the Lorenz system, the water level system responds to and is shaped by external inputs.

3.1.2 System Identification

System identification is a methodology used to develop a mathematical model of a dynamic system. In this method, a function is chosen to describe the relationship between

the system's inputs and outputs. Analyzing process characteristics and relationships between variables is crucial for predicting, controlling and monitoring systems. The process of developing a mathematical description of the studied process, known as a model, is a key step. Two main approaches are used for model development: a theoretical approach based on fundamental physics laws, and an empirical approach based on analyzing observed data from a system (Tangirala, 2014). Figure 3 illustrates the model with a mathematical representation of the relationship between input and output in a dynamic system.

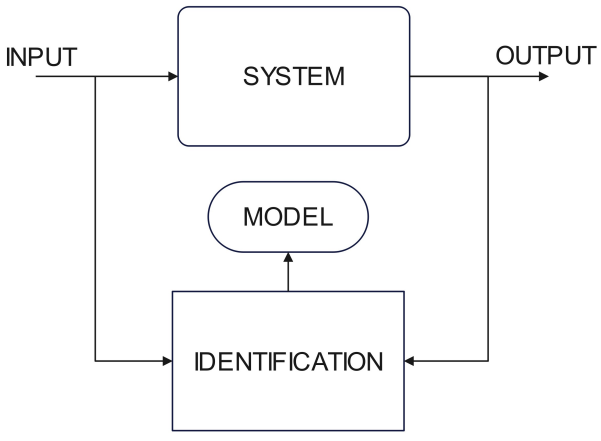
Figure 3 – System model



Source: Author

In control systems, normally inputs are variables that can be directly adjusted, just like pressing keys on a keyboard or turning the knobs on a stove. Outputs correspond to the data gathered through human observation or instruments, such as the characters displayed on a screen in response to keyboard inputs or the temperature reading obtained from the stove's display panel (Jordanou, Jean Panaioti, 2019). Figure 4 shows a system identification diagram. The objective is to develop a model from observed data.

Figure 4 – System identification



Source: Author

System identification consists of using input and output data to find a model that approximates the physics of the system. Acquiring a model for a dynamical system allows the prediction of future states in response to different inputs, which can be used later for system control. A range of algorithms and computational resources are available to conduct system modeling, and they can be categorized into three different approaches:

- **White-box model**: Models are based on the physical equations that dictate the system's behavior. It is modeled based on physics equations that fully explain the dynamics of the system.
- **Grey-box model**: Some information about the system is known, although part of the dynamics may remain unknown or be overly complex for direct modeling, requiring identification procedures.
- **Black-box model**: Absence of prior knowledge about the system or its complexity inhibits modeling. System identification with data is necessary to construct a model that mimics the operation of the real system within a specific region. Due to excessive complexity, a simpler model becomes more manageable for control and/or optimization purposes.

The fundamental aspect of various applications relies on employing models for simulations and predictions, which serve as the primary motivations for developing these models. Simulations provide a cost-effective and time-efficient alternative to experiments. Advances in computational science and technology have elevated simulations to powerful tools for comprehending physical processes.

Predictions involve using the model to anticipate process behavior under current operating conditions over a limited time interval. Predictions play a crucial role in activities such as design, control, anomaly detection, and testing novel approaches. Nevertheless, the precision requirements for models in prediction-based applications are less stringent than in simulation-based scenarios (Tangirala, 2014). However, it is important to note that there exists a distinction between a mathematical model and an actual real-life system. A model typically constitutes a simplified approximation of a system that exists in the real world.

3.1.3 Models and Simulation

First-principle models originate from fundamental principles, utilizing physics laws and relationships, resulting normally in Ordinary Differrential Equations (ODE) or Differential-Algebraic Equations (DAE) or Partial Differential Equations (PDE). Despite their effectiveness and reliability, simulations of these models require proficient numerical and algebraic solvers.

ODEs capture relations involving an independent variable, often time (t), and their respective solutions depend on initial conditions, which describe the initial state of the system in relation to the dependent variables. A system of these equations is a collection that involves dependent variables, their derivatives, and the independent variable. In a system of ODEs, all equations are differential. A generalized representation of ODEs system is presented in Equation (6):

$$\begin{aligned} \dot{x}_1 &= g_1(t,x_1,x_2,\ldots,x_n) \\ \dot{x}_2 &= g_2(t,x_1,x_2,\ldots,x_n) \\ &\vdots \\ \dot{x}_n &= g_n(t,x_1,x_2,\ldots,x_n) \end{aligned} \tag{6}$$

Alternatively, the definition of DAEs includes non-differential algebraic equations in the system. DAEs are systems of equations similar to ODEs that have additional algebraic constraints related to the dependent variable. A generic form of a DAE is expressed in Equation (7). These complexities intensify when additional algebraic terms are involved, making DAEs more challenging to solve numerically.

$$\begin{aligned} \dot{x}_1 &= g_1(x_1, x_2, t) \\ 0 &= g_2(x_1, x_2, t) \end{aligned} \tag{7}$$

PDEs consider multiple independent variables, incorporating spatial dimensions alongside time, and are widely utilized in scenarios like heat diffusion and fluid dynamics. A generic PDE is expressed in Equation (8):

$$\frac{\partial x}{\partial t} + \frac{\partial^2 x}{\partial l^2} = 0, \tag{8}$$

where l is the spatial dimensional independent variable. Solving PDEs requires specifying both the initial conditions of the system, similar to ODEs, and the boundary conditions that define the conditions at the domain boundaries where the solution is determined. Typically, these systems of equations are much more complex to solve analytically or numerically compared to ODEs.

Modeling and simulation using these equations aim to replicate real-world system evolution. Simulation provides significant advantages, sparing the costs of conducting experiments on actual processes. These simulations facilitate performance analysis over time and predict behavior under varying input conditions. The versatility of simulations empowers researchers to evaluate system behavior over time, envisage complex phenomena, and predict responses under diverse scenarios. This multifaceted approach facilitates a comprehensive understanding of dynamic systems across various domains, spanning from engineering to the natural sciences.

3.2 NEURAL NETWORKS

3.2.1 Neural Networks Architectures

Artificial Neural Networks (ANNs) have become widely popular as an extremely useful tool for solving problems involving tasks such as classification, pattern recognition, and prediction (Abiodun et al., 2018). Silva, Spatti, and Flauzino (2016) highlights that ANNs can be used in process identification and control, time series forecasting, and system optimization.

An ANNs consists of layers of basic computational units called neurons. The outputs from one layer of neurons serve as inputs for the following layers, resulting in the formation of multiple layers of neurons. Neurons serve as fundamental processing units and include activation functions denoted as $f(\cdot)$. These activation functions play a crucial role in non-linearly

mapping input signals (u) to output signals (y), as illustrated in Figure 5. Mathematically, a neuron can be expressed as:

$$y = f\left(\sum_{i=1} (w_i u_i + b)\right), \tag{9}$$

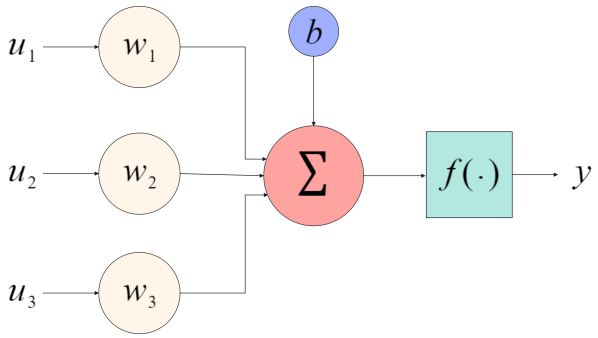
where: w_i represents the weights that adjust the connections between the inputs u_i to the output y during the forward pass; b represents the bias, a fixed value that introduces an offset to the activation function. Activation functions introduce nonlinearity into the neural network's computations. For example, hyperbolic tangent function restrict the output, a result of the weighted sum of inputs, to be within the interval from -1 to 1. These functions enhance the network's capacity to capture complex patterns (Silva; Spatti; Flauzino, 2016).

Training involves learning from errors through the calculation of gradients using the backpropagation method. This method computes derivatives across the layers of neurons in the network, facilitating the adjustment of weights and biases through an optimization algorithm such as gradient descent or Adam. It operates on the premise of assessing how alterations in weights and biases impact the objective function. The primary objective of the network is to minimize the discrepancy between predicted and actual outputs, quantified by the loss function in Equation (10) utilizing mean square error. Alternative loss functions, such as cross-entropy, can be used for training.

$$\min_{w,b} \frac{1}{N_t} \sum_{i=1}^{N_t} \left[\hat{y}_i - y_i(w_i,b)\right]^2, \tag{10}$$

where $\hat{y}_i$ is the desired value, $y_i(w_i,b)$ is the neural network output and N_t is the number of training data points.

Figure 5 – Non-linear neuron model

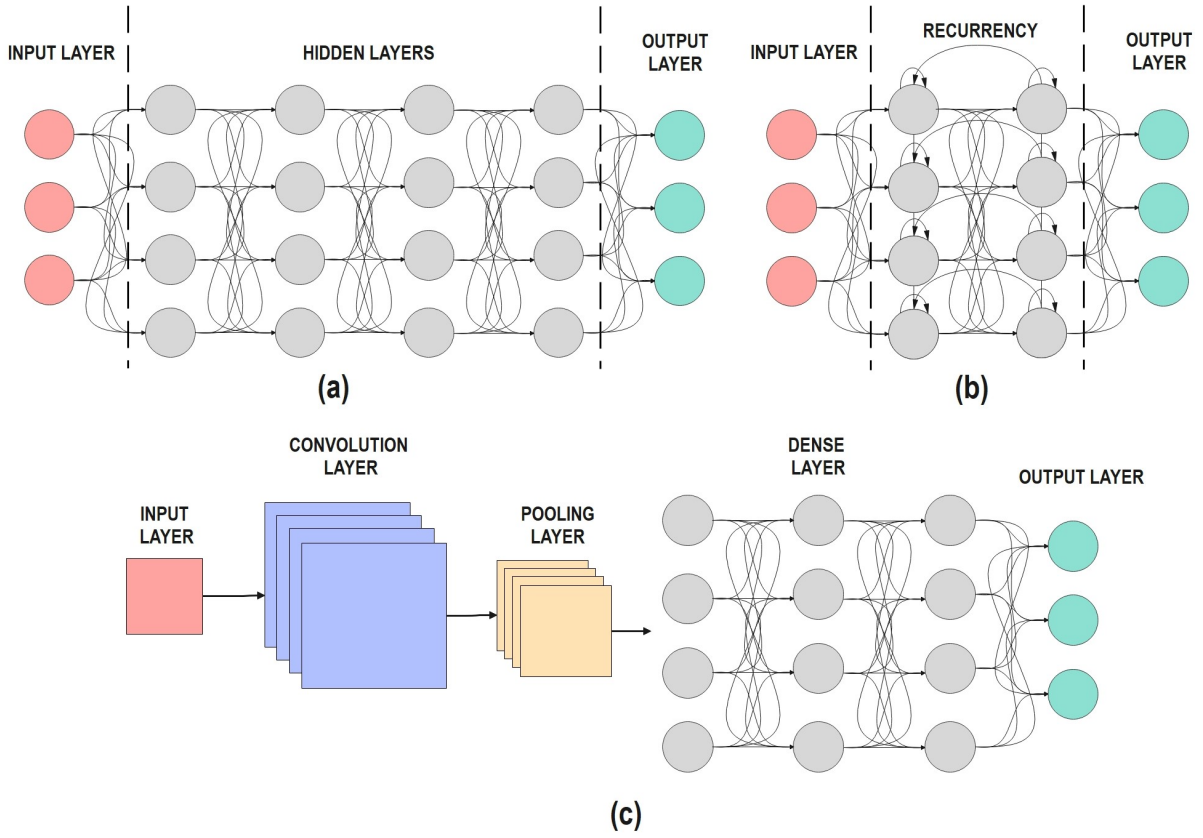


Source: Author

Moreover, neuron units can be organized into more than one layer and interconnected by a large number of connections, each associated with weights that store all the knowledge obtained by ANNs and serve to weigh the input of each neuron in the network (Braga, 2007;

Elsheikh et al., 2019). Based on this, various network architectures are built upon this multi-layered organization to solve problems, as illustrated in Figure 6.

Figure 6 – Example of Multi-layered ANN: a) DNN; b) RNN; c) CNN



Source: Author

Deep Neural Networks (DNN) consist of multiple interconnected layers and are designed for extracting complex features from structured data. Recurrent Neural Networks (RNNs) contain feedback connections within the internal structure, maintaining memory across steps to model temporal dependencies; thus, are tailored for sequential data. Convolutional Neural Networks (CNNs), on the other hand, excel in extracting features from grid-like data, such as images, by applying convolution operations. While DNNs specialize in deep feature learning, RNNs handle sequential data, and CNNs excel in grid-like data processing, these categories can also be combined in hybrid architectures to tackle more intricate problems, often seen when using CNNs and RNNs together for tasks involving sequential image analysis, such as speech recognition and image captioning.

For instance, Savalia and Emamian (2018) employed Multilayer Perceptrons (MLP) and Convolutional Neural Networks (CNN) to classify cases of cardiac arrhythmia. Schwedersky, Flesch, and Dangui (2020) used Echo State Networks (ESN), a type of recurrent neural network, for the identification of multivariable and nonlinear systems. Similarly, Yang et al. (2019) and Eric Aislan Antonelo, Camponogara, and Foss (2017) employed ESNs in

algorithms for time series predictions and for estimating the pressure of oil extraction wells, respectively.

ESNs, introduced by Jaeger (2001), are known for their faster training compared to LSTMs, another type of recurrent neural network architecture. Additionally, they exhibit superior performance in modeling chaotic dynamical systems, as demonstrated in a study by Shahi, Fenton, and Cherry (2022). They find applications across various domains, including robot localization (Antonelo, E. A.; Schrauwen; Stroobandt, 2008), navigation (Antonelo, E.; Schrauwen, 2014), IoT (Zhou et al., 2022), power systems (Roberts et al., 2022), and oil well control (Jordanou, J.; Antonelo, E.; Camponogara, 2019; Antonelo, E. A.; Camponogara; Foss, 2017). This work is focused on Echo State Networks because of their ability to predict dynamic systems and fast training compared to LSTM.

3.2.2 Physics-Informed Neural Networks

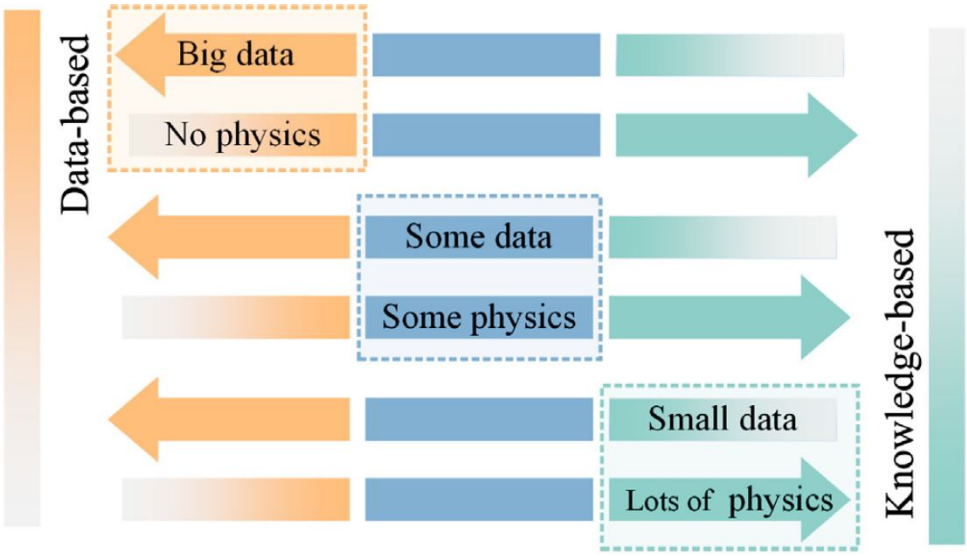
Machine learning models typically rely on a significant amount of data to achieve accurate predictions. However, this amount of information is not always available when dealing with real-world systems, often because measuring certain quantities can be very difficult or even impossible. Even in cases where there is a substantial amount of data, machine learning models can exhibit physically inconsistent predictions due to extrapolation and generalization performed by the neural network. As a result, models have been proposed where the architecture of the neural network incorporates the physics laws governing the system. This approach aims to merge the known physics with the available data (Karniadakis, George Em et al., 2021).

Figure 7 illustrates three different categories of physics problems and their corresponding available data. In the small data regime, it is assumed that a comprehensive understanding of all physical aspects is available, allowing for predictions with a high level of precision through mathematical modeling of the system's physics, with data provided for both initial and boundary conditions. In real-world applications, the most prevalent scenario falls within the middle category, where partial knowledge of both data and physics exists. In the realm of big data, one might assume that the quantity of data available is sufficient to fully describe the physics of the system. However, it is understood that due to the generalization tendencies of neural networks, deviations can still occur, potentially leading to violations of certain physical laws (Na et al., 2023).

Physics-Informed machine learning operate at the interface of physics and neural networks, providing a solution that connects data with physical laws, where prior knowledge underlying empirical, observational, physics, or mathematical understanding is harnessed to enhance learning algorithm performance (Karniadakis, George Em et al., 2021).

The work by Raissi, Perdikaris, and George E Karniadakis (2019) introduced the concept of Physics-Informed Neural Networks (PINNs), which involves training deep neural networks in a supervised manner to adhere to physical laws described by PDEs. This ap-

Figure 7 – Data-based and knowledge-based scenarios



Source: Na et al. (2023)

proach empowers the automatic discovery of data-driven solutions for PDEs or ODEs, where the underlying physics function can be represented by an ODE as shown in Equation (11). These solutions are subsequently employed to approximate the states of dynamic systems governed by ODEs.

$$\mathcal{F}(\mathbf{y}) \equiv \partial_t \mathbf{y} - \mathcal{N}(\mathbf{y}), \tag{11}$$

where: $\mathcal{F}$ represents a general non-linear operator, ∂_t is the time derivative, and $\mathcal{N}$ is a nonlinear differential operator. The first term corresponds to the time derivative of the neural network output $\mathbf{y}$, while the second term, $\mathcal{N}(\mathbf{y})$, pertains to the ODE followed by the same output. When these two terms are equated ($\mathcal{F}(\mathbf{y}) = 0$), the neural network solution aligns with the ODE.

To train a PINN, it is essential to define appropriate loss functions that consider both observed data and physical constraints. Typical loss functions for ODE-based PINNs consist of two main components:

- Data Loss: This term quantifies the error between the neural network's predictions and the observed data. Typically, a regression cost function such as the Mean Squared Error (MSE) is used to measure this discrepancy. The formula for the data loss is shown in Equation (12).

$$\mathcal{L}_d = \frac{1}{N_t} \sum_{i=1}^{N_t} \left(\hat{y}(t_d^i) - y(t_d^i) \right)^2 \tag{12}$$

where $\{t_d^i\}_{i=1}^{N_t}$ denotes the training data, $y(t_d^i)$ represents the neural network's prediction, $\hat{y}(t_d^i)$ is the observed value in the training data and N_t is the number of training data.

- Physics Loss: This term is essential to ensure that the solution approximated by the PINNs also satisfies the underlying differential equations. For ODEs, this term involves evaluating the residual of the physical equations at the solutions provided by the neural network. The formula for the physics loss term is shown in Equation (13).

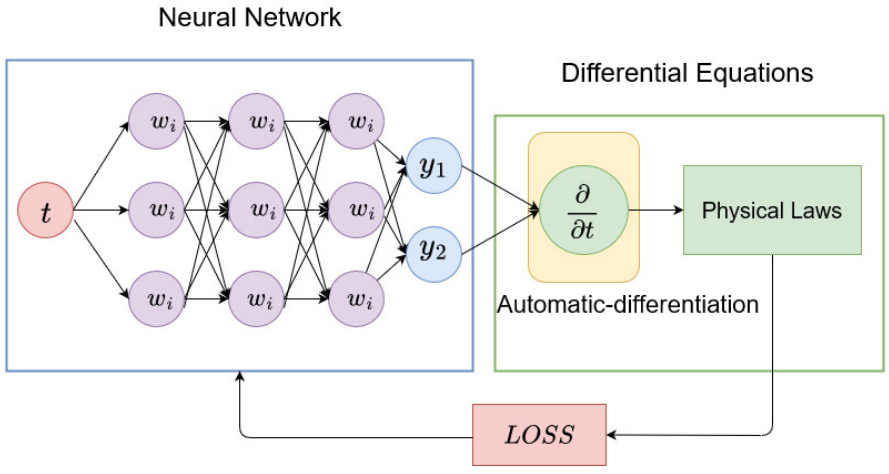
$$\mathcal{L}_{\mathcal{F}} = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| \mathcal{F}(y(t_f^i)) \right|^2, \tag{13}$$

where $\left\{ t_f^i \right\}_{i=1}^{N_f}$ denotes the collocation points, $\mathcal{F}(y(t_f^i))$ represents the evaluation of the residual of the equations at the collocation points using the approximated solution $y(t_f^i)$ and N_f is the number of collocation points. The data-fit term and the physics-fit term are combined to form the overall cost function of the PINNs:

$$\mathcal{L}_t = \mathcal{L}_{\mathcal{F}} + \mathcal{L}_d. \tag{14}$$

This total loss function is then minimized during the neural network training process, allowing the PINNs to learn from both observed data and the underlying physical laws, resulting in an accurate and physically meaningful solution to the ODE. Furthermore, the representation of the PINNs concept is depicted in Figure 8, which illustrates the utilization of automatic differentiation to compute partial derivatives concerning time. The total loss function (14) steers the network's optimization process, aiming to minimize it through iterative adjustments of the network layers, driven by the weights w_i.

Figure 8 – Example of a PINNs applied to an ODE.



Source: Author

A generic PDE can be described by Equation (15) where $t > 0$ and $x \in (a, b)$:

$$y_t = y_{xx}, \tag{15}$$

where y_{xx} represents the second partial derivative with respect to space, and y_t is the partial derivative with respect to time. The boundary conditions are $y(t,a) = 0$ and $y(t,b) = 0$, and the initial condition is a known function $y(0, x) = f(x)$. In the context of PDE applications as shown in Figure 9, the total loss function takes a form where additional terms representing boundary conditions and initial conditions are incorporated, resulting in:

$$\mathcal{L}_t = \mathcal{L}_d + \mathcal{L}_{pde} + \mathcal{L}_{bc} + \mathcal{L}_{ic}. \tag{16}$$

Here, the first loss term $\mathcal{L}_d$ corresponds to the typical loss function for regression (Bishop; Nasrabadi, 2006), which quantifies the error between the network's predictions and the observed data, ensuring that the PINN fits the data accurately. The second loss term $\mathcal{L}_{pde}$ ensures the physical nature of the solution by penalizing discrepancies in the behavior of $y(t, x)$, similar to $\mathcal{L}_{\mathcal{F}}$. This term ensures that the PINN solutions adhere to the partial differential equations. Additionally, two other terms are introduced in Equation (16):

- $\mathcal{L}_{bc}$: represents the loss term associated with enforcing boundary conditions. This term ensures that the PINN solutions satisfy the specified boundary conditions, which are critical in many PDE problems.

$$\mathcal{L}_{bc} = \frac{1}{N_b} \sum_{t=1}^{N_b} \Big(\hat{y}(t_b^i,a) - y(t_b^i,a)\Big)^2 + \Big(\hat{y}(t_b^i,b) - y(t_b^i,b)\Big)^2 \tag{17}$$

where $\{t_b^i\}_{i=1}^{N_b}$ corresponds to the collocation points on the boundary, $\hat{y}(t_b^i, a) - y(t_b^i,a)$ represents the evaluation of the residual of the boundary conditions and N_b corresponds to the number of collocation points on the boundary.

- $\mathcal{L}_{ic}$: represents the loss term for enforcing initial conditions. This term is particularly important for time-dependent PDEs, as it tends to force the PINN to start with the correct initial conditions at the beginning of the simulation.

$$\mathcal{L}_{ic} = \frac{1}{N_0} \sum_{i=1}^{N_0} \Big(f(x_0^i) - y(0,x_0^i)\Big)^2, \tag{18}$$

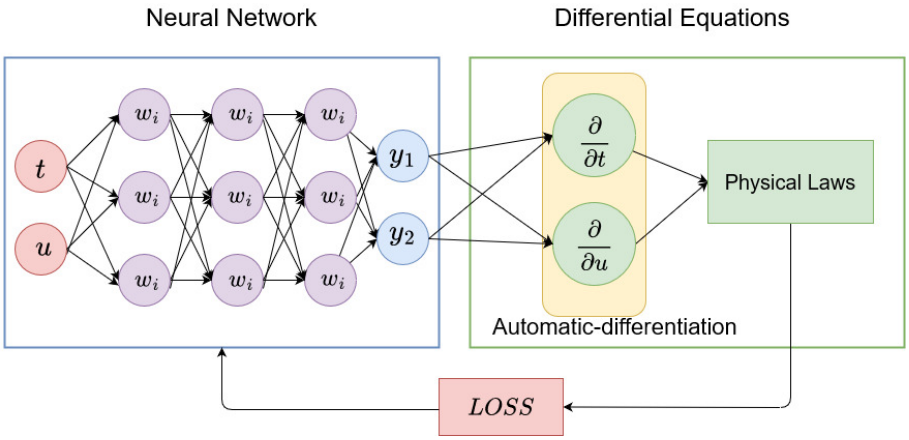
where $\{x_0^i\}_{i=1}^{N_0}$ denotes the initial data, $f(x_0^i) - y(0, x_0^i)$ represents the evaluation of the residuals at $t = 0$, leading to the neural network solution $y(0, x_0^i)$ approaching the initial conditions behavior of the PDE, and N_0 corresponds to the number of initial conditions. It is possible to assign weights to the loss terms in the PDE as follows:

$$\mathcal{L}_t = \lambda_d \mathcal{L}_d + \lambda_{pde} \mathcal{L}_{pde} + \lambda_{bc} \mathcal{L}_{bc} + \lambda_{ic} \mathcal{L}_{ic}. \tag{19}$$

where $(\lambda_d, \lambda_{pde}, \lambda_{bc}, \lambda_{ic})$ represent the loss weights for each term. This adjustment is made to enhance the accuracy of the neural network.

In summary, the total loss function for PINNs in ODE applications uses two terms, combining the data loss and physics loss. In PDE applications, it combines these four terms

Figure 9 – Example of a PINNs applied to an PDE.



Source: Author

to simultaneously fit the observed data, adhere to the governing PDE, the boundary conditions, and the initial conditions. This approach leads to accurate and physically consistent solutions for ODE/PDE problems. To train the weights, gradient descent methods such as ADAM, stochastic gradient descent, or L-BFGS are employed to minimize the loss function as closely as possible, aiming for convergence to zero.

FINAL REMARKS

This chapter provided a comprehensive review of concepts pertinent to this dissertation. The subsequent chapter will delve into an explanation of ESN and the contributions made to the PI-ESNs employed in this study. These contributions encompass the incorporation of external inputs in PI-ESNs, along with the introduction of self-adaptive weights during PI-ESN training (PI-ESN-a). These adaptive weights dynamically balance each term (data loss and physics-based loss) within the loss function.

4 TOWARDS ECHO STATE NETWORKS WITH PHYSICS-INFORMED TRAINING

This chapter explains the architecture of the ESN and the physics regularization implemented for the system identification task. The chapter is organized as follows:

- **Echo State Network (Section 4.1):** This section provides a description of ESN characteristics, outlining key concepts and offering a general overview of the network's operation.
- **Echo State Network Architecture (Section 4.2):** In this section, the ESN architecture with output feedback used in this dissertation is presented, focusing on the training mode and the free-run mode, while presenting the equations for calculating the states and the network's output.
- **Echo State Network Weights and Parameters (Section 4.3)**: This section discusses key parameters that configure an ESN. Understanding these parameters is essential for optimizing performance.
- **Hyperparameter Optimization (Section 4.4)**: This section focuses on hyperparameter optimization within ESNs, covering the process of selecting and tuning the network's hyperparameters to improve its overall performance. It discusses techniques such as grid search, random search, and Bayesian optimization.
- **Physics-Informed Echo State Network (Section 4.5):** This section describes the implementation of physics within the Echo State Network, focusing on how to incorporate physics laws into its loss function.
- **Self-adaptive loss balanced Physics-Informed Echo State Network (Section 4.6):** This section describes the proposed mechanism for a dynamic balance between its loss functions (data loss and physics loss), leading to improved adaptability and performance in complex physics-informed tasks.

4.1 ECHO STATE NETWORKS

The ESN is a type of recurrent neural network that employs supervised learning principles and reservoir computing. It includes a special layer known as the "reservoir", consisting of a large number of randomly interconnected neurons (Jaeger, 2001).

The fundamental concept of reservoir computing is that the reservoir operates dynamically, allowing it to capture complex temporal information from input data. This layer is followed by an output layer that performs a linear transformation to map the captured information to the specific task at hand, such as classification or prediction (Verstraeten et al., 2007).

ESNs consist of a fixed-weight reservoir network and a trainable readout output layer. The reservoir, a randomly generated recurrent layer, transforms inputs into a high-dimensional nonlinear space, while the readout layer maps reservoir states to desired outputs using linear regression (Verstraeten et al., 2007).

ESNs possess memory, allowing the creation of models for dynamic systems based on their historical behavior, making them particularly valuable for tasks like system identification and time series prediction. One of the advantages of ESNs is their low computational resource requirements compared to other recurrent neural network architectures, enabling rapid training and achieving accurate results.

4.2 ECHO STATE NETWORK ARCHITECTURE

Given an input signal $\mathbf{u}[n] \in \mathbb{R}^{N_u}$ and the corresponding output signal $\mathbf{y}[n] \in \mathbb{R}^{N_y}$ for $n = 1, \ldots, N$ time steps, the state update equation for the reservoir states $\mathbf{x}[n] \in \mathbb{R}^{N_x}$ is given by:

$$\mathbf{x}[n+1] = (1-\alpha)\mathbf{x}[n] + \alpha f(\mathbf{W}^{in}\mathbf{u}[n+1] + \mathbf{W}\mathbf{x}[n] + \mathbf{W}^{fb}\mathbf{y}[n] + \mathbf{W}^{b}), \tag{20}$$

where: f is the activation function, usually $\tanh$; $\alpha \in (0,1]$ is the leak rate (Jaeger, 2001); $\mathbf{W}^{in} \in \mathbb{R}^{N_x \times N_u}$ represent the connections from input to the reservoir, $\mathbf{W} \in \mathbb{R}^{N_x \times N_x}$ are the recurrent connections in the reservoir, and $\mathbf{W}^{fb} \in \mathbb{R}^{N_x \times N_y}$ represent the feedback connections from the readout output to the reservoir, $\mathbf{W}^{b} \in \mathbb{R}^{N_x}$ are the bias. The readout output $\mathbf{y}$ is given by:

$$\mathbf{y}[n+1] = \mathbf{W}^{out}\mathbf{x}[n+1], \tag{21}$$

where: $\mathbf{W}^{out} \in \mathbb{R}^{N_y \times N_x}$ are the adaptive weights of the output layer. The ESN architecture is shown in Figure 10 [1]. The solid lines represent the weights that are not trained, whereas the dotted lines represent the trained matrices. The ESN output ($\mathbf{y}$) is fed back to the reservoir to calculate the states with Equation (20).

All connections leading to the reservoir ($\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{b}$ and $\mathbf{W}^{fb}$) are initialized randomly and remain constant. This deliberate randomness creates a matrix of connections, and this randomness is crucial to ensure that each neuron responds in a unique and nonlinear to the input and feedback output signals.

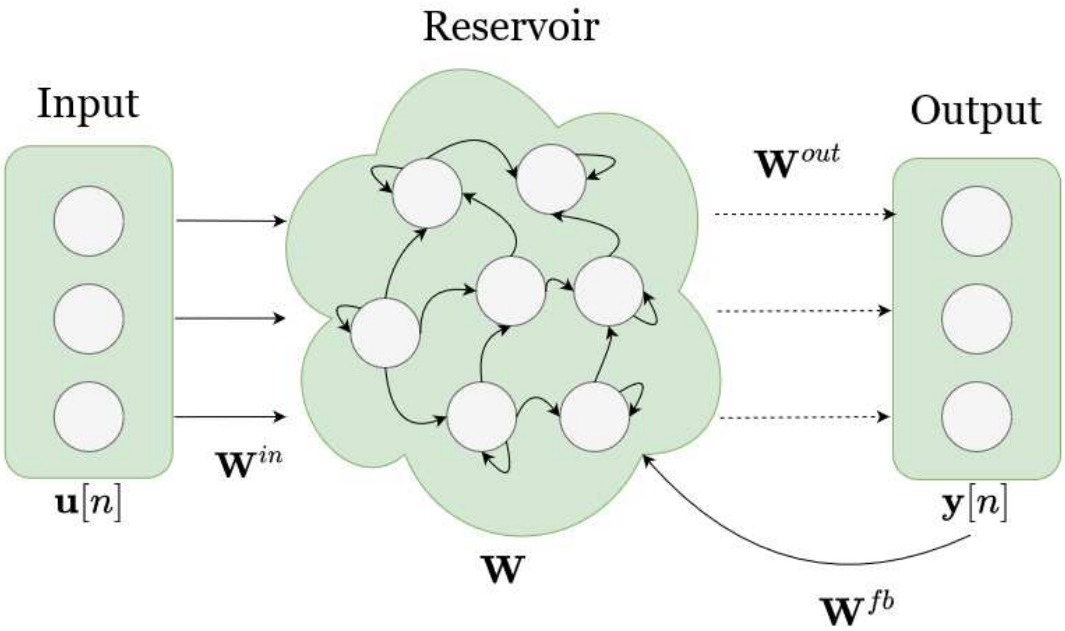
The Echo State Property, a crucial enabling precondition for ESN learning algorithms, refers to the reservoir's ability in an ESN to retain relevant information from previous inputs over a period of time. It involves having a reservoir with fading memory, asymptotically washing out any information from initial conditions.

Figure 11 illustrates that the state update equation comprises three distinct projections: the input projection, the reservoir projection, and the output projection (bias omitted). These projections primarily serve to map the input and output data into the high-dimensional reservoir space, thereby endowing the network with rich dynamic characteristics and self-adaptability.

The visualization process of these projections is shown in Figure 12. This transformation is crucial for the Echo State Network's ability to handle complex information and adapt to

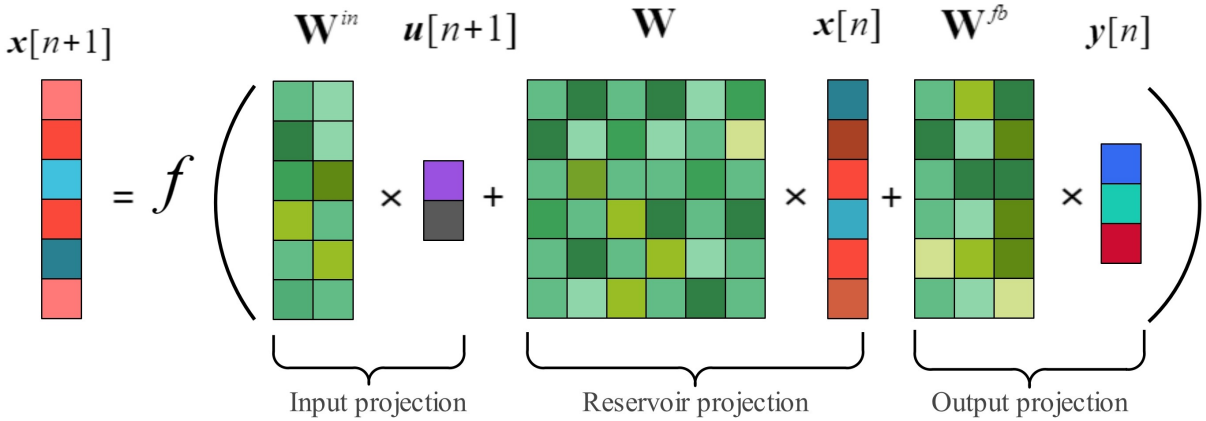
[1] It should be noted that the images presented in this Echo State model have the bias weights omitted

Figure 10 – Echo State Network (ESN) architecture with feedback output.



Source: Author

Figure 11 – Visualizing Projections: Input data, Reservoir, and Output Data in the State Equation



Source: Author

dynamic patterns in the training data (Na et al., 2023). During the state's generation through the projections, the red zone is known to contain transient and unstable states. For training purposes, these states are typically discarded in a process known as reservoir warm-up.

The output layer is typically adjusted with the aid of linear regression techniques, aiming to minimize the mean squared error between the network output $\mathbf{y}$ and the desired output $\hat{\mathbf{y}}$ as shown in Equation (22), while considering a linear mapping from the high-dimensional reservoir layer to the readout output layer (Lukoševičius, 2012).

$$J_{data} = \frac{1}{N_y} \sum_{i=1}^{N_y} \frac{1}{N_t} \sum_{n=1}^{N_t} \Big[\hat{y}_i[n] - y_i[n] \Big]^2, \tag{22}$$

where: N_t represents the number of training data samples available; N_y represents the number of outputs. Ridge regression, expressed in Equation (23), is usually employed to find $\mathbf{W}^{out}$:

$$\mathbf{W}^{out} = \hat{\mathbf{Y}}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \gamma\mathbf{I})^{-1}, \tag{23}$$

where: $\mathbf{X}$ and $\hat{\mathbf{Y}}$ represent column concatenation of the n instants of the ESN states $\mathbf{x}$ and the desired output $\hat{\mathbf{y}}[n+1]$, respectively; and γ is the Tikhonov regularization factor. Each row of matrix $\mathbf{X}$ (Equation (24)) represents a state, and each column represents the instant at which they were calculated. $\hat{\mathbf{Y}}$ is defined similarly (Equation (25)):

$$\mathbf{X} \in \mathbb{R}^{N_x \times N_t} = \begin{bmatrix} x_1[1] & \dots & x_1[N_t] \\ \vdots & \ddots & \vdots \\ x_{N_x}[1] & \dots & x_{N_x}[N_t] \end{bmatrix} \tag{24}$$

$$\hat{\mathbf{Y}} \in \mathbb{R}^{N_y \times N_t} = \begin{bmatrix} \hat{y}_1[1] & \dots & \hat{y}_1[N_t] \\ \vdots & \ddots & \vdots \\ \hat{y}_{N_y}[1] & \dots & \hat{y}_{N_y}[N_t] \end{bmatrix} \tag{25}$$

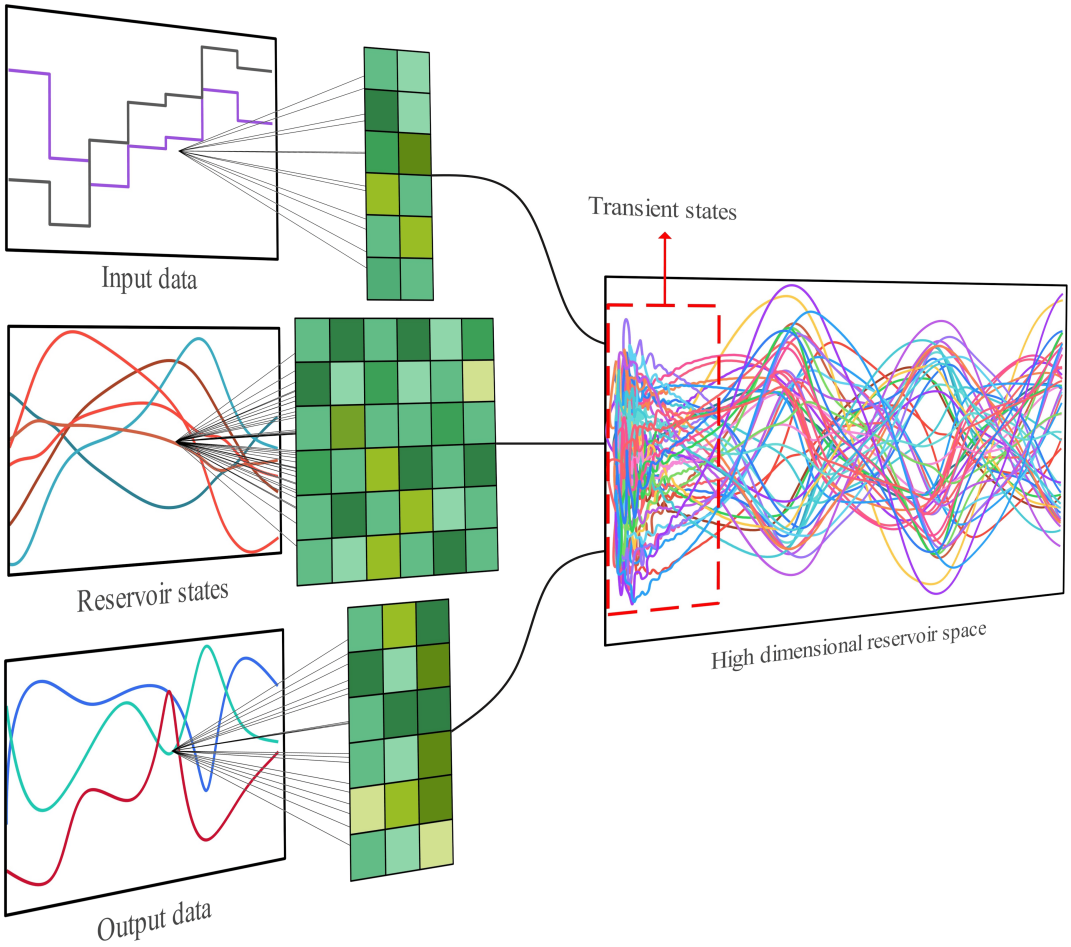
When generating $\mathbf{x}[n]$ to construct the design matrix $\mathbf{X}$ in Equation (24), the output follows two different modes of operation:

- In training mode (Figure 13 (a)), the so-called teacher forcing technique is employed, so that the desired output $\hat{\mathbf{y}}[n]$ is transmitted back to the reservoir with the weights in $\mathbf{W}^{fb}$, as exemplified in Equation (26). Hence, during this phase, $\hat{\mathbf{y}}[n]$ is employed to compute $\mathbf{X}$, $\hat{\mathbf{Y}}$, and subsequently, train the output layer ($\mathbf{W}^{out}$).
- In the free-run mode (Figure 13 (b)), which is adopted after the output layer $\mathbf{W}^{out}$ has been trained, the actual output $\mathbf{y}[n]$ (ESN output prediction) is fed back into the reservoir to update the states, as demonstrated in Equation (20). Therefore, this mode operates independently of the training data, and the network no longer relies on it to calculate the states and the output $\mathbf{y}[n]$.

$$\mathbf{x}[n+1] = (1-\alpha)\mathbf{x}[n] + \alpha f(\mathbf{W}^{in}\mathbf{u}[n+1] + \mathbf{W}\mathbf{x}[n] + \mathbf{W}^{fb}\hat{\mathbf{y}}[n] + \mathbf{W}^{b}) \tag{26}$$

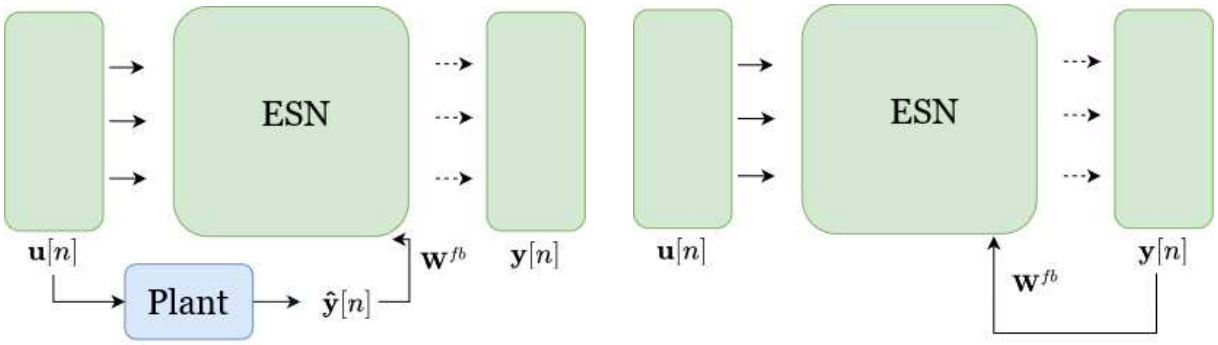
This desired output $\hat{\mathbf{y}}$ comes from an industrial plant or from a phenomenological model as collected data, for instance. In Figure 13, the input $\mathbf{u}$ is also fed to the plant, which in turn generates the target $\hat{\mathbf{y}}$. After training, Equation (26) is used for a few initial time steps to warm-up the reservoir, in the sequence Equation (20) is employed, where the actual output prediction $\mathbf{y}$ is fed back to the reservoir.

Figure 12 – Visualization of ESN reservoir formation through Projections.



Source: Author

Figure 13 – Echo state network in training mode and free-run mode



(a) Teacher-forcing mode

(b) Free-run mode

Source: Author

4.3 ECHO STATE NETWORK WEIGHTS AND PARAMETERS

The weight matrices, particularly the reservoir matrix $\mathbf{W}$, play a crucial role in determining the effectiveness and stability of the model. This is especially true for the reservoir

matrix, as it is essential for ensuring the Echo State Property [2]. The method by which the weights were generated for the experiments in the dissertation is presented:

- $\mathbf{W}$ is randomly generated from a uniform distribution in the interval $[-1, 1]$.
- $\mathbf{W}^{in}$ values are selected from the set 0, δ_{in}, $-\delta_{in}$ with probabilities of 0.5, 0.25, and 0.25, respectively, where δ_{in} serves as an input scaling factor.
- $\mathbf{W}^{fb}$ is randomly generated from a uniform distribution in the interval [δ_{fb}, $-\delta_{fb}$], where δ_{fb} functions as an output scaling factor.
- $\mathbf{W}^{b}$ is generated from a uniform distribution in the interval [δ_{b}, $-\delta_{b}$], where δ_{b} functions as a bias scaling factor.

The following parameters are essential in the Echo State Network (ESN) for enhancing its performance:

- **Spectral Radius** (ρ): This parameter, denoted as $\rho(\mathbf{W})$, is a scaling factor selected to ensure that $\rho(\mathbf{W})$ is less than 1; this helps ensure the Echo State Property [3]. It is defined as $\rho(\mathbf{W}) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } \mathbf{W}\}$. The spectral radius is usually selected as close as possible to 1, where the reservoir operates at the edge of stability. This is done to generate more diverse signals that can contribute to the identification of the dynamics of a system. It is worth mentioning that, although $\rho(\mathbf{W}) > 1$ causes self-induced internal dynamics in the reservoir, such a network can still be trained for some tasks, as it can become stable. Therefore, the spectral radius is a hyperparameter that can be selected to control the learning process. To this end, the selected spectral radius (ρ^*) is used in Equation (27) (Yildiz; Jaeger; Kiebel, 2012). Here, ρ^* is a hyperparameter that scale $\mathbf{W}$ so that its spectral radius is equal to the value of ρ^* (Yildiz; Jaeger; Kiebel, 2012):

$$\mathbf{W} = \mathbf{W}^* \frac{\rho^*}{\rho(\mathbf{W}^*)}, \tag{27}$$

  where $\mathbf{W}^*$ represents random values before the application of the desired spectral radius value.

- **Leak Rate** (α): It is a parameter that regulates the extent to which states "leak" over time, and it takes values between 0 and 1. A smaller value of α moves the reservoir toward a slower dynamic state, enhancing its memory of previous states. Conversely, a leak rate of 1 corresponds to complete state decay, effectively resetting the state at each time step (Jaeger, 2001). Typically set between 0 and 1, a moderate leak rate strikes a balance, enabling the ESN to retain some information from pastime steps while accommodating new input data. This balance can be

[2] In the literature, various techniques are proposed for reservoir weight design without relying on random initialization (Verstraeten et al., 2007; Schrauwen et al., 2008; Boccato; Attux; Von Zuben, 2014)

[3] This does not guarantee the Echo State Property, as demonstrated in the work of Yildiz, Jaeger, and Kiebel (2012), where further considerations are discussed.

fine-tuned to control the network's memory and its capacity to capture temporal dependencies within the data.

- **Regularization factor** (γ): Applied to the training of the output layer's weight matrix ($\mathbf{W}^{out}$) in Equation (23), this factor is utilized to prevent excessively large weights during training. Large weights can lead to overfitting, where the network closely fits the training data, capturing noise rather than underlying patterns, resulting in poor generalization to new data. Tikhonov's regularization addresses this issue by introducing a penalty term to the training objective, which prevents the $\mathbf{W}^{out}$ values from becoming too large.
- **Reservoir Size** (N_x): The choice of reservoir size should be based on task complexity and available training data. A larger reservoir can capture intricate patterns and often leads to better performance but requires more data. If the available training data is limited, a large reservoir might not be able to generalize well and could lead to overfitting. In contrast, a smaller reservoir is suitable for simpler tasks or limited data situations.
- **Scaling** ($\delta_{in}, \delta_{fb}, \delta_b$): Input, output, and bias scaling parameters control the influence of their respective weights on the network. They regulate the magnitude of the input data, output data, and bias in the state calculation.
- **Warm-up**: The primary purpose of the warm-up phase is to allow the network to transition from its initial state, often set to all zeros, to a state that captures the underlying dynamics of the input data (as can be seen in Figure 12). The duration of the warm-up phase can vary depending on the complexity of the system. Generally, it is recommended to discard the initial states during the warm-up phase, as they may contain transient and unstable states.
- **Data Normalization**: Data normalization is a preprocessing step in ESNs. It involves transforming the input and output data to have a standardized range or distribution. Common methods for data normalization include min-max scaling, z-score normalization, or any other technique that scales the data to a common range or centers it around zero with a standard deviation of one. The min-max scaling can be defined as:

$$y_{scaled} = \frac{y - y_{min}}{y_{max} - y_{min}}, \tag{28}$$

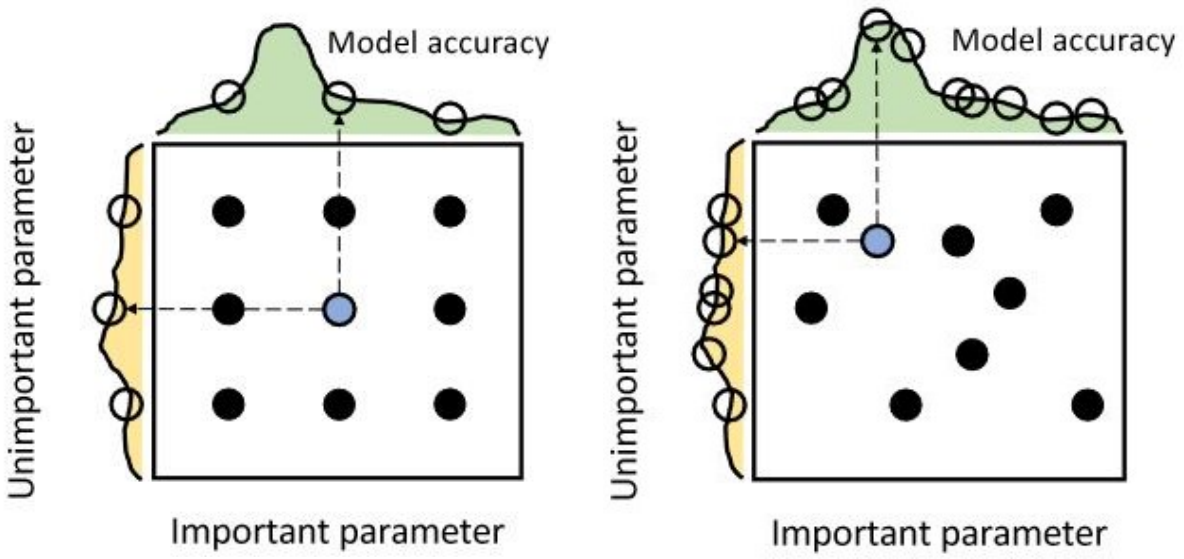
where y is the original value, y_{scaled} is the scaled value, y_{min} is the minimum value in the dataset, and y_{max} is the maximum value in the training dataset. Normalizing the data ensures that input and output signals are within a consistent range, preventing issues related to varying data scales. This preprocessing step contributes to the stability and effectiveness of the ESN by ensuring that the activation function can capture patterns across different input features and data ranges.

4.4 HYPERPARAMETERS OPTIMIZATION

To optimize the ESN hyperparameters, the dataset $(1, \ldots, N_t)$ is divided into a training set $(1, \ldots, N_{te})$ and a validation set $(1 + N_{te}, \ldots, N_t)$. Hence, $N_{te} + N_{ve} = N_t$, where N_{te} represents the number of training data samples and N_{ve} stands for the number of validation data samples used in the hyperparameter search. Three types of optimization methods will be discussed: grid search, random search and Bayesian optimization.

Figure 14 displays and compares grid search and random search optimization. The black points represent the hyperparameter values that are being searched, namely, important and unimportant parameters. Model accuracy, typically represented as Mean Squared Error (MSE), is depicted as a function of each search dimension. The grid search and random search differ in how they explore the hyperparameter space.

Figure 14 – Hyperparameter Tuning with Grid Search and Random Search



(a) Grid Search (b) Random Search

Source: Pilario, Cao, and Shafiee (2021)

In grid search, a predetermined set of hyperparameter values is defined for each parameter, and all possible combinations are systematically evaluated, resembling a structured grid search. It adopts a discrete approach by specifying a predefined grid of values for each hyperparameter and evaluating the model's performance at each grid point.

It essentially tests all possible combinations of hyperparameters on the grid. Grid search is relatively straightforward to implement and comprehend, but it may become inefficient when the region containing the best hyperparameters is unknown, as it evaluates all combinations, including less promising ones (Jaeger et al., 2007). The random search selects hyperparameter values randomly from predefined ranges, exploring the space through random sampling, which can be more efficient, especially with extensive search spaces.

Figure 15 represents how Bayesian optimization operates by deriving and optimizing the score function or acquisition function of the model, which expresses the most promising setting for the next iteration, based on the difference between observations and the objective function (in red), typically measured using MSE and the uncertainty. The model's quality improves progressively over time as successive measurements are incorporated (Greenhill et al., 2020). Bayesian optimization is a continuous approach that explores the hyperparameter space iteratively. It utilizes a probabilistic model, often a Gaussian regression model with mean represented in black and uncertainty in blue, to estimate the model's performance concerning hyperparameters.

This probabilistic model, known as surrogate function, helps guide the optimization process towards promising regions. The surrogate function is a simplification of the objective function. It is chosen to be a simpler, often parameterized function that represents an approximation of the behavior of the real objective function. Bayesian optimization is generally more efficient than grid search, as it focuses on areas with a high likelihood of containing the best hyperparameters. However, it can be more complex to set up due to the need to define a search space, an acquisition function, and select an appropriate probabilistic model (Yperman; Becker, 2017).

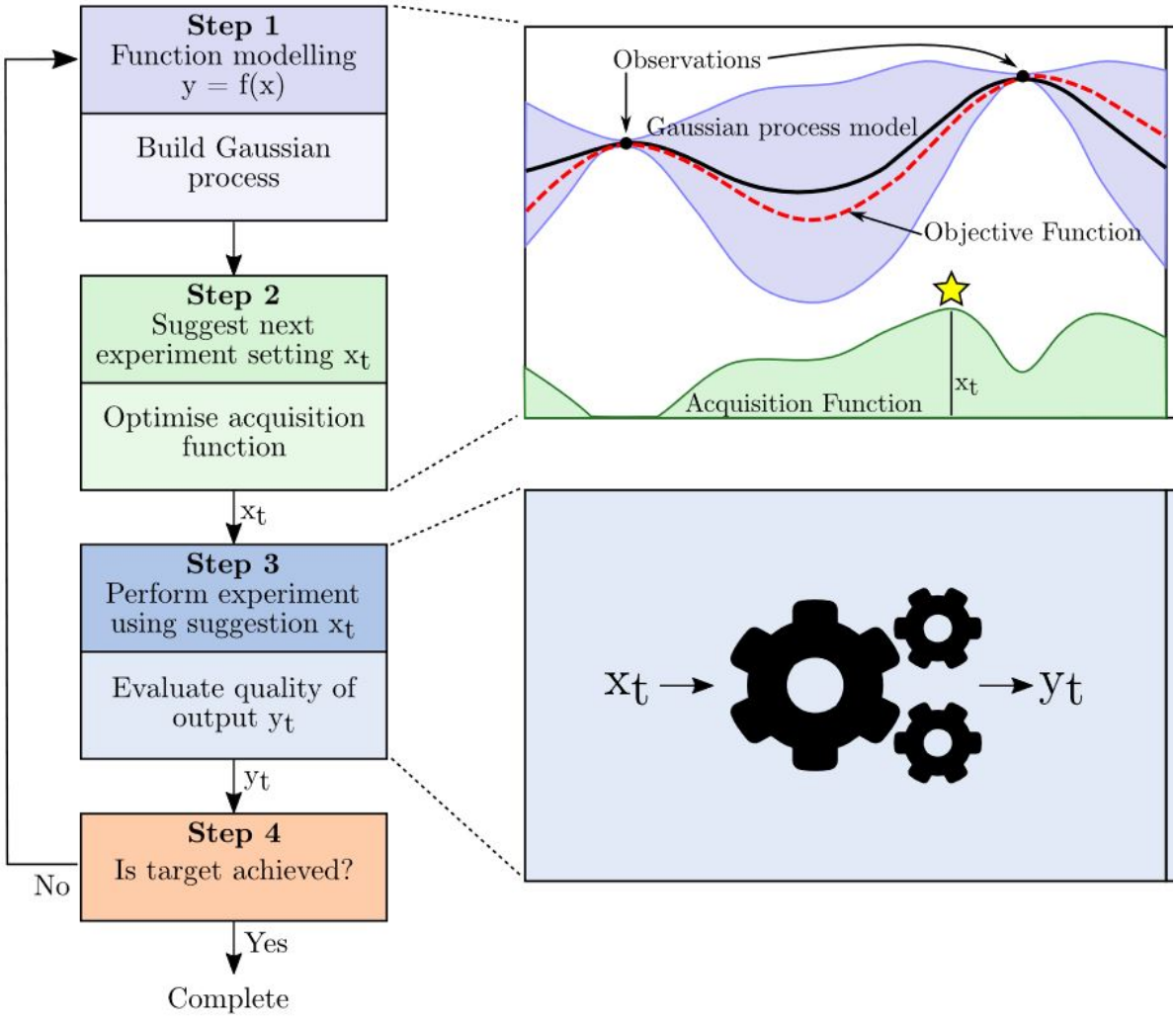
Each approach, Bayesian optimization and grid search, has its advantages and limitations. The decision between these two methods depends on several factors, including the problem's nature and the resources available for hyperparameter optimization in an ESN. While these methods are practical for most use cases, it is important to remark that they primarily rely on statistical measures. As a result, they can only approximate the local optimal, leaving room for further optimization (Thiede; Parlitz, 2019).

In this study, the optimization process was customized for each dynamic system, utilizing grid search or Bayesian optimization to select the hyperparameters for tuning. Once the best hyperparameter values were identified, the ESN was retrained using all available data $(1, \ldots, N_t)$.

4.5 PHYSICS-INFORMED ECHO STATE NETWORK

A traditional PINNs (Figure 8) needs a time t as input. This input does not exist in the PI-ESN since the ESN is a discrete-time recurrent network that implicitly incorporates time through the network state update equation (Equation (20)), where the next state depends on the previous state. This architecture has already been applied to autonomous chaotic systems in which the PI-ESN has the feedback output into the reservoir, but does not have any other input besides the feedback (Doan; Polifke; Magri, 2019b). In this work, the PI-ESN was extended to accept external inputs, such as a plant control, in addition to the output feedback itself as shown in Figure 16. The ESN output $(\mathbf{y})$ is used to calculate the physics loss function. Then the $(\frac{\partial J}{\partial \mathbf{W}^{out}})$ is calculated by automatic differentiation and used to update the values of the output layer $(\mathbf{W}^{out})$ in an optimization process that aims to minimize the

Figure 15 – Bayesian Optimization.



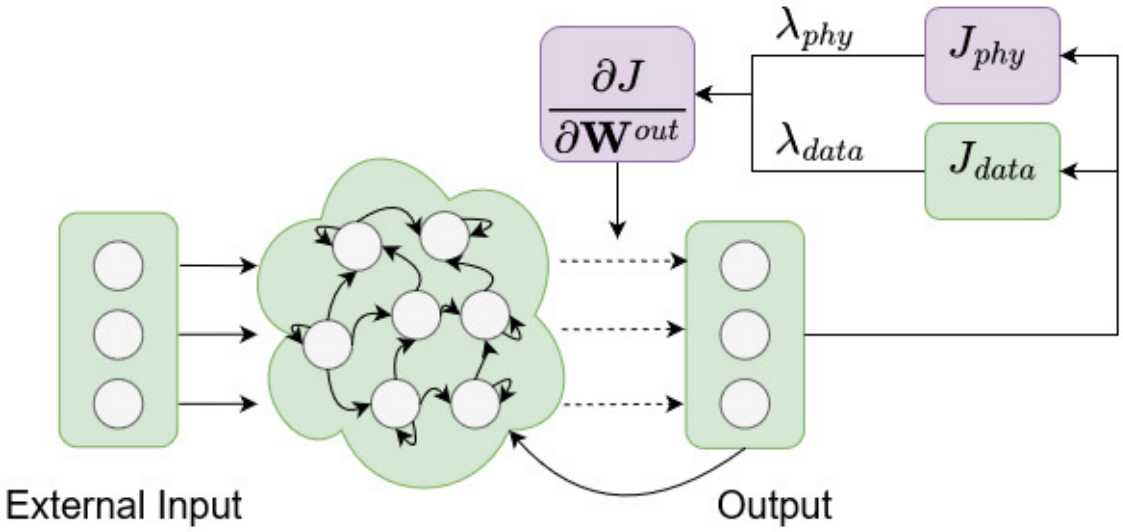
Source: Greenhill et al. (2020)

total loss function.

As with standard ESNs, the only trainable layer is the output layer $\mathbf{W}^{out}$. For PI-ESN training, the starting point is the output layer calculated through traditional ESN training (within the time window $[0, T]$ with N_t data) as an initial guess for computing the new values of $\mathbf{W}^{out}$. To use the physics laws in the ESN, it is needed the neural network output ($\mathbf{y}[n]$) obtained during the ESN free-run mode, in the time window $t \in (T, T_f]$, with $N_t + 1, \ldots, N_f + N_t$ data samples, where N_f is the number of collocation points as shown in Figure 17.

These values can be applied to approximate the states of a dynamical system described by ODE's. In Figure 17 the solid black represents the input data $\mathbf{u}[n]$, which is used in both the data loss function (in the time steps $n, \ldots, N_t$) and the physics loss function (in the time steps $N_t + 1, \ldots, N_t + N_f$). The training output data $\hat{\mathbf{y}}[n]$ (during time steps $n, \ldots, N_t$) is shown in dark green. These values are used for teacher forcing in the ESN training (see Figure 13) and to calculate Equation (25). After the training of the ESN, these values will be used in Equation (22) for the PI-ESN training. ESN output ($\mathbf{y}[n]$) in the time steps $N_t + 1, \ldots, N_t + N_f$

Figure 16 – Physics-Informed Echo State Network (PI-ESN)



Source: Author

is shown in pink. These values are used in Equation (29), as represented in the scheme inside the blue rectangle, to calculate the physics loss in Equation (30). In light green, it is the real system output that is desired for approximation, but is not used to calculate the loss function or any other equation in physics training.

The PI-ESN has no time input explicitly and works in discrete time. Hence, Equation (11) must be discrete using numeric methods, such as explicit Euler, or Runge-Kutta, for example (Doan; Polifke; Magri, 2019b) or by utilizing more accurate differentiation techniques as suggested in Racca and Magri (2021). Using explicit Euler and assuming that $\partial_t \mathbf{y} - \mathcal{N}(\mathbf{y}) = 0$, it has the following:

$$\mathcal{F}(\mathbf{y}[n]) \equiv \mathbf{y}[n+1] - (\mathbf{y}[n] + \mathcal{N}(\mathbf{y}[n])\Delta t) \tag{29}$$

This $\mathcal{F}$ function is applied at each collocation point, generating the physics loss function J_{phy}:

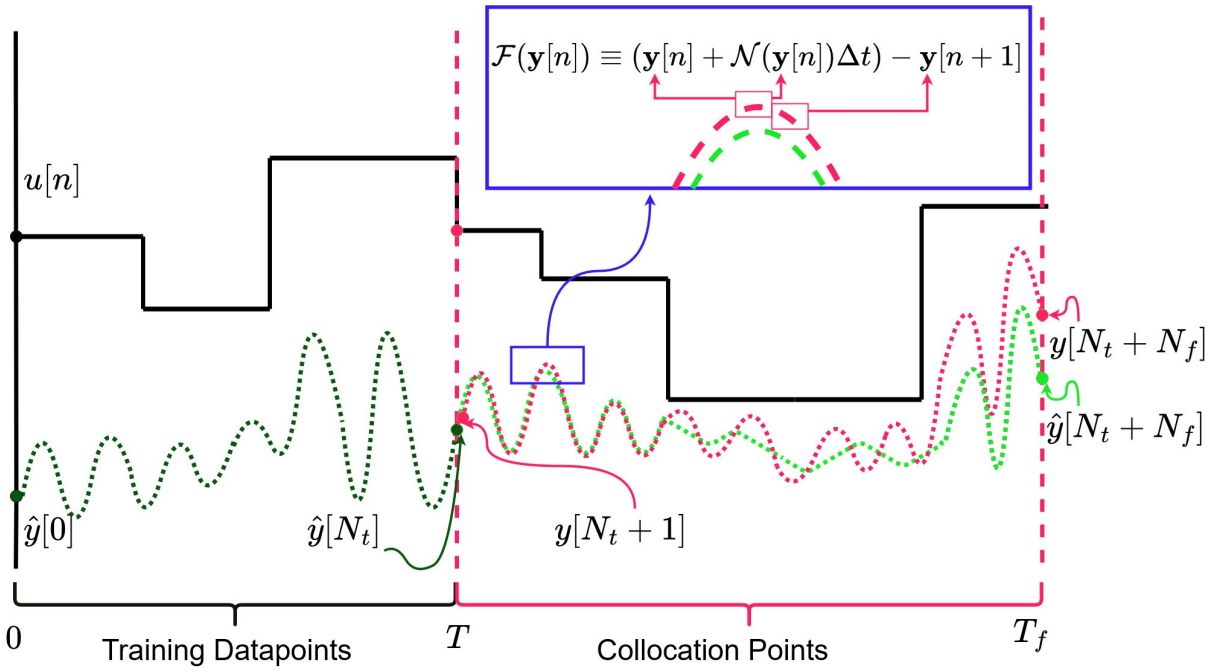
$$J_{phy} = \frac{1}{N_y} \sum_{i=1}^{Ny} \frac{1}{N_f} \sum_{n=N_t+1}^{N_f} |\mathcal{F}(y_i[n])|^2 \tag{30}$$

The total loss function to be minimized considers both data loss and the residual function of physics laws:

$$J = \lambda_{data} \cdot J_{data} + \lambda_{phy} \cdot J_{phy}, \tag{31}$$

where λ_{data} and λ_{phy} are hyperparameters used to balance the importance of the loss functions during optimization. Typically, the parameter λ_{data} is set to a value of 1, while only the value of the parameter λ_{phy} is adjusted. The total loss function is minimized by finding the

Figure 17 – Representation of the PI-ESN collocation points used to calculate the physics loss function.



Source: Author

value of the output weight matrix $\mathbf{W^{out}}$ through the minimization of the objective function $J(\mathbf{W^{out}})$, employing ADAM or L-FBGS optimizers available in frameworks like TensorFlow or PyTorch. It is important to recall that the automatic differentiation does not pass through the recurrence of the reservoir, that is, there is no retropropagation of errors through time.

This happens because the matrix $\mathbf{X}_f$ in Equation (32) (similar to the Equation (24) but with time steps $N_t + 1, \ldots, N_t + N_f$) is used to stabilize the PI-ESN training, updating the values of $\mathbf{X}_f$ every j optimization iterations. Since $\mathbf{X}_f$ depends on the value of $\mathbf{W}^{out}$ because there is feedback from the previous output $\mathbf{y}[n]$ in the states $\mathbf{x}[n+1]$, the values of all states would change in each iteration j of the optimizer, i.e., in each update of $\mathbf{W}^{out}$. To prevent this from happening, the states are only updated after K iterations, stabilizing the training process.

$$\mathbf{X}_f \in \mathbb{R}^{N_x \times N_f} = \begin{bmatrix} x_1[N_t+1] & \ldots & x_1[N_t+N_f] \\ \vdots & \ddots & \vdots \\ x_{N_x}[N_t+1] & \ldots & x_{N_x}[N_t+N_f] \end{bmatrix} \tag{32}$$

This is equivalent to use two versions of $\mathbf{W}^{out}$, one that is constantly updated by training, and another used to calculate the ESN output (Equation (21)) which is updated every K iteration to the value of the first.

Algorithm 1 shows how the generation of $\mathbf{X}_f$ and the update of $\mathbf{W}^{out}$ work with the adaptive loss function. Although the temporal dependence exists, for practical purposes, the training disregards this, which simplifies and speeds up the training. Experiments carried out

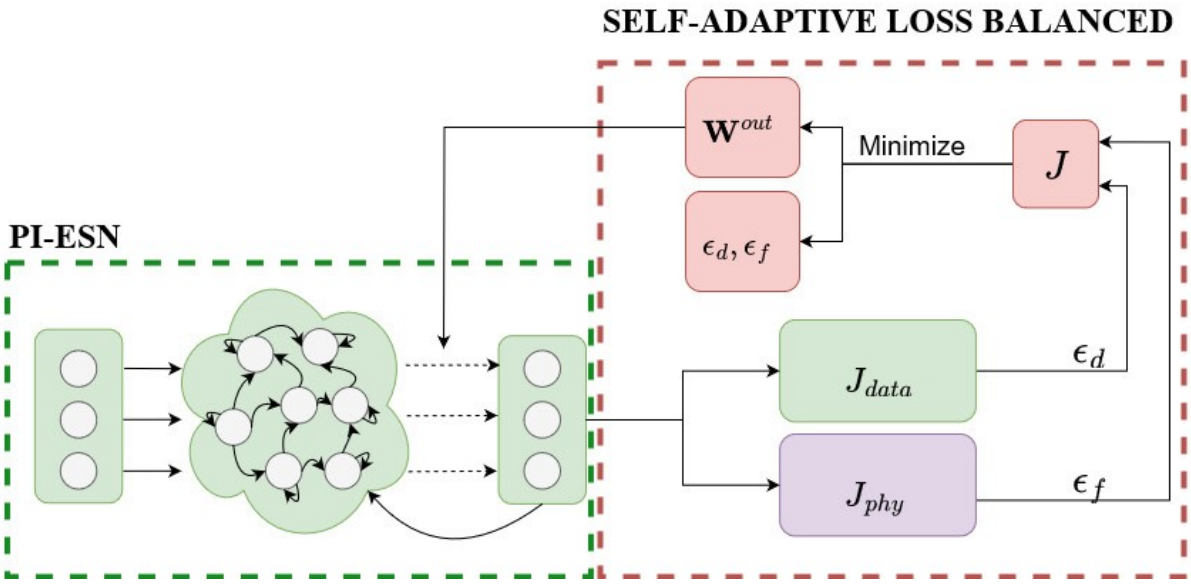
considering the temporal dependence did not achieve good results.

4.6 ADAPTIVE LOSS FUNCTION FOR PI-ESN

The Balanced Self-Adaptive Loss Physics-Informed Neural Networks (lb-PINNs), proposed by (Xiang et al., 2022), is a self-adaptive loss balanced approach that automatically updates weights for each loss term during the optimization process. Their work applied lb-PINNs to solve partial differential equations such as Navier-Stokes, Allen-Cahn, and Poisson. The main idea is based on Maximum Likelihood Estimation (MLE) of Gaussian parameters, adapting it to a minimization problem. MLE is a technique in machine learning to accomplish the task of creating a statistical model that can perform a given task on unseen data (data not available for training phase). This technique can be used to determine the values of λ_{data} and λ_{phy}, without requiring additional data during the physics training phase. It relies solely on the training data that are initially available.

The balanced self-adaptive loss algorithm, originally developed for traditional PINNs, is here extended to PI-ESNs, leading to the version henceforth denoted as PI-ESN-a. An illustration of the PI-ESN-a is shown in Figure 18. The values of the output ($y[n]$) are utilized to compute the total loss (Equation (35)). Subsequently, an optimizer is employed to minimize this function and obtain the new values for the weight matrix $\mathbf{W}^{out}$ as well as for the adaptive weights ϵ_d and ϵ_f. This optimization procedure can be repeated multiple times as outlined in Algorithm 1.

Figure 18 – Self-Adaptive Physics-Informed Echo State Network (PI-ESN-a).



Source: Author

Assuming that the output of our Gaussian probability model consists of two vectors, denoted as $\hat{y}_1$ and $\hat{y}_2$, following a Gaussian distribution ($\mathcal{N}_g$), the likelihood function is defined

as a Gaussian distribution with mean equal to the approximation of the PI-ESN $y(\mathbf{W}^{out};\mathbf{u})$, and the uncertainty captured by the variances ϵ_d and ϵ_f:

$$p(\hat{y}_1,\hat{y}_2|y(\mathbf{W}^{out};\mathbf{u})) = p(\hat{y}_1|y(\mathbf{W}^{out};\mathbf{u}))\cdot p(\hat{y}_2|y(\mathbf{W}^{out};\mathbf{u})) \tag{33}$$

$$p(\hat{y}_1,\hat{y}_2|y(\mathbf{W}^{out};\mathbf{u})) = \mathcal{N}_g(y(\mathbf{W}^{out};\mathbf{u}),\epsilon_d^2)\cdot\mathcal{N}_g(y(\mathbf{W}^{out};\mathbf{u}),\epsilon_f^2), \tag{34}$$

The uncertainty parameters ϵ_d and ϵ_f are predetermined as part of the weight decay process in physics-informed neural network. The objective is to minimize the negative log-likelihood of the model while simultaneously pursuing the minimization of $\mathbf{W}^{out}$ and the uncertainty parameters ϵ_d and ϵ_f (Equation (35)). Furthermore, the network can automatically assign weights to data loss by updating the uncertainty parameter ϵ_d in each epoch using maximum likelihood estimation.

$$\begin{aligned}-\log p(\hat{y}_1,\hat{y}_2|y(\mathbf{W}^{out};\mathbf{u})) \propto \frac{1}{2\epsilon_d^2}\left\|\hat{y}_1 - y(\mathbf{W}^{out};\mathbf{u}))\right\|^2 \\ + \frac{1}{2\epsilon_d^2}\left\|\hat{y}_2 - y(\mathbf{W}^{out};\mathbf{u}))\right\|^2 + \log\epsilon_d\epsilon_f,\end{aligned} \tag{35}$$

where $J_{data} = \left\|\hat{y}_1 - y(\mathbf{W}^{out};\mathbf{u})\right\|^2$ and $J_{phy} = \left\|\hat{y}_2 - y(\mathbf{W}^{out};\mathbf{u})\right\|^2$. The minimization objective of the model can be expressed as:

$$L(\mathbf{W}^{out},\epsilon;\mathbf{N}) = \frac{1}{2\epsilon_d^2}J_{data}(\mathbf{W}^{out};N_t) + \frac{1}{2\epsilon_f^2}J_{phy}(\mathbf{W}^{out};N_y,N_f) + \log\epsilon_d\epsilon_f, \tag{36}$$

where $\epsilon = \{\epsilon_d,\epsilon_f\}$ denotes the adaptive weights for the data loss and physics loss, respectively, and $\mathbf{N} = \{N_t,N_y,N_f\}$. The variance parameters can be expressed as $\mathbf{s} = \{s_d,s_f\}$ to use the exponential mapping ($\mathbf{s} = \log(\epsilon^2)$), as shown in Equation (37). By taking the exponential of the variance parameters $\mathbf{s}$, which guarantees that the resulting value is in the positive domain, the adaptive weight will not converge to zero too quickly and is more numerically stable for the training process (Xiang et al., 2022).

$$\begin{aligned}L(\mathbf{W}^{out},\mathbf{s};\mathbf{N}) = \frac{1}{2}\exp(-s_d)J_{data}(\mathbf{W}^{out};N_t) \\ + \frac{1}{2}\exp(-s_f)J_{phy}(\mathbf{W}^{out};N_y,N_f) + s_d + s_f\end{aligned} \tag{37}$$

Algorithm 1: Training of PI-ESN with external inputs and Self-Adaptive Balancing Loss.

input: M, K, $\mathbf{s} = [s_d, s_f]^T$, $\mathcal{F}$, $\{\mathbf{u}[n] : n = 1, \ldots, N_t + N_f\}$, $\{\hat{\mathbf{y}}[n] : n = 1, \ldots, N_t\}$;

Using $\{(\mathbf{u}[n], \hat{\mathbf{y}}[n]) : n = 1, \ldots, N_t\}$, build $\mathbf{X}$ by Equation (24) and $\hat{\mathbf{Y}}$ by Equation (25); `// training data`

Pretrain PI-ESN weights $\mathbf{W}^{out}$ by ridge regression with Equation (23) using $\mathbf{X}$ and $\hat{\mathbf{Y}}$;

for M_1 iterations **do**

  Generate $\mathbf{X}_f$ using Equation (20) and Equation (21), updated $\mathbf{W}^{out}$, and $\mathbf{u}[n]$ for timesteps $n = N_t + 1, \ldots, N_t + N_f$; `// collocation points`

  for M_2 iterations **do**

    `// Adapting` $\mathbf{W}^{out}, s_d, s_f$ `to minimize total loss`

    Compute ESN's outputs for the data points $\mathbf{Y}_t = \mathbf{W}^{out}\mathbf{X}$ and for collocation points $\mathbf{Y}_f = \mathbf{W}^{out}\mathbf{X}_f$

    Define $J_{\text{data}}(\mathbf{W}^{out})$ loss on $\mathbf{Y}_t$ and target $\hat{\mathbf{Y}}$ as in Equation (22);

    Define $J_{\text{physics}}(\mathbf{W}^{out})$ loss on $\mathcal{F}(\mathbf{Y}_f)$ as in Equation (30);

    Combine both losses into $L(\mathbf{W}^{out}; \mathbf{s})$ as in Equation (37) and compute its gradient with respect to $\mathbf{W}^{out}, s_d, s_f$;

    Update $\mathbf{W}^{out}, s_d, s_f$ with an optimizer and the obtained gradients;

output: $\mathbf{W}^{out}$

FINAL REMARKS

This chapter presented the architectures of ESNs and their adaptation for Physics-Informed Echo State Networks (PI-ESNs). It outlined the contributions made by incorporating an external input into the PI-ESN architectures and adding adaptive weights to balance the loss functions of the PI-ESN-a. In the next chapter, experiments utilizing the proposed architectures from this chapter will be showcased, applying them to three systems. These include two described by ODEs: the van der Pol oscillator and a four-tank system, and one described by DAEs, specifically an electric submersible pump.

5 EXPERIMENTS

In this chapter, the experiments conducted using PI-ESN and PI-ESN-a are described on three different systems: the Van der Pol system, the Four-Tank system, and an Electric Submersible Pump. The chapter is structured as follows:

- **Van der Pol Oscillator (Section 5.1):** This section provides a description of both the unforced and forced Van der Pol systems, both of which are utilized for system identification with PI-ESN and PI-ESN-a. Additionally, a step-by-step explanation of the alterations made to an ESN when applying physics regularization is provided.
- **Four-tank system (Section 5.2):** In this section, the Four-Tank system is introduced. It also highlights the application of physics-informed machine learning techniques in the experimental approach with PI-ESN-a.
- **Electric Submersible Pump (Section 5.3):** In this section, an introduction is provided to artificial lifting systems within the oil and gas industry, with a particular emphasis on the ESP (Electric Submersible Pump) system and the utilization of PI-ESN-a for system identification.

5.1 VAN DER POL OSCILLATOR

Extensive research has been conducted on the Van der Pol Oscillator, which is described by Equation (38), with the aim of enhancing the approximations of solutions to nonlinear systems. This self-oscillatory dynamical system is widely recognized as a valuable mathematical model that can be utilized for more complex systems. It is a second-order ordinary differential equation featuring cubic nonlinearity (Tsatsos, 2008).

$$\ddot{h} - \mu(1 - h^2)\dot{h} + h = 0, \tag{38}$$

where μ represents the damping parameter, which influences the system's oscillation as shown in Figure 19. Equation (38) can be write in bidimension form, in the unforced form:

$$\begin{aligned} \dot{h}_1 &= h_2 \\ \dot{h}_2 &= \mu(1 - {h_1}^2)h_2 - h_1 \end{aligned} \tag{39}$$

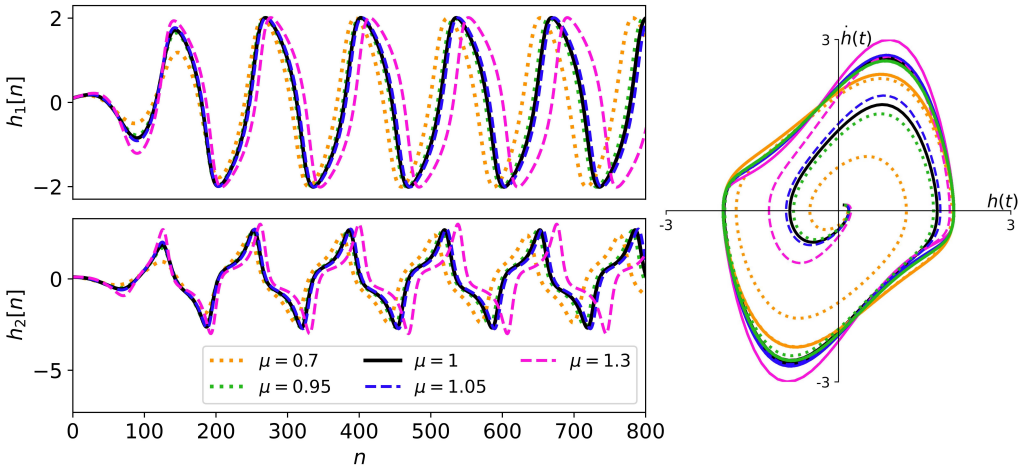
In the forced form, an external input u is added, resulting in the following model:

$$\begin{aligned} \dot{h}_1 &= h_2 \\ \dot{h}_2 &= \mu(1 - {h_1}^2)h_2 - h_1 + u \end{aligned} \tag{40}$$

Figure 19, generated with initial conditions $h_1(0) = h_2(0) = 0.1$, illustrates the oscillatory behavior, which is dependent on the damping parameter (μ). On the left-hand side of the image, the influence of varying the parameter μ on the outputs h_1 and h_2 can be observed. On the right-hand side, the phase space of the oscillator is represented, describing the trajectory

of the dynamic system in a multidimensional space. As the system oscillates over time, the trajectories in the phase space depict how the variables evolve. The Van der Pol oscillator is known for exhibiting various behaviors in phase space, including limit cycles and chaotic behavior, depending on the system parameters such as the damping coefficient μ and initial conditions.

Figure 19 – Unforced Van der Pol oscillator system with oscillation variation.



Source: Author

5.1.1 Unforced Equation

5.1.1.1 Dataset

The dynamic behavior of the Van der Pol oscillator system was simulated using Equation (39), and the simulation was carried out with an explicit Euler method, employing a time step of $\Delta t = 0.05$ seconds. The initial conditions were set to $h_1 = h_2 = 2$, and the damping parameter was $\mu = 1$.

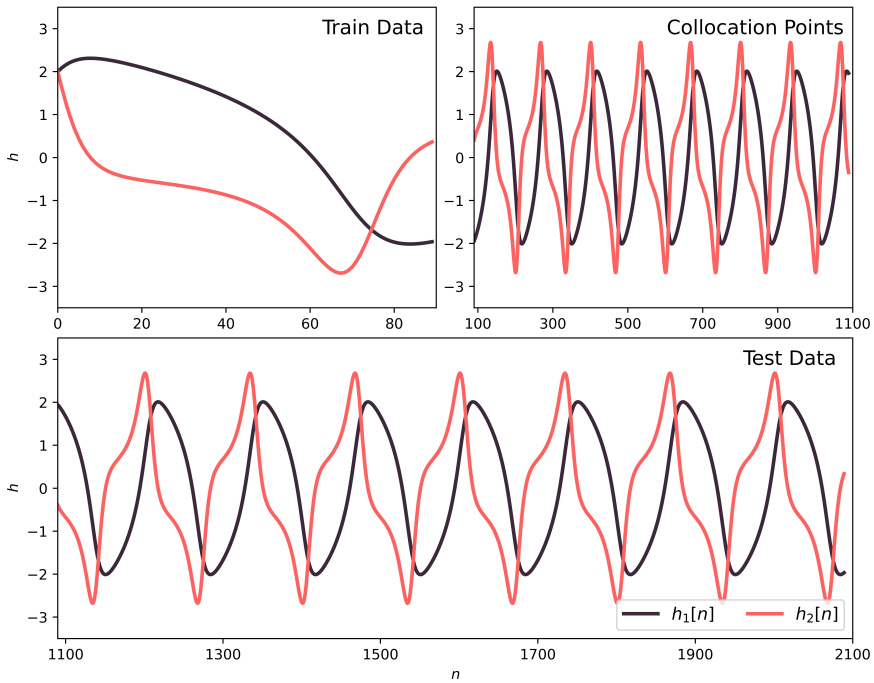
The simulation schematic, shown in Figure 20, includes distinct datasets: training data, collocation points region, and the subsequent test dataset for evaluating ESN and PI-ESN.

- **Training Data Region**: This is the initial phase of the simulation. Training data is generated by simulating the behavior of the Van der Pol oscillator using the given equations and initial conditions. This data is used to train the conventional ESN.
- **Collocation Points Region**: During this phase, the simulation introduces the concept of physics regularization. In the context of traditional ESN, no additional training or utilization of extra information takes place within this region. The ESN generates outputs exclusively based on the training conducted with historical data. In practice, this area can serve as a test dataset for conventional ESNs. In contrast, the Physics-Informed Echo State Network (PI-ESN) operates uniquely within this region. It applies physics regularization without the need for any additional data.

This means that no extra information or datasets are introduced. Instead, physics regularization is employed, as depicted in Figure 17, to ensure that the ESN conforms to the underlying physical principles governing the Van der Pol oscillator. This approach enhances the PI-ESN accuracy and adherence to the known physics without relying on supplementary data.

- **Test dataset**: After the implementation of physics regularization, this phase assesses the system's performance after the physics training. It evaluates the system's behavior and its adherence to the governing equations, even when subjected to conditions beyond those encountered during training or within the collocation points region. This region demonstrates how the physics regularization applied in the collocation points region extends its influence to test dataset where collocation points are not employed.

Figure 20 – Simulation of the unforced Van der Pol oscillator system.



Source: Author

5.1.1.2 PI-ESN settings

For the experiments reported, the proposed PI-ESN was trained in two stages: 1) first pretraining an ESN by conventional ridge regression and optimizing its main hyperparameters; and 2) subsequently refining this ESN by applying the physics-informed method.

The ESN was trained using the parameters described in Table 1 [1] with 90 training

[1] The spectral radius value was chosen to effectively illustrate the enhancement resulting from the implementation of physical regularization. Additional experiments were conducted with values in proximity to 1, showcasing improvements that, albeit present, were less visually pronounced.

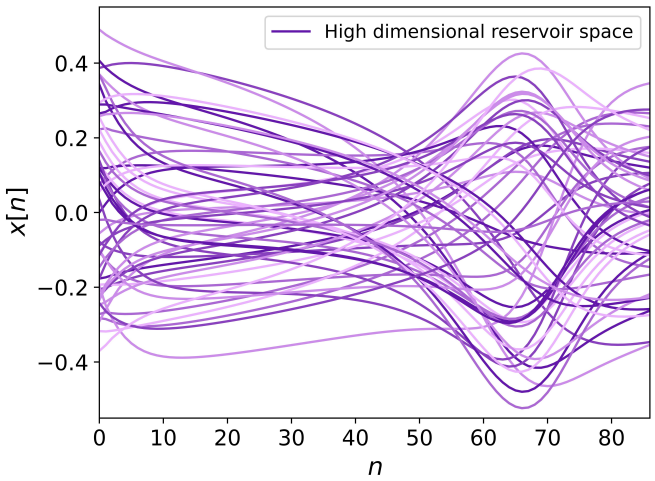
Table 1 – ESN's parameters training for Van der Pol unforced system.

Parameter	Value
α	1
δ_{in}	0
δ_{fb}	0.15
δ_b	0
γ	10^{-5}
$\rho(\mathbf{W})$	0.4
N_x	50

Source: Author

points illustrated in Figure 20. The ESN architecture does not employ bias weights, input weights (since the system being modeled is autonomous) and is non-leaky. A warm-up was applied to the first 5 states to discard unstable states. The state reservoir is depicted in Figure 21. This reservoir remains constant throughout the optimization conducted by the PI-ESN.

Figure 21 – ESN reservoir $\mathbf{X}$ for unforced Van der Pol system.



Source: Author

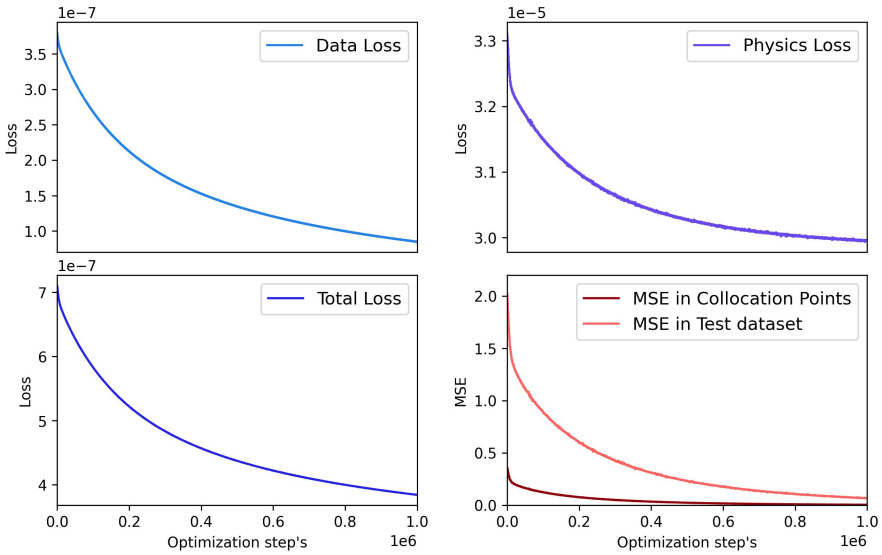
After the linear regression for the calculation of $\mathbf{W}^{out}$, 1000 collocation points were employed for the physics loss function. For the PI-ESN training, λ_{phy} was set to 0.01, and λ_{data} was set to 1. The implementation was conducted using the PyTorch framework, and the ADAM optimizer was employed to update the value of $\mathbf{W}^{out}$. The learning rate for the optimizer was set to 10^{-5}.

5.1.1.3 PI-ESN results

In Figure 22, the data loss, physics loss, total loss and MSE during the PI-ESN training are shown. A consistent reduction is observed during the training. The data, physics, and total loss continuously decrease, reflecting the model's improvement in fitting the data and adhering to physics laws. This is confirmed with the MSE calculated for the collocation points

region and test dataset, demonstrating the model's increasing accuracy in generalizing to new data.

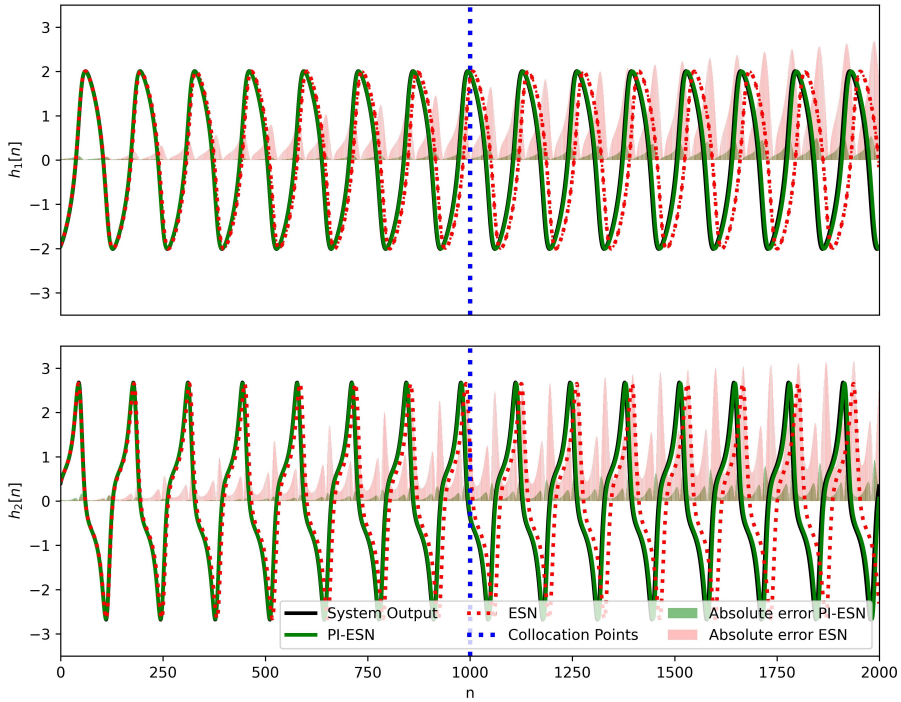
Figure 22 – Loss function and Error evolution during PI-ESN training for unforced Van der Pol system.



Source: Author

The prediction of the PI-ESN for the unforced Van der Pol oscillator after training is shown in Figure 23.

Figure 23 – PI-ESN and ESN prediction for unforced Van der Pol system.

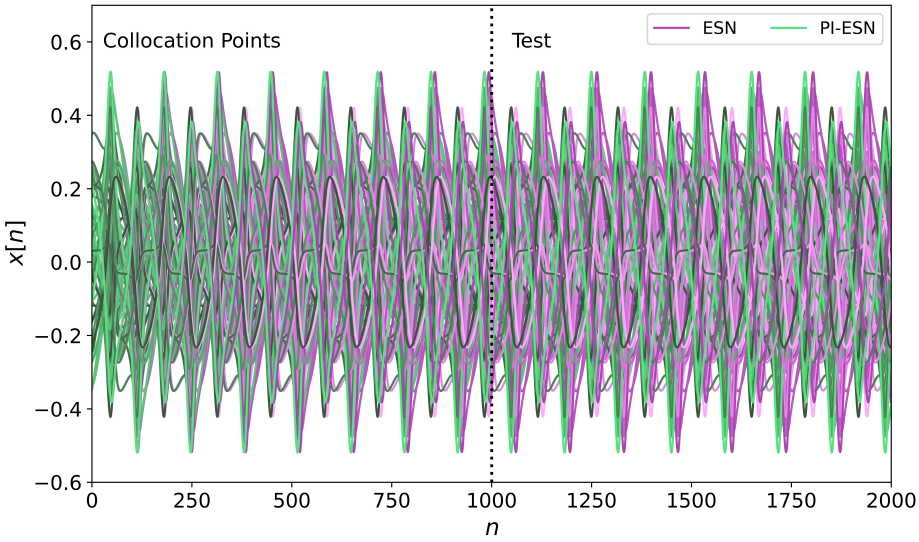


Source: Author

In the background, it is possible to observe the absolute error of the ESN and PI-ESN with the actual system output. This prediction refers to the collocation points and test data of the system presented in Figure 20. The MSE values in the collocation points region were 0.3575 for the ESN and 0.0054 for the PI-ESN. In the test dataset, the corresponding MSE values were 2.0209 and 0.0673, respectively.

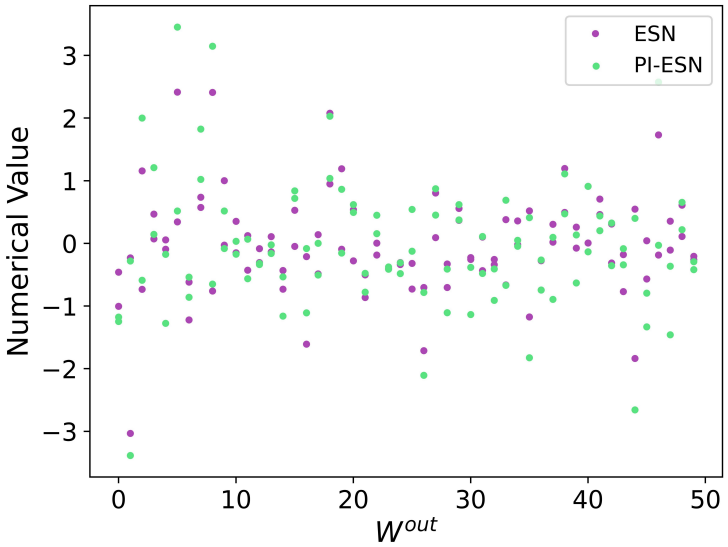
The new calculated states for the collocation points and test dataset after the conventional ESN training and PI-ESN training are displayed in Figure 24, along with the new $\mathbf{W}^{out}$ shown in Figure 25.

Figure 24 – Evolution of states after training PI-ESN and ESN.



Source: Author

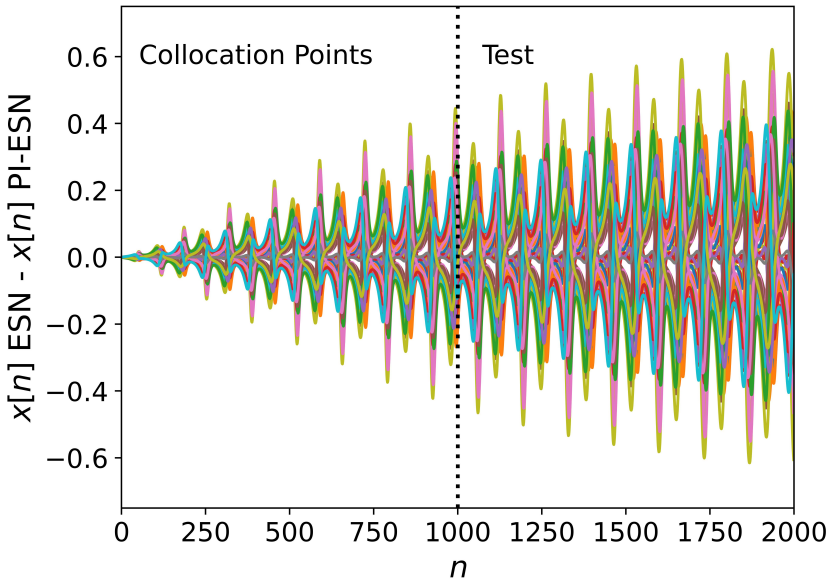
Figure 25 – Comparison of $\mathbf{W}^{out}$ values between conventional ESN and PI-ESN.



Source: Author

It is possible that the difference between the states of ESN and PI-ESN results in increasing divergence in their respective output calculations over time (Figure 26). This divergence arises from the recurrent nature of state propagation inherent in the neural network architecture, gradually separating the response from the system's output. In this example, changes in network states, combined with adjustments to the $\mathbf{W}^{out}$ weights, have enhanced the neural network's response, benefiting from the application of physics laws during training.

Figure 26 – Difference between the states $x[n]$ calculated by ESN and PI-ESN.



Source: Author

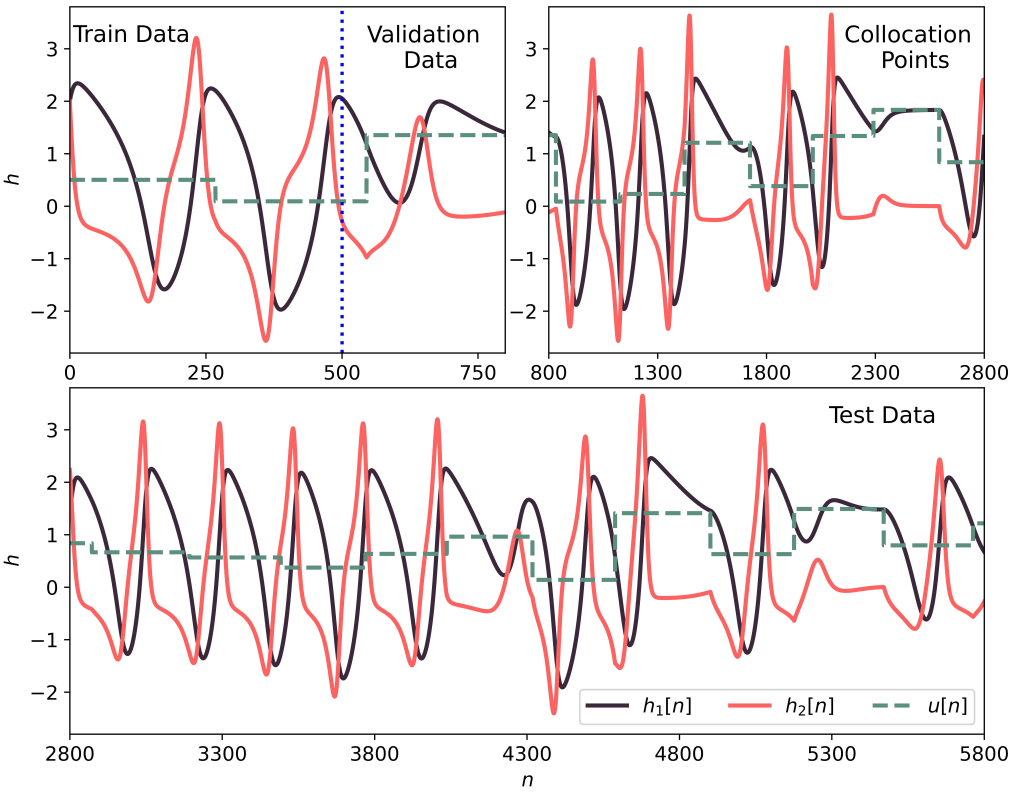
5.1.2 Forced Equation

5.1.2.1 Dataset

The Van der Pol oscillator's dynamic system was generated using Equation (40) through an explicit Euler method with a time step of $\Delta t = 0.03$ seconds. Initial values of $h_1 = h_2 = 2$ and $\mu = 1$ were employed. In this case, an input signal was required, and it was generated using Amplitude Modulated Pseudo-Random Bit Sequences (APRBS) with an amplitude ranging from 0 to 2 and a signal variation occurring every 500 to 800 time steps. Figure 27 provides an illustrative example of a simulated Van der Pol system, presented in three plots.

In the first plot, the training set N_{te} of 500 time steps and the validation set N_{ve} of 300 time steps are separated by a dashed blue line, which was used for hyperparameter search during ESN training. This plot encompasses the entire training set, totaling $N_t = 800$ points, with a warm-up of 50 time steps. The second plot represents the region where physics-informed training occurs, involving 2000 collocation points. In this region, only the random

Figure 27 – Simulation of the forced Van der Pol oscillator system



Source: Author

input values u at the N_f collocation points are used for regularization purposes, without employing the labels h_1 and h_2.

This highlights the distinction from the case without external input (unforced equation). While input values are necessary in the collocation point region, these values can be predetermined, such as a constant or specific input value not used in conventional network training. To evaluate the performance of both ESN and PI-ESN in the region where physics-based training occurs, a random signal was employed. Lastly, the third plot illustrates the test dataset consisting of 3000 points, which is utilized for the analysis of the PI-ESN.

5.1.2.2 PI-ESN-a settings and results

A grid search was conducted to explore the values of input scaling (δ_{in}) and feedback scaling (δ_{fb}) within the range of 0.05 to 0.95, with increments of 0.05. Additionally, the Tikhonov regularization factor γ was investigated over magnitudes ranging from 10^{-2} to 10^{-7}, using a resolution of 10^{-1}. The values for N_x, α and $\rho(\mathbf{W})$ remained constant during the optimization. The values found are available in Table 2.

The resulting ESN was then used as the initial guess for the PI-ESN-a. The optimization was carried out using the L-BFGS algorithm implemented in the TensorFlow framework. The PI-ESN-a training process is illustrated in Figure 28, where the data and physics loss values,

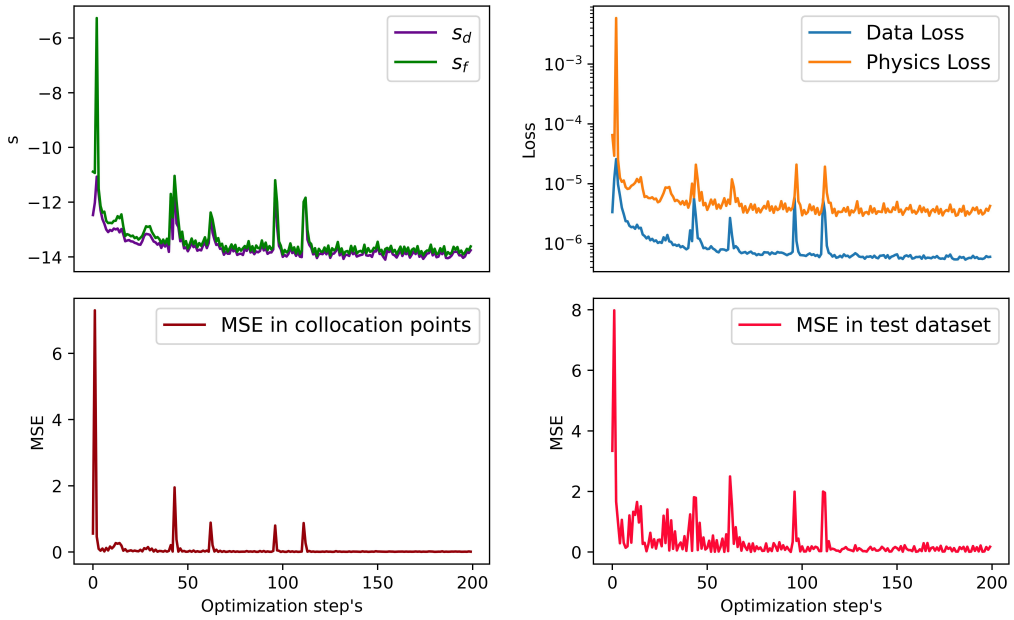
MSE for the collocation points, MSE for the test dataset, and the adaptive parameters s_f and s_d are displayed. These results correspond to the simulated system illustrated in Figure 27.

Table 2 – Parameters for ESN training: Van der Pol Forced System

Parameter	Value
α	1
δ_{in}	0.1
δ_{fb}	0.1
δ_b	0
γ	10^{-7}
$\rho(\mathbf{W})$	0.8
N_x	200

Source: Author

Figure 28 – Evolution of the adaptive weights (s_d, s_f), the loss functions (J_{data}, J_{phy}) and the MSE during the physics training of the PI-ESN for the forced Van der Pol system.



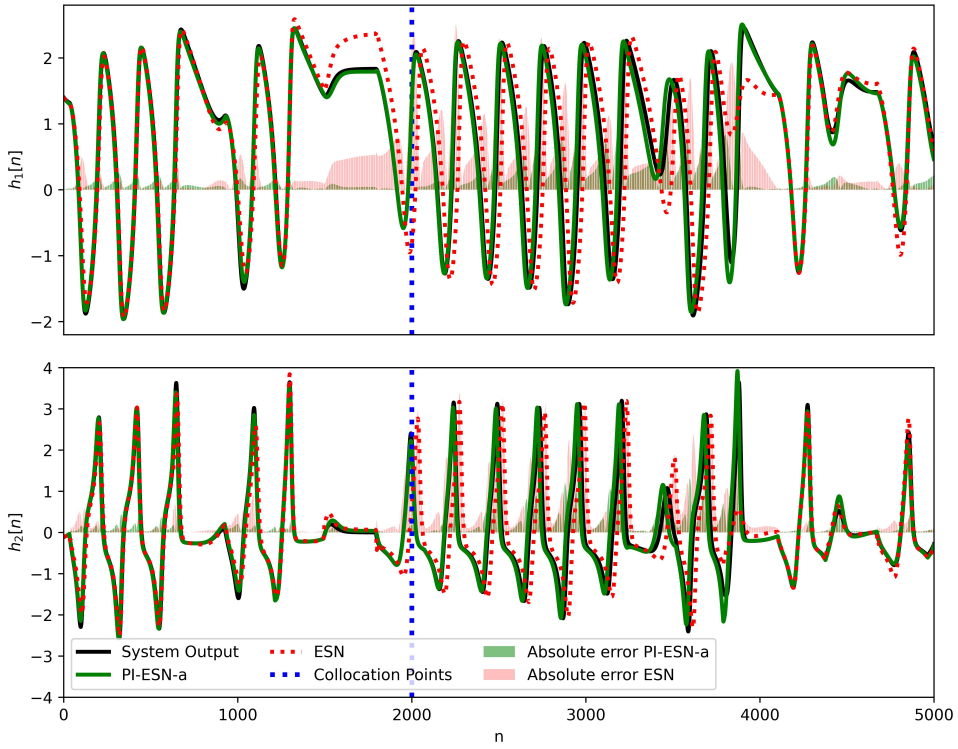
Source: Author

By analyzing the different graphs, it is evident that there was a reduction in both physics and data loss functions, as well as errors in the test dataset and collocation points. This improvement helps the model adhere more closely to the physics equations, similar to what is observed in the unforced Van der Pol case. The J_{data} and J_{phy} values at the end of the experiment are 6×10^{-7} and 4.25×10^{-6}, respectively. Moreover, the final values of the $[s_d,s_f]$ parameters are $[-13.75, -13.61]$, this implies that approximately 53.5% of the data loss function's value and 46.5% of the physics loss function's value contributed to the calculation of the total loss function. However, it is essential to clarify that these percentages do not directly reflect the contribution to the final value of the total loss function, due to

the inherent difference in magnitude between the physics and data loss functions. Rather, these values denote the proportional weight by which each function multiplies their respective contributions.

The evaluation of the results for h_1 and h_2 outputs is presented in Figure 29. The blue dashed vertical line splits the region between collocation points (left) and test dataset (right). In the background, it is possible to observe the absolute error of the ESN and PI-ESN with the actual system output.

Figure 29 – Prediction of the PI-ESN-a for Van der Pol oscillator after training.



Source: Author

This prediction refers to the collocation points and test data of the system presented in Figure 27. The MSE for the collocation points region was found to be 0.1204 for the ESN and 0.0101 for the PI-ESN-a. In the test data, the corresponding MSE values were 0.8601 and 0.1741, respectively.

The achieved results demonstrate that training with physics laws can adjust the behavior of the ESN for data that are not used in training. It is important to highlight that physics training does not use output data from collocation points region, only the input values.

5.1.2.3 Comparative Analysis: PI-ESN vs. PI-ESN-a vs. ESN

This experiment consists of validating the proposed PI-ESN in a limited training data scenario. Table 3 shows the MSE for the collocation points and the test set, for three ESN architectures: conventional ESN, PI-ESN, and PI-ESN-a. For each case, the performance

was averaged over five different randomly generated ESNs and input values. $N_{te}, N_{ve}, N_t = [500, 300, 800]$ was employed for ESN training. The ESN was constructed with $N_x = 100$, $\alpha = 1$, $\rho(\mathbf{W}) = 0.8$ and a warm-up of 50 time steps. For each run, a hyperparameter search was conducted to determine suitable values for δ_{in}, δ_{fb}, and γ with a grid search optimization, while N_x, α and $\rho(\mathbf{W})$ remained constant.

Table 3 – Forced Van der Pol System: Average MSE for ESN and PI-ESN architectures.

Architecture	Collocation Points	Test
ESN	1.5217	1.5485
PI-ESN	0.1967	0.2575
PI-ESN-a	**0.1161**	**0.1912**

Source: Author

The PI-ESN-a was initiated with $s_d, s_f = [1,1]$, and for the PI-ESN, the parameters were set to $\lambda_{data} = \lambda_{phy} = 1$. $N_f = 2000$ collocation points were utilized for both PI-ESN and PI-ESN-a during physics-informed training. Mean squared error was calculated for the collocation points region with 2000 points and for the test dataset containing 3000 points.

Table 3 shows that the mean squared error is reduced when employing the adaptive architecture. On average, it achieves superior results compared to the PI-ESN and the ESN. It can be observed that the PI-ESN-a reduces the MSE of the ESN by an average of 89.5%, while the PI-ESN reduces it by an average of 84.8% compared to the ESN.

5.1.2.4 Effect of reservoir size and train data size

Table 4 displays the mean squared error concerning variations in the reservoir size. For this experiment, 5 neural networks were employed for each reservoir size, spanning across the 6 randomly selected inputs. The spectral radius, training size, collocation points, and test size are consistent with those used in the experiment detailed in Table 3. Each presented value was obtained by averaging the MSE from approximately 30 experiments. A total of 120 runs were conducted for this experiment, with 22 runs exhibiting training instability and subsequently being excluded from the final analysis presented in the table. The analysis demonstrates that the mean squared error consistently maintains a similar magnitude across different reservoir sizes. This observation can be attributed to the inherent error introduced by the explicit Euler approximation utilized in the physics training process, as emphasized in Racca and Magri (2021).

It is also important to notice that a traditional ESN network can make predictions with low error, but in such cases, a larger amount of data would be required. To illustrate this difference between PI-ESN-a and traditional ESN, Figure 30 presents an experiment in which the Van der Pol oscillator system is executed with varying sizes of the training set (N_t). The presentation is based on six runs with different random seeds influencing the initial ESN weights, while the external signal maintains a constant seed for generating consistency.

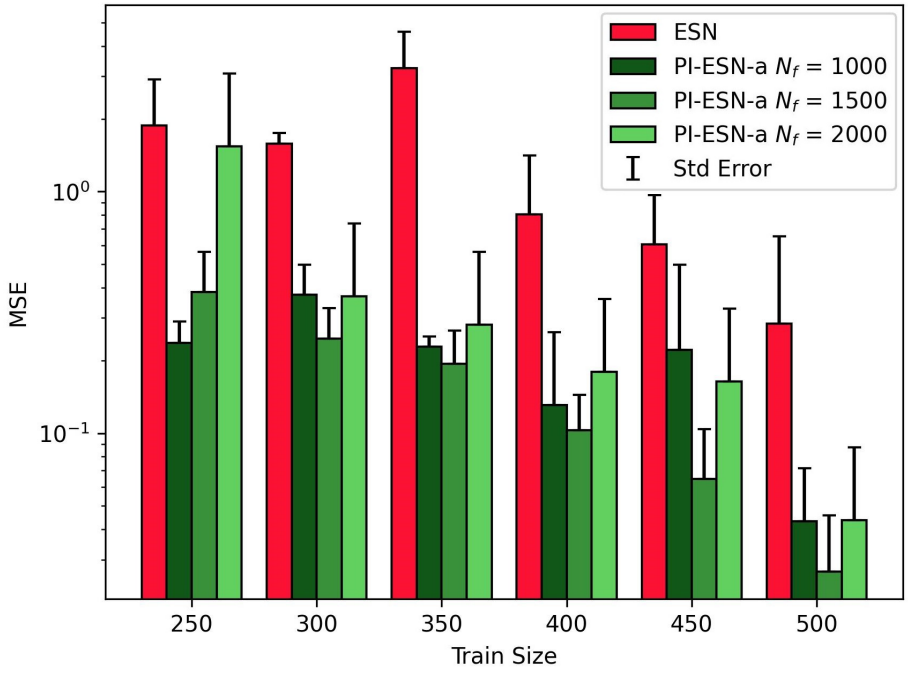
Table 4 – Average MSE for the conventional ESN and the PI-ESN-a as a function of the reservoir size (N_x) for the Forced Van der Pol system.

Reservoir Size	Collocation Points		Test	
	ESN	PI-ESN-a	ESN	PI-ESN-a
100	1.255	0.346	1.758	0.435
200	0.873	0.400	1.230	0.481
300	**0.514**	**0.274**	**0.815**	**0.346**
400	1.373	0.450	1.762	0.614

Source: Author

The results include the MSE of the ESN, allowing for comparative analysis with physics-based training for different collocation point sizes ($N_f = [1000, 1500, 2000]$). The evaluation of the mean squared error is performed over a duration of 2100 time steps. The average MSE of the PI-ESN-a is plotted against the size of the labeled training dataset, N_t, for the three different cases of N_f.

Figure 30 – The average MSE of the PI-ESN-a shown for different values of N_t and N_f.



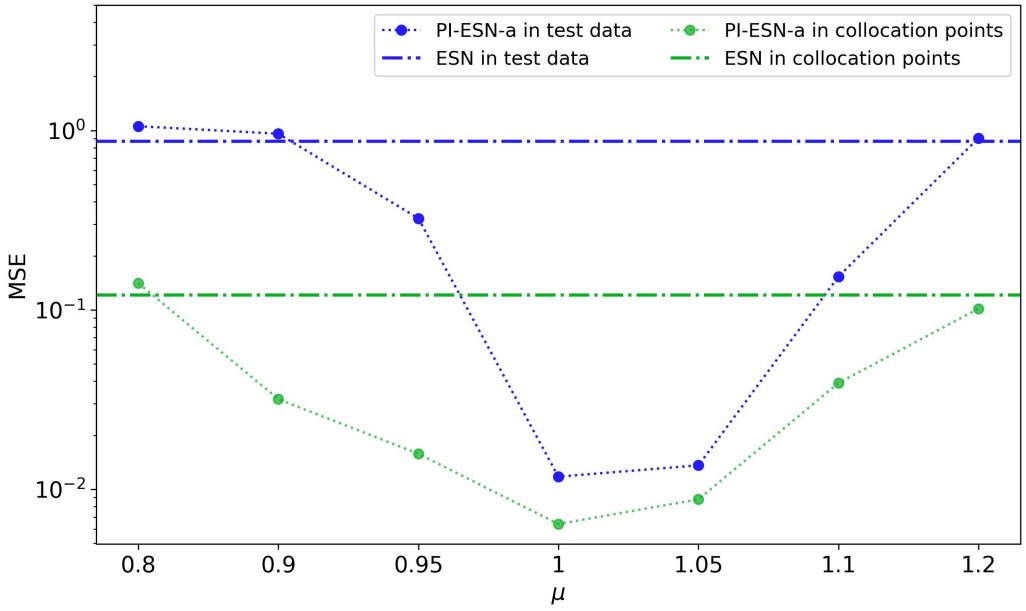
Source: Author

In Figure 30, it is noticeable that for small amounts of data, the error and standard deviation of the PI-ESN-a are lower than those of the ESN. As the quantity of data increases, the ESN's performance improves, resulting in a decrease in the error value. By evaluating the number of collocation points, it is evident that the error and standard deviation of the PI-ESN-a are not proportional to the number of collocation points. When using a training dataset with $N_f = 2000$, the error and standard deviation are found to be higher compared to the cases when 1000 or 1500 points are utilized.

5.1.2.5 PI-ESN-a's robustness to parameter model uncertainty

In real-world scenarios, it is common to observe disparities between the model and the actual system dynamics. To evaluate the network's robustness, a parametric uncertainty was introduced by manipulating the parameter μ, which directly impacts the oscillation behavior. As shown in Figure 19, the parameter was exclusively altered within the physics function utilized during the physics training process, while the system data was kept at the reference value of $\mu = 1$. The results for the disturbed damping parameter (μ) from Equation (40), which were used in the physics-based loss for training the PI-ESN, are presented in Figure 31. The parameters of the adaptive network were the same as those presented in the previous experiment concerning Figure 28 and Figure 29. The MSE is displayed for the collocation points region, test set, and total dataset (as shown in the data split Figure 27) for the PI-ESN-a. The horizontal lines represent the constant MSE values of the ESN for the collocation points region and test set, as the damping parameter alteration only affects the calculation of the physics function (Equation (29)).

Figure 31 – Adaptive PI-ESN training with parametric error in μ



Source: Author

By disturbing the parameter μ, it is possible to observe variations in the network's error. When disturbance is too strong, i.e., μ exceeds 1.1 or falls below 0.95, the test error (blue dots) of PI-ESN-a is slightly worse than that of ESN (horizontal blue line). However, for disturbances resulting in $\mu \in [0.95, 1.1]$, the proposed PI-ESN-a continues to improve and regularize ESN's prediction, as seen by the dots below their respective horizontal lines of the same color. Consequently, this outcome highlights the capability of the PI-ESN to achieve lower error rates than the ESN, despite the presence of a parametric error associated with the μ variable in the physics equation, as well as errors arising from the derivative calculated using the explicit Euler method.

5.2 FOUR-TANK SYSTEM

The Four-Tank system consists of interconnected tanks with two pumps that can be used to control the flow rate into the tanks. In this process, it is desired to control the levels of the tanks by manipulating the voltages applied to pumps (Alvarado et al., 2006; Johansson, 2000). The determination of the nonlinear model parameters for the Four-Tank system is critical in order to develop and implement an effective control strategy. Therefore, accurately identifying the system dynamics is important for the application of predictive controllers. The process is characterized by the following system of differential equations:

$$\begin{aligned}
\frac{dh_1(t)}{dt} &= -\frac{a_1}{A_1}\sqrt{2gh_1(t)} + \frac{a_3}{A_1}\sqrt{2gh_3(t)} + \frac{\gamma_1 k_1}{A_1}V_1(t) \\
\frac{dh_2(t)}{dt} &= -\frac{a_2}{A_2}\sqrt{2gh_2(t)} + \frac{a_4}{A_2}\sqrt{2gh_4(t)} + \frac{\gamma_2 k_2}{A_2}V_2(t) \\
\frac{dh_3(t)}{dt} &= -\frac{a_3}{A_3}\sqrt{2gh_3(t)} + \frac{(1-\gamma_2)k_2}{A_3}V_2(t) \\
\frac{dh_4(t)}{dt} &= -\frac{a_4}{A_4}\sqrt{2gh_4(t)} + \frac{(1-\gamma_1)k_1}{A_4}V_1(t)
\end{aligned} \tag{41}$$

where the variable h_i denotes the level of each tank i, and V_1 and V_2 represent the voltage applied to the pumps. The cross-sectional area of each tank and the cross-sectional area of the bottom orifice are represented by A_i and a_i, respectively. The constants k_1 and k_2 relate the flow rate to the applied voltage in the pump, while the valves that have fixed openings are denoted by γ_1 and γ_2. The corresponding values for each parameter, along with their units, are presented in Table 5.

Table 5 – Model parameters of the Four-Tank system.

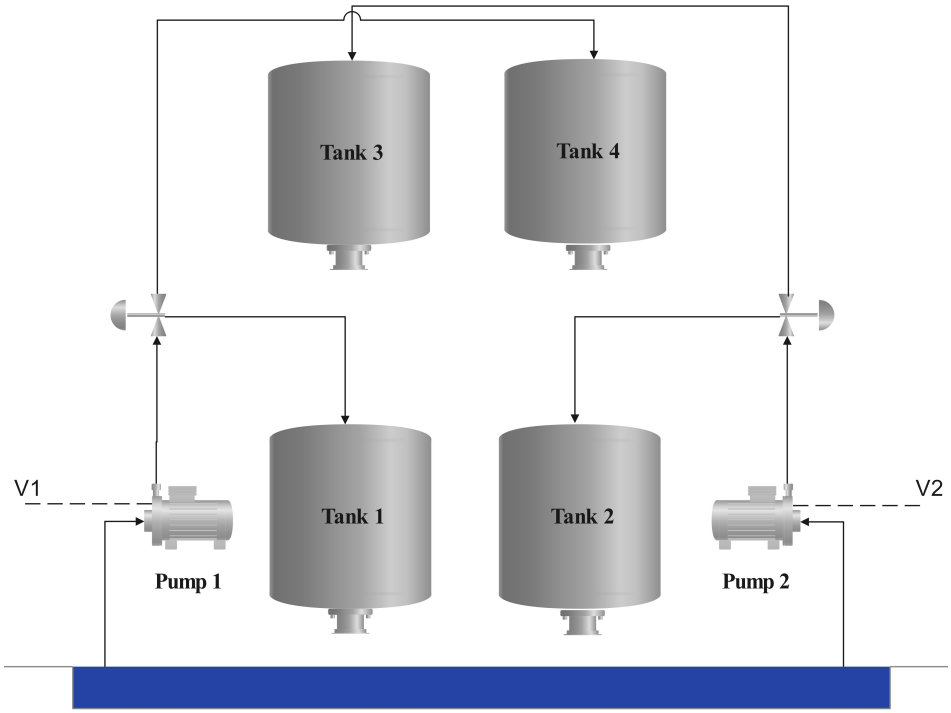
Parameter	Value	Unit
A_1, A_3	28	cm^2
A_2, A_4	32	cm^2
a_1, a_3	0.071	cm^2
a_2, a_4	0.071	cm^2
g	981	$cm^2.s^{-2}$
k_1, k_2	1	$cm^3.V^{-1}.s^{-1}$
γ_1	0.7	
γ_2	0.6	

Source: Author

5.2.1 Dataset

The Four-Tank system was generated with Equation (41) using an explicit Euler method with a time step $\Delta t = 1$ sec, initial values of $h_1 = h_2 = h_3 = h_4 = 2$. The input signal was generated using an APRBS with an amplitude ranging from 0 to 5 and a signal oscillation of

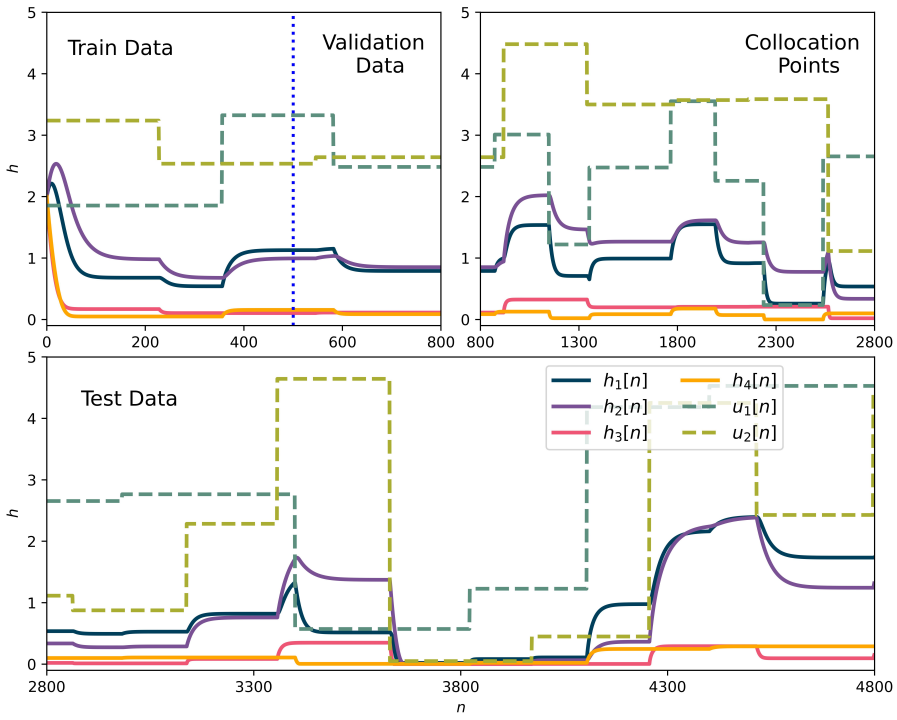
Figure 32 – Nonlinear process to control the water levels in a Four-Tank system.



Source: Author

100 to 200 time steps. The regions where conventional training, physics-informed training, and testing occur are represented Figure 33.

Figure 33 – Simulated Four-Tank systems



Source: Author

The first plot displays the training set N_{te} and the validation set N_{ve}, separated by a dashed blue line used for hyperparameter tuning during the training of the ESN. The first plot corresponds to the total number of points in the training set $N_t = 800$. The second plot represents the region where physics-informed training is performed using 2000 collocation points. In this region, only the random input values u at the N_f collocation points are utilized for regularization purposes, while the labels h_1, h_2, h_3 and h_4 are not used. Lastly, the third plot illustrates the test dataset consisting of 2000 points, which is utilized for the analysis of the PI-ESN.

5.2.2 PI-ESN-a settings and results

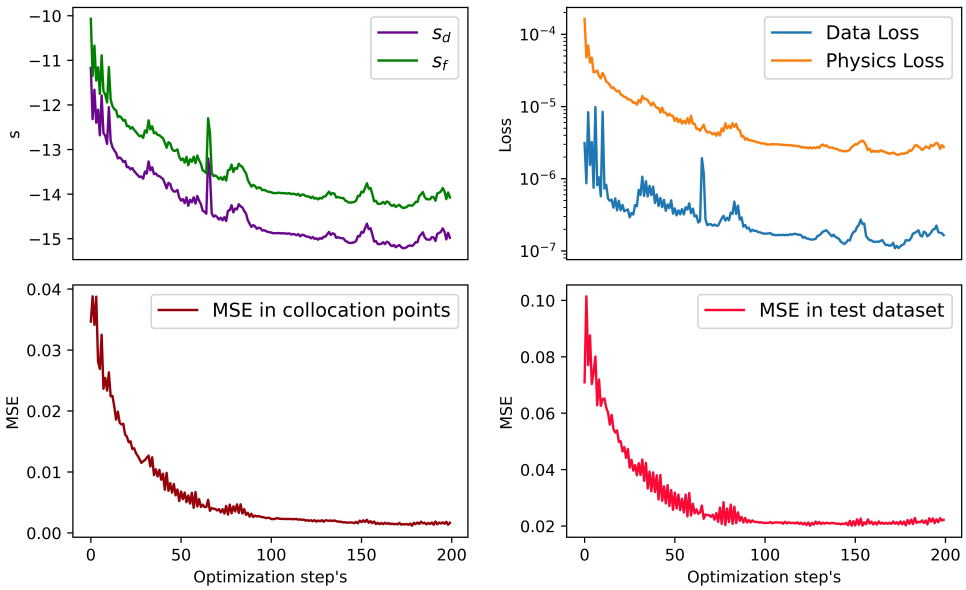
The ESN employed for the simulated system illustrated in Figure 33 was generated with the following parameters: $N_x = 400$, $\alpha = 1$, and $\rho(\mathbf{W}) = 0.8$, $\delta_b = 0$ and a warm-up of 50 time steps. The grid search was carried out similarly to the Van der Pol experiments, with a training set (N_{te}) of 500 time steps and a validation set (N_{ve}) of 300 time steps. The ESN was then retrained using a total of $N_t = N_{te} + N_{ve} = 800$ time steps using the values $\delta_{fb} = 0.2$, $\delta_{in} = 0.1$, $\gamma = 10^{-5}$ found in the grid search. Subsequently, the resulting ESN was used as the initial approximation for the PI-ESN.

The PI-ESN-a training process is presented in Figure 34, which displays the data and physics loss values, the mean squared error at the collocation points and test dataset, as well as the adaptive parameters s_f and s_d. Throughout the process, the data loss and physics loss decrease, enhancing the PI-ESN-a model's capability to fit the data and adhere to the principles of physics. At the end of the experiment, the J_{data} and J_{phy} values were found to be 1.65×10^{-7} and 2.76×10^{-6}, respectively. The final values of the $[s_d, s_f]$ parameters were determined to be $[-14.97, -14.07]$, this implies that approximately 71.1% of the data loss function and 28.9% of the physics loss function contributed to the calculation of the total loss function. As discussed previously, these values represent the proportional weights of each function.

The evaluation of the results for the tank levels outputs is presented in Figure 35, where the blue dashed vertical line splits the region between collocation points (left) and test set (right). In the background, it is possible to observe the absolute error of the ESN and PI-ESN with the actual system output. This prediction refers to the collocation points and test data of the system presented in Figure 33. The MSE for the collocation points region was found to be 0.0468 for the ESN and 0.0016 for the PI-ESN-a. In the test dataset, the corresponding MSE values were 0.1266 and 0.0221, respectively.

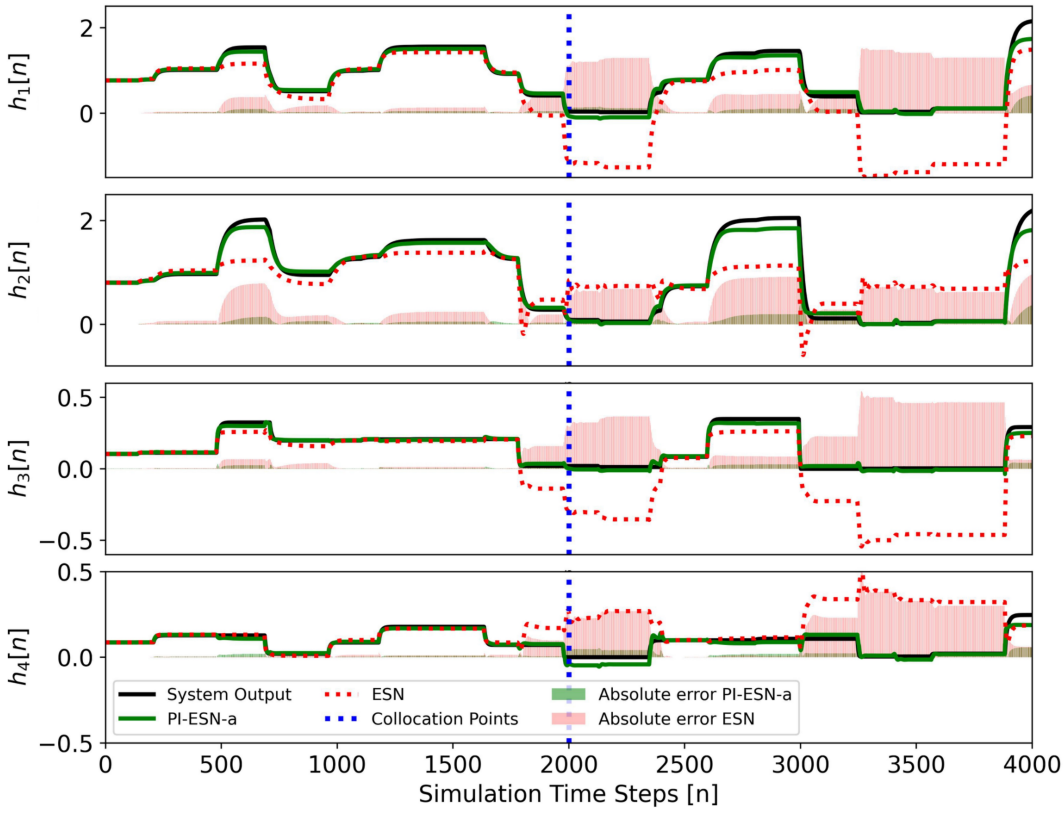
It can be seen that the PI-ESN-a prediction performance is significantly improved and consistently over the initial ESN, not only the collocation points (left area of the dashed vertical blue line), but also on new points in the test set which were generated by random inputs. Thus, for small data regimes, physics-informed training of the ESN is relevant and useful if physical laws are available.

Figure 34 – Evolution of the adaptive weights (s_d, s_f), the loss functions (J_{data}, J_{phy}) and the MSE during the physics training of the PI-ESN for four tank system.



Source: Author

Figure 35 – Prediction of the PI-ESN-a for Four-Tank system after training.



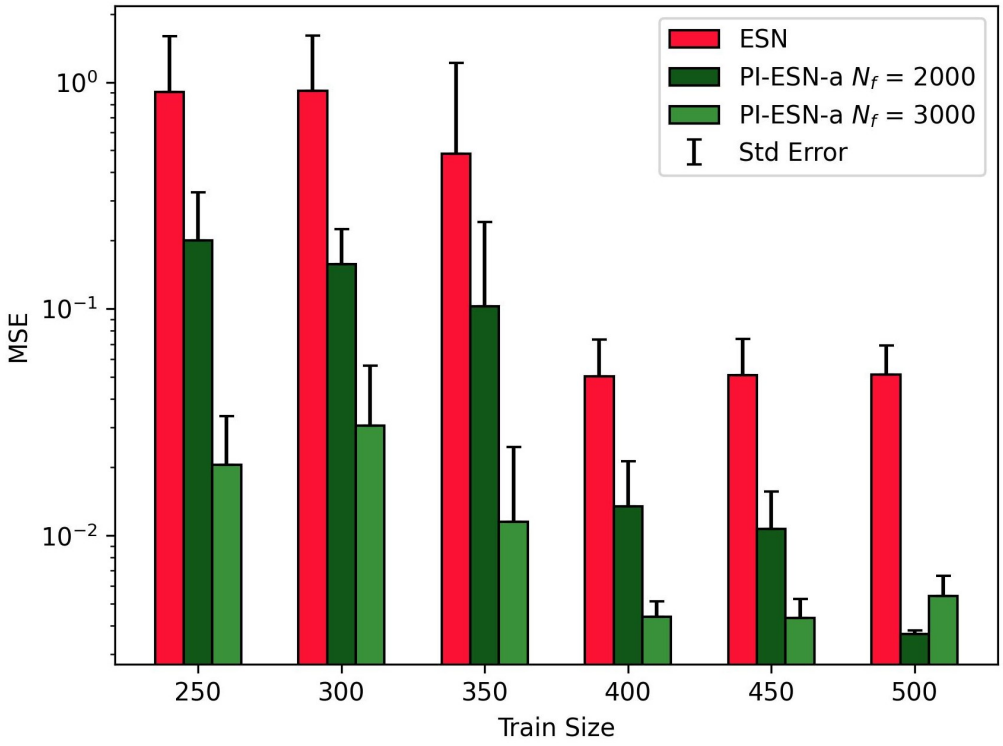
Source: Author

5.2.3 Effect of reservoir size and train data size

To demonstrate the performance of PI-ESN-a relative to traditional ESNs, in small data regime, an experiment was conducted as depicted in Figure 36. The experiment involved executing the Four-Tank system for various training set sizes (N_t). Specifically, ESNs were run for each of the six training dataset sizes (250, 300, 350, 400, 450, 500 while employing physics training for different collocation point sizes ($N_f = [2000, 3000]$).

The evaluation of mean square error was carried out over 4000 time steps. For each run, a hyperparameter search was conducted to determine suitable values for δ_{in}, δ_{fb}, and γ with a grid search optimization. The results also include the MSE of a traditional trained ESN for comparison. Standard deviation is shown for 5 runs with different random seeds, which affect the ESN's initial weights, but not the control signal, which remains fixed.

Figure 36 – MSE values for the PI-ESN-a were computed for various N_t sizes, considering both $N_f = 2000$ and $N_f = 3000$.



Source: Author

Similarly to the results presented by the oscillator experiment, the PI-ESN-a was able to reduce the mean square error of the system even in cases where there is a reduced amount of data. It can be observed that the ESN gradually improves the performance as the amount of data increases, but PI-ESN-a was able to reduce the MSE of the original ESN for all training set sizes. With more data, ESN reduces its prediction error, but so does PI-ESN-a, which shows the potential of the proposed physics-informed ESN training in small data regimes.

Table 6 displays the average MSE concerning variations in the reservoir size. For this experiment, 5 neural networks were employed for each reservoir value, considering the 4

randomly selected inputs. Between parenthesis, the error reduction percentage of PI-ESN-a over the respective ESN is shown. The spectral radius, training size, collocation points, and test size mirror those used in the experiment detailed in Figure 35. A total of 60 runs were conducted for this experiment, with 6 runs experiencing training instability and subsequently being excluded from the final analysis presented in the table. Each value is obtained by averaging the MSE of around 20 experiments.

The analysis indicates that the mean squared error consistently demonstrates a comparable magnitude across various reservoir sizes, aligning with the behavior observed in the Van der Pol oscillator experiment. This behavior can be attributed to the inherent error introduced by the explicit Euler approximation utilized during the physics training process. To address this error, including the instability in physics-informed training, an additional possibility is to employ automatic gradient techniques.

Table 6 – Average MSE for the conventional ESN and the PI-ESN-a with different values of reservoir size (N_x) for the four tank system.

Reservoir	Collocation Points		Test set	
Size	ESN	PI-ESN-a	ESN	PI-ESN-a
200	0.415	0.032 (-92%)	0.504	0.057 (-87%)
400	0.289	**0.028** (-90%)	0.510	**0.044** (-91%)
800	0.366	0.101 (-72%)	0.465	0.094 (-80%)

Source: Author

5.3 ELECTRIC SUBMERSIBLE PUMP

Oil is found in porous/fractured and permeable rock formations, and the location is referred to as an oil reservoir, which typically has high pressure. Impermeable rocks usually isolate the reservoir. The fluids (hydrocarbons and water) contained within the reservoir require a certain amount of energy to overcome the physical resistance of the porous channels and migrate to the production units on the surface (Filho, 2011).

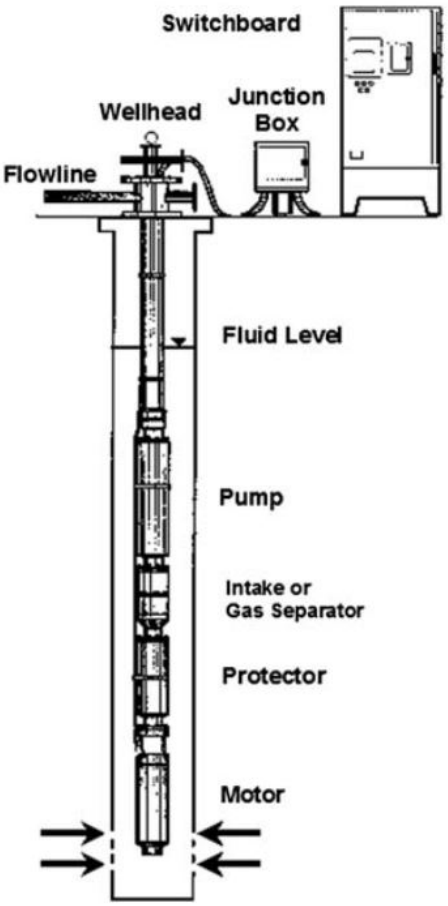
An oil reservoir can have multiple extraction wells. When referring to wells, it typically means the equipment necessary to transport gas and oil from the reservoir to the production facility. Initially, when production begins, there is a natural pressure within the oil reservoir, known as primary energy, which results from all geological processes the deposit underwent during its formation (Thomas, 2004). However, as production continues, this energy decreases due to the decreasing pressure in the well, leading to financial losses and potentially making oil extraction economically impracticable. Therefore, various methods of artificial lift are employed to extend the production life of wells (Filho, 2011).

One of these methods is the Electric Submersible Pump (ESP) shown in Figure 37. The electric submersible motor is located at the unit's base and is cooled by the surrounding

well stream. It connects to the protector section, which serves essential functions for safe unit operation (Takacs, 2018).

The pump intake or gas separator allows well fluids to enter the centrifugal pump while simultaneously removing quantities of gas from the well stream. The separated gas rises through the liquid column in the casing annulus, reaching the casing head, from where it is directed into the flow line. The connection between the casing head and the flow line maintains a constant fluid level above the submersible pump. The multistage centrifugal pump, at the core of the ESP system, lifts the liquid to the surface. Produced fluids flow through the tubing string to the surface, where a wellhead facilitates the introduction of the electric cable into the well. On the surface, the equipment for supplying power consists of a junction box where electric cables are connected and a switchboard that offers measurement and control functions (Takacs, 2018).

Figure 37 – Conventional ESP installation



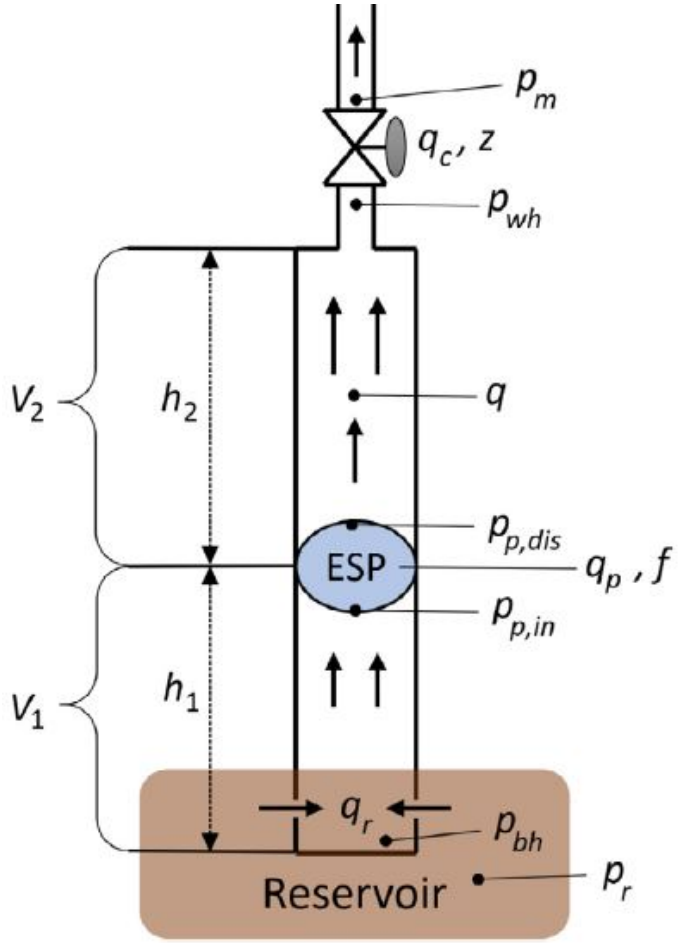
Source: Takacs (2018)

ESPs are primarily employed in high-production oil wells, both offshore and onshore, due to their proficiency in extracting substantial liquid volumes. However, they come with notable drawbacks, encompassing high implementation, operational, and maintenance costs, as well as susceptibility to gas presence at the suction inlet and potential lifespan reduction

due to solid materials like sand. Despite these challenges, ESPs offer advantages such as efficient liquid extraction from moderately deep wells, high operational efficiency exceeding 1000 barrels per day (bpd), adaptability to deviated well configurations, and the potential for low maintenance when operated correctly. Furthermore, their compact design renders them well-suited for offshore installations (Hernes, 2020).

The ESP modeling employed in this study is illustrated in Figure 38, which is based on the work of Jean P. Jordanou et al. (2022), Hernes (2020) and Binder, Pavlov, and Johansen (2015). Table 7 shows the model variables. The model outputs are the bottom hole pressure p_{bh}, wellhead pressure p_{wh} and average flow rate q, whereas the inputs are the ESP frequency f and choke valve opening z.

Figure 38 – ESP Model System



Source: Hernes (2020) and Binder, Pavlov, and Johansen (2015)

The DAE system is described by the differential part shown in Equation (42) and the algebraic part in Equations (43) and (44):

Table 7 – Variables Description for ESP model

Symbol	Description
q	Average liquid flow rate
q_r	Flow rate from reservoir into the well
q_c	Flow rate through production choke
p_m	Production manifold pressure
p_{wh}	Wellhead pressure
p_{bh}	Bottom hole pressure
$p_{p,in}$	ESP intake pressure
$p_{p,dis}$	ESP discharge pressure
p_r	Reservoir pressure
f	ESP frequency
z	Choke valve opening
V_1	Pipe volume below ESP
V_2	Pipe volume above ESP
h_1	Height from reservoir to ESP
h_2	Height from ESP to production choke

Source: Binder, Pavlov, and Johansen (2015), Hernes (2020) and Jean P. Jordanou et al. (2022)

$$\begin{aligned} \dot{p}_{bh} &= \frac{V_1}{\beta_1}\left(q_r - q\right) \\ \dot{p}_{wh} &= \frac{V_2}{\beta_2}\left(q - q_c\right) \\ \dot{q} &= \frac{1}{M}\left(p_{bh} - p_{wh} - \rho g(h_1 + h_2) - \Delta p_f + \Delta p_p\right) \end{aligned} \tag{42}$$

where β_1 and β_2 represent the bulk modulus below and above the ESP, respectively. M stands for the fluid inertia parameter. Δp_f and Δp_p denote pressure losses attributed to friction and ESP dynamics, respectively.

$$\begin{aligned} q_r &= PI\left(p_r - p_{bh}\right), \\ q_c &= C_c\sqrt{p_{wh} - p_m}z, \\ \Delta p_f &= F_1 + F_2, \end{aligned} \tag{43}$$

where PI is the production index calculated for the reservoir. C_c signifies the choke valve constant, while F_1 and F_2 symbolize frictional pressure drops below and above the ESP, respectively.

$$\begin{aligned} F_i &= 0.158\frac{\rho L_i q^2}{D_i A_i^2}\left(\frac{\mu}{\rho D_i q}\right)^{\frac{1}{4}}, \\ \Delta p_p &= \rho g H, \\ H &= C_H(\mu)\left(c_0 + c_1\left(\frac{q}{C_Q(\mu)}\frac{f_0}{f}\right) - c_2\left(\frac{q}{C_Q(\mu)}\frac{f_0}{f}\right)^2\left(\frac{f}{f_0}\right)^2\right), \end{aligned} \tag{44}$$

where L_1 and L_2 designate the length from the reservoir to the ESP and from the ESP to the choke, respectively. A_1 represents the cross-sectional area of the pipe below the ESP, and A_2 stands for the cross-sectional area of the pipe above the ESP. D_1 and D_2 correspond to the pipe diameters below and above the ESP, respectively. μ signifies the fluid viscosity, and ρ denotes the density of the produced fluid. H represents the head developed by the ESP, and g denotes the gravitational acceleration constant. Additionally, the viscosity correction factors are denoted as $C_Q(\mu)$ and $C_H(\mu)$.

The parameters used in this work are based on the parameters from Binder, Pavlov, and Johansen (2015). The bulk modulus, density, viscosity and manifold pressure are assumed constant in this dissertation as done by Hernes (2020). Table 8 shows the parameters used to simulate the ESP system.

Table 8 – Parameter Values

Symbol	Value	Unit	Symbol	Value	Unit
g	9.81	m/s^2	β_1	1.5×10^9	pa
C_c	2×10^{-5}	*	β_2	1.5×10^9	pa
A_1	0.008107	m^2	P_r	1.26×10^7	pa
A_2	0.008107	m^2	μ	0.025	$pa \cdot s$
D_1	0.1016	m	ρ	950	kg/m^3
D_2	0.1016	m	PI	2.32×10^9	$m^3/s/pa$
h_1	200	m	P_m	20×10^5	pa
h_2	800	m	f_0	60	Hz
L_1	500	m	M	1.992×10^8	kg/m^4
L_2	1.2	m	c_0	9.5970×10^2	*
V_1	4.054	m^3	c_1	7.4959×10^3	*
V_2	9.729	m^3	c_2	1.2454×10^6	*

Source: Hernes (2020) and Jean P. Jordanou et al. (2022)

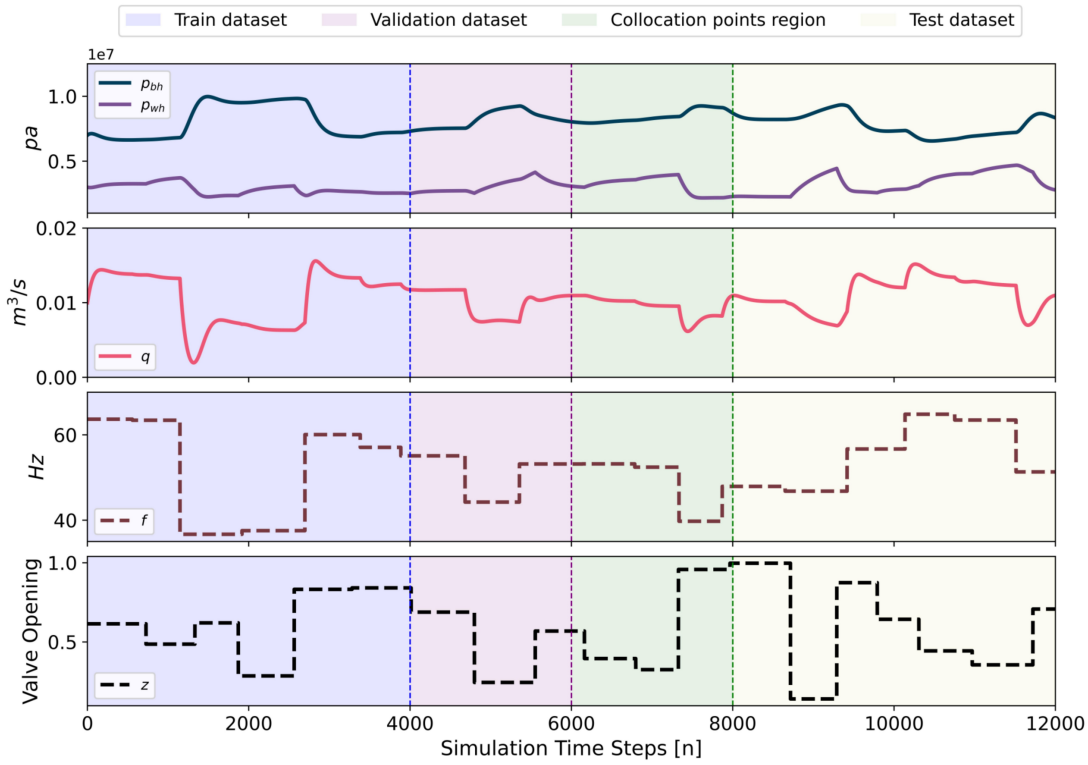
5.3.1 Dataset

To simulate the system, the Gekko library, which is a Python package primarily designed for solving dynamic optimization and control problems, particularly well-suited for Differential Algebraic Equations (DAE) in the context of dynamic systems. Gekko provides tools for modeling and solving complex systems by optimizing their performance over time, making it valuable for various applications, including process control, engineering design, and advanced simulations.

The ESP system was simulated using a time step of $\Delta t = 0.01\ s$. The initial conditions for the system were $p_{bh} = 70 \times 10^6\ pa$, $p_{wh} = 20 \times 10^6\ pa$, and $q = 0.01\ m^3/s$. The simulated system is shown in Figure 39.

An APRBS input signal was generated with values for z ranging from 0.1 to 1 and f ranging from 35 to 65 Hz and a signal variation occurring every 500 to 800 time steps. All the

Figure 39 – Simulated ESP system



Source: Author

plots in Figure 39 depict different data regions. The training set consists of $N_{te} = 4000$ time steps, and the validation set includes $N_{ve} = 2000$ time steps. These two sets are separated by a dashed blue line used for hyperparameter tuning during the ESN training.

Following hyperparameter tuning, there is a purple-dashed line marking the start of the ESN's final training phase, covering $N_t = 6000$ time steps. Afterward, there is the collocation points region, spanning $N_f = 2000$ time steps, indicated by a green dashed line. Finally, the test data comprises 4000 time steps. The data were normalized for ESN training using min-max scaling (Equation (28)).

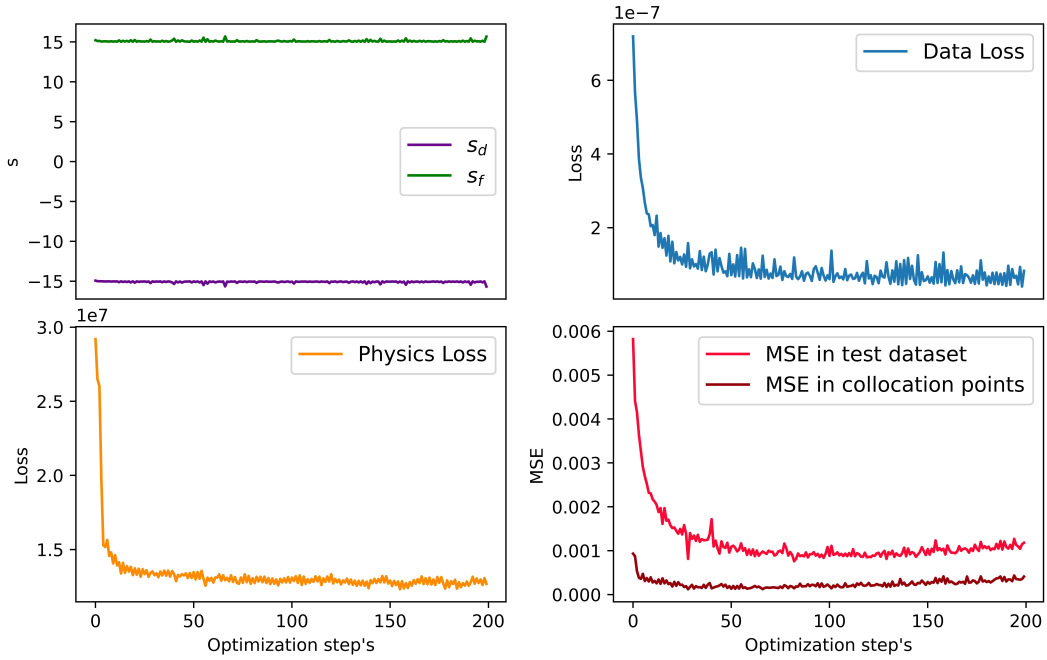
5.3.2 PI-ESN-a settings and results

The ESN was initially configured with hyperparameters close to those suggested in Jean P. Jordanou et al. (2022), utilizing $N_x = 300$ and a warm-up period of 50 time steps. Bayesian optimization was employed to fine-tune the hyperparameters α, $\rho(\mathbf{W})$, δ_b, δ_{fb}, and δ_{in} using the *BayesianOptimization* package in Python. This choice of optimization method was driven by the quantity of hyperparameters involved.

Subsequently, the ESN was retrained with a total of $N_t = 6000$ time steps, utilizing the following values: $\delta_{fb} = 0.1$, $\delta_{in} = 0.1$, $\delta_b = 0.1$, $\gamma = 0.0599$, $\alpha = 0.15$, and $\rho(\mathbf{W}) = 0.8$, as determined through the Bayesian search. Figure 40 shows the physics-informed training process, which showcases the physics and data loss functions, MSE at collocation points,

the test dataset, and the adaptive parameters s_f and s_d.

Figure 40 – Evolution of the adaptive weights (s_d, s_f), the loss functions (J_{data}, J_{phy}) and the MSE during the physics training of the PI-ESN-a for the ESP system.



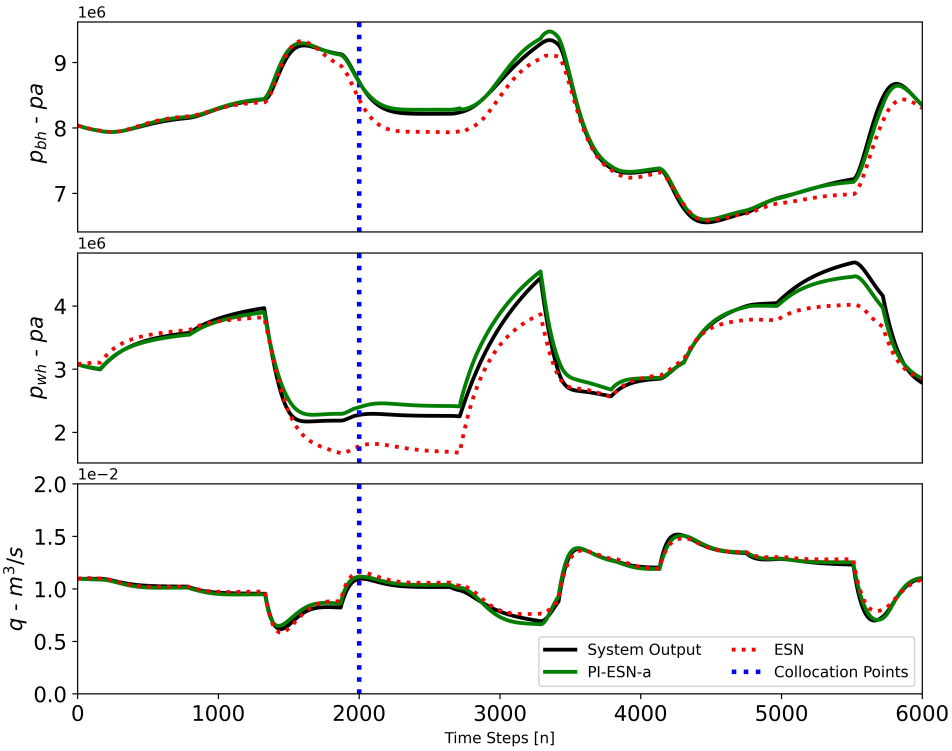
Source: Author

Due to data normalization for ESN training application, a subsequent descaling process became necessary during for the physics-informed training. This led to significant disparities in the magnitudes of the loss functions. Unlike the experiments conducted on the Van der Pol and Four-Tank system, the optimization of s_d and s_f values is more challenging due to disparities in the magnitudes of the physics and data loss functions. This magnitude difference posed a challenge for the balanced self-adaptive loss algorithm, resulting in instabilities during the adaptive training.

After completing the physics-informed training, the values of J_{data} and J_{phy} were determined to be 8.26×10^{-8} and 1.26×10^{7}, respectively. Additionally, the final values for the $[s_d,s_f]$ parameters were calculated as $[-15.67, 15.66]$, this implies that approximately 99.9% of the data loss function while 0.1% of the physics loss function contributed to the calculation of the total loss function. However, it is crucial to notice that the value applied in the physics loss function is non-normalized, whereas in the data loss function, it is normalized. This results in a difference of 15 orders of magnitude between the physics loss function and the data loss function.

The evaluation of the ESP's output results is depicted in Figure 41, the blue dashed vertical line splits the region between collocation points (left) and test set (right). The MSE for the collocation points region was found to be 0.0026 for the ESN and 0.0004 for the PI-ESN-a. In the test dataset, the corresponding MSE values were 0.0084 and 0.0011, respectively.

Figure 41 – Prediction of the PI-ESN-a for ESP after training.



Source: Author

Regarding the execution time of the dataset presented in Figure 39, a comparison was made between the dynamic simulator Gekko and the PI-ESN-a, with 10 runs. The average execution time for PI-ESN was 3.9 seconds, while the Gekko optimizer took 98.5 seconds. This experiment was conducted on an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz processor, with a clock speed of 2592 MHz, 6 cores, and 12 logical processors. This result shows that the execution time of PI-ESN is approximately 96.32% faster than the numerical solution of the DAE.

6 CONCLUSION

In this dissertation, the PI-ESN architecture was extended to incorporate external inputs, making it applicable to dynamical systems requiring control. Moreover, some improvements were proposed for the training process of PI-ESN as follows. Self-adaptive weights originally proposed for PINNs were introduced in the PI-ESN framework to dynamically balance each term (data loss and physics-based loss) within the loss function to be minimized, enhancing the final PI-ESN prediction performance. Additionally, an extensive series of experiments were conducted, showcasing the performance improvements and predictive capabilities of the proposed PI-ESN, especially when compared to ESN, in scenarios characterized by limited data availability. These experiments encompassed three distinct representative dynamical systems modeled by ODEs and DAEs.

In summary, these accomplishments emphasize the versatility and effectiveness of PI-ESN as a tool for modeling and forecasting dynamical systems, particularly within contexts where data is scarce. However, certain issues related to the training instability of this neural network can be addressed in future research, as proposed below:

- Exploring the use of automatic differentiation instead of explicit Euler for calculating the derivatives in the physics loss function could be an option to improve training stability in the physics-based aspect (Racca; Magri, 2021).
- Investigating alternative self-weight adaptive balancing loss functions available in the literature that may offer more stable training.
- Applying PI-ESN to different oil and gas systems with experimental data to assess whether the technique can still be beneficial even if there are differences between the model used in the physics-based training and the real-world plant.
- Examining the impact of changes in initial conditions on the performance of PI-ESN.
- Additionally, applying PI-ESN for the reconstruction of unmeasured hidden states and integrating them into a Predictive Nonlinear Model Predictive Controller (PNMPC) for estimating and controlling these unmeasured states in various applications.

REFERENCES

ABIODUN, O. I.; JANTAN, A.; OMOLARA, A. E.; DADA, K. V.; MOHAMED, N. A.; ARSHAD, H. State-of-the-art in artificial neural network applications: A survey. **Heliyon**, v. 4, n. 11, e00938, 2018.

ALVARADO, I.; LIMON, D.; GARCÍA-GABÍN, W.; ALAMO, T; CAMACHO, E.F. An educational plant based on the quadruple-tank process. **IFAC Proceedings Volumes**, v. 39, n. 6, p. 82–87, 2006. 7th IFAC Symposium on Advances in Control Education.

ANTONELO, Eric; SCHRAUWEN, Benjamin. On Learning Navigation Behaviors for Small Mobile Robots With Reservoir Computing Architectures. **IEEE Transactions on Neural Networks and Learning Systems**, v. 26, n. 4, p. 763–780, 2014.

ANTONELO, Eric Aislan; CAMPONOGARA, Eduardo; FOSS, Bjarne. Echo State Networks for data-driven downhole pressure estimation in gas-lift oil wells. **Neural Networks**, v. 85, p. 106, 2017.

ANTONELO, Eric Aislan; CAMPONOGARA, Eduardo; SEMAN, Laio Oriel; SOUZA, Eduardo Rehbein de; JORDANOU, Jean P; HUBNER, Jomi F. Physics-Informed Neural Nets for Control of Dynamical Systems. **arXiv preprint arXiv:2104.02556**, 2021.

ANTONELO, Eric Aislan; SCHRAUWEN, Benjamin; STROOBANDT, Dirk. Event detection and localization for small mobile robots using reservoir computing. **Neural Networks**, v. 21, n. 6, p. 862–871, 2008.

BINDER, Benjamin J.T.; PAVLOV, Alexey; JOHANSEN, Tor A. Estimation of Flow Rate and Viscosity in a Well with an Electric Submersible Pump using Moving Horizon Estimation. **IFAC-PapersOnLine**, v. 48, n. 6, p. 140–146, 2015. 2nd IFAC Workshop on Automatic Control in Offshore Oil and Gas Production (OOGP) 2015.

BISHOP, Christopher M; NASRABADI, Nasser M. **Pattern Recognition and Machine Learning**. 1. ed. New York: Springer, 2006. (Information Science and Statistics).

BOCCATO, Levy; ATTUX, Romis; VON ZUBEN, Fernando J. Self-organization and lateral interaction in echo state network reservoirs. **Neurocomputing**, v. 138, p. 297–309, 2014. ISSN 0925-2312.

BRAGA, Antônio de Pádua. **Redes Neurais Artificiais: Teoria e Aplicações**. 2. ed. Rio de Janeiro: LTC Editora, 2007.

BRUNTON, Steven L.; KUTZ, J. Nathan. **Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control**. 2. ed. [S.l.]: Cambridge University Press, 2022.

CHEN, Chi-Tsong. **Linear System Theory and Design**. 4. ed. Cary, NC: Oxford University Press, Nov. 2012. (Oxford Series in Electrical and Computer Engineering).

CZAJKOWSKI, Andrzej. Robust Control with Disturbance Estimation Using Echo State Networks for the Twin Rotor Aero-Dynamical System Application. **IFAC Proceedings Volumes**, v. 47, n. 3, p. 11305–11310, 2014. 19th IFAC World Congress.

DEVANEY, Robert L. **An Introduction to Chaotic Dynamical Systems**. [S.l.]: Chapman and Hall/CRC, Oct. 2021.

DOAN, Nguyen Anh Khoa; POLIFKE, Wolfgang; MAGRI, Luca. **A physics-aware machine to predict extreme events in turbulence**. [S.l.: s.n.], 2019. arXiv: 1912.10994 [physics.flu-dyn].

DOAN, Nguyen Anh Khoa; POLIFKE, Wolfgang; MAGRI, Luca. **Learning Hidden States in a Chaotic System: A Physics-Informed Echo State Network Approach**. [S.l.]: arXiv preprint arXiv: 2001.02982, 2020.

DOAN, Nguyen Anh Khoa; POLIFKE, Wolfgang; MAGRI, Luca. Physics-informed echo state networks for chaotic systems forecasting. In: SPRINGER. International Conference on Computational Science. [S.l.: s.n.], 2019. P. 192–198.

EDWARDS, Chris. Neural Networks Learn to Speed up Simulations. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 65, n. 5, p. 27–29, 2022.

ELSHEIKH, Ammar H.; SHARSHIR, Swellam W.; ABD ELAZIZ, Mohamed; KABEEL, A. E.; GUILAN, Wang; HAIOU, Zhang. Modeling of solar energy systems using artificial neural network: A comprehensive review. **Solar Energy**, v. 180, p. 622–639, 2019.

FILHO, H. S. A. RIZZO. **Otimização de Gás lift na Produção de Petróleo: Avaliação da Curva de Performance do Poço**. 2011. Dissertation (Master in Energy Planning) – Universidade Federal do Rio de Janeiro.

GREENHILL, Stewart; RANA, Santu; GUPTA, Sunil; VELLANKI, Pratibha; VENKATESH, Svetha. Bayesian Optimization for Adaptive Experimental Design: A Review. **IEEE Access**, v. 8, p. 13937–13948, 2020.

HERNES, Sondre Bø. **Practical NMPC of Electrical Submersible Pumps based on Echo State Networks**. 2020. Dissertation (Master in Cybernetics and Robotics) – Norwegian University of Science and Technology.

HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, Nov. 1997. ISSN 0899-7667.

JAEGER, Herbert. The ‘‘echo state’’ approach to analysing and training recurrent neural networks – with an Erratum note. In: German National Research Center for Information Technology. [S.l.]: Fraunhofer Institute for Autonomous Intelligent Systems, 2001.

JAEGER, Herbert; LUKOŠEVIČIUS, Mantas; POPOVICI, Dan; SIEWERT, Udo. Optimization and applications of echo state networks with leaky- integrator neurons. **Neural Networks**, v. 20, n. 3, p. 335–352, 2007. Echo State Networks and Liquid State Machines.

JOHANSSON, Karl Henrik. The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero. **IEEE Transactions on Control Systems Technology**, v. 8, n. 3, p. 456–465, 2000.

JORDANOU, Jean; ANTONELO, Eric; CAMPONOGARA, Eduardo. Online learning control with Echo State Networks of an oil production platform. **Engineering Applications of Artificial Intelligence**, v. 85, p. 214–228, 2019.

JORDANOU, Jean P.; CAMPONOGARA, Eduardo; ANTONELO, Eric Aislan; AGUIAR, Marco Aurélio Schmitz. Nonlinear Model Predictive Control of an Oil Well with Echo State Networks. **IFAC-PapersOnLine**, v. 51, n. 8, p. 13–18, 2018.

JORDANOU, Jean P.; OSNES, Iver; HERNES, Sondre B.; CAMPONOGARA, Eduardo; ANTONELO, Eric Aislan; IMSLAND, Lars. Nonlinear Model Predictive Control of Electrical Submersible Pumps based on Echo State Networks. **Advanced Engineering Informatics**, v. 52, p. 101553, 2022.

JORDANOU, Jean Panaioti. **Echo State Networks for Online Learning Control and MPC of Unknown Dynamic Systems: Applications in the Control of Oil Wells**. 2019.

Dissertation (Master in Automation and System Engineering) – Universidade Federal de Santa Catarina.

KARNIADAKIS, George Em; KEVREKIDIS, Ioannis G.; LU, Lu; PERDIKARIS, Paris; WANG, Sifan; YANG, Liu. Physics-informed machine learning. **Nature Reviews Physics**, v. 3, n. 6, p. 422–440, 2021.

KHALIL, Hassan K. **Nonlinear Systems**. 3. ed. Upper Saddle River, NJ: Pearson, Dec. 2001.

LORENZ, Edward N. Deterministic Nonperiodic Flow. **Journal of the Atmospheric Sciences**, American Meteorological Society, v. 20, n. 2, p. 130–141, Mar. 1963.

LUKOŠEVIČIUS, Mantas. A Practical Guide to Applying Echo State Networks. In: **Neural Networks: Tricks of the Trade: Second Edition**. Ed. by Grégoire Montavon, Geneviève B. Orr and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. P. 659–686.

NA, Xiaodong; LI, Yuan; REN, Weijie; HAN, Min. Physics-informed hierarchical echo state network for predicting the dynamics of chaotic systems. **Expert Systems with Applications**, v. 228, p. 120155, 2023.

NICODEMUS, Jonas; KNEIFL, Jonas; FEHR, Jörg; UNGER, Benjamin. Physics-informed Neural Networks-based Model Predictive Control for Multi-link Manipulators. **IFAC-PapersOnLine**, v. 55, n. 20, p. 331–336, 2022. 10th Vienna International Conference on Mathematical Modelling (MATHMOD) 2022.

OH, Dong Keun. **Toward the Fully Physics-Informed Echo State Network – an ODE Approximator Based on Recurrent Artificial Neurons**. [S.l.: s.n.], 2020. arXiv: 2011.06769 [cs.LG].

PATHAK, Jaideep; WIKNER, Alexander; FUSSELL, Rebeckah; CHANDRA, Sarthak; HUNT, Brian R.; GIRVAN, Michelle; OTT, Edward. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. **Chaos: An Interdisciplinary Journal of Nonlinear Science**, AIP Publishing, v. 28, n. 4, p. 041101, 2018.

PILARIO, Karl Ezra Salgado; CAO, Yi; SHAFIEE, Mahmood. A Kernel Design Approach to Improve Kernel Subspace Identification. **IEEE Transactions on Industrial Electronics**, v. 68, n. 7, p. 6171–6180, 2021.

RACCA, Alberto; MAGRI, Luca. Automatic-differentiated Physics-Informed Echo State Network (API-ESN). **CoRR**, abs/2101.00002, 2021.

RAISSI, Maziar; PERDIKARIS, Paris; KARNIADAKIS, George E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. **Journal of Computational Physics**, v. 378, p. 686–707, 2019.

ROBERTS, Ciaran; LARA, José Daniel; HENRIQUEZ-AUBA, Rodrigo; BOSSART, Matthew; ANANTHARAMAN, Ranjan; RACKAUCKAS, Chris; HODGE, Bri-Mathias; CALLAWAY, Duncan S. Continuous-time echo state networks for predicting power system dynamics. **Electric Power Systems Research**, v. 212, p. 108562, 2022.

SALMEN, M.; PLOGER, P.G. Echo State Networks used for Motor Control. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation. [S.l.: s.n.], 2005. P. 1953–1958.

SAVALIA, S.; EMAMIAN, V. Cardiac Arrhythmia Classification by Multi-Layer Perceptron and Convolution Neural Networks. **Bioengineering (Basel)**, v. 5, n. 2, 2018.

SCHRAUWEN, Benjamin; WARDERMANN, Marion; VERSTRAETEN, David; STEIL, Jochen J.; STROOBANDT, Dirk. Improving reservoirs using intrinsic plasticity. **Neurocomputing**, v. 71, n. 7, p. 1159–1171, 2008. Progress in Modeling, Theory, and Application of Computational Intelligenc. ISSN 0925-2312.

SCHWEDERSKY, Bernardo B.; FLESCH, Rodolfo C. C.; DANGUI, Hiago A. S. Identificação de Sistemas Dinâmicos Não Lineares Multivariáveis com Redes de Estado de Eco. In: Congresso Brasileiro de Automática (CBA). [S.l.]: Sociedade Brasileira de Automática, 2020. v. 2.

SHAHI, Shahrokh; FENTON, Flavio H.; CHERRY, Elizabeth M. Prediction of chaotic time series using recurrent neural networks and reservoir computing techniques: A comparative study. **Machine Learning with Applications**, v. 8, p. 100300, 2022.

SILVA, Ivan Nunes da; SPATTI, Danilo Hernane; FLAUZINO, Rogério Andrade. **Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas: Fundamentos Teóricos e Aspectos Práticos**. 2. ed. São Paulo: Artliber, 2016.

TAKACS, Gabor. **Electrical Submersible Pumps Manual**. 2nd Edition. [S.l.]: Elsevier, 2018.

TANGIRALA, Arun K. **Principles of system identification: Theory and practice**. Boca Raton, FL: CRC Press, Dec. 2014.

THIEDE, Luca Anthony; PARLITZ, Ulrich. Gradient based hyperparameter optimization in Echo State Networks. **Neural Networks**, v. 115, p. 23–29, 2019.

THOMAS, Jose Eduardo. **Fundamentos de Engenharia de Petróleo**. [S.l.]: Editora Interciência, 2004.

TSATSOS, Marios. **Theoretical and Numerical Study of the Van der Pol equation**. 2008. Dissertation (Master in Astrophysics Astronomy and Mechanics Mechanics) – Aristotle University of Thessaloniki.

VERSTRAETEN, D.; SCHRAUWEN, B.; D'HAENE, M.; STROOBANDT, D. An experimental unification of reservoir computing methods. **Neural Networks**, v. 20, n. 3, p. 391–403, 2007.

WAEGEMAN, Tim; WYFFELS, Francis; SCHRAUWEN, Benjamin. Feedback Control by Online Learning an Inverse Model. **IEEE Transactions on Neural Networks and Learning Systems**, v. 23, n. 10, p. 1637–1648, 2012.

XIANG, Zixue; PENG, Wei; LIU, Xu; YAO, Wen. Self-adaptive loss balanced Physics-informed neural networks. **Neurocomputing**, v. 496, p. 11–34, 2022.

YANG, C.; QIAO, J.; AHMAD, Z.; NIE, K.; WANG, L. Online sequential echo state network with sparse RLS algorithm for time series prediction. **Neural Networks**, v. 118, p. 32–42, 2019.

YILDIZ, Izzet B.; JAEGER, Herbert; KIEBEL, Stefan J. Re-visiting the echo state property. **Neural Networks**, v. 35, p. 1–9, 2012.

YPERMAN, Jan; BECKER, Thijs. **Bayesian optimization of hyper-parameters in reservoir computing**. [S.l.: s.n.], 2017. arXiv: 1611.05193 [cs.LG].

ZHENG, Yingzhe; WU, Zhe. Physics-Informed Online Machine Learning and Predictive Control of Nonlinear Processes with Parameter Uncertainty. **Industrial & Engineering Chemistry Research**, v. 62, n. 6, p. 2804–2818, 2023.

ZHOU, Jian; HAN, Taotao; XIAO, Fu; GUI, Guan; ADEBISI, Bamidele; GACANIN, Haris; SARI, Hikmet. Multiscale Network Traffic Prediction Method Based on Deep Echo-State Network for Internet of Things. **IEEE Internet of Things Journal**, v. 9, n. 21, p. 21862–21874, 2022.