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

Jean Panaioti Jordanou

Methods for Analysis and Model Predictive Control of Dynamic Systems Modeled with Echo State Networks

Florianópolis
2024

Jean Panaioti Jordanou

Methods for Analysis and Model Predictive Control of Dynamic Systems Modeled with Echo State Networks

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de doutor em Engenharia de Automação e Sistemas.
Orientador: Prof. Eduardo Camponogara, Dr.
Coorientador: Prof. Eric A. Antonelo, Dr.

Florianópolis
2024

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

Jordanou, Jean Panaioti
Methods for Analysis and Model Predictive Control of Dynamic Systems Modeled with Echo State Networks / Jean Panaioti Jordanou ; orientador, Eduardo Camponogara, coorientador, Eric Aislan Antonelo, 2024.
128 p.

Tese (doutorado) - 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. Echo State Networks.. 3. Model Order Reduction. 4. Stability Analysis. 5. Model Predictive Control. I. Camponogara, Eduardo. II. Antonelo, Eric Aislan. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Jean Panaioti Jordanou

Methods for Analysis and Model Predictive Control of Dynamic Systems Modeled with Echo State Networks

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

Prof. Leandro Coelho, Dr.
Escola Politécnica
Pontifícia Universidade Católica - PR

Prof. Claudio Gallicchio, Dr.
Departamento de Ciências da Computação
Universidade de Pisa

Prof. Eugênio Castelan Neto, 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 doutor em Engenharia de Automação e Sistemas.

ICP-Edu
Documento assinado digitalmente
Julio Elias Normey Rico
Data: 09/02/2024 11:10:49-0300
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
Eduardo Camponogara
Data: 08/02/2024 15:10:49-0300
CPF: ***.988.180-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Eduardo Camponogara, Dr.
Orientador

Florianópolis, 2024.

ACKNOWLEDGEMENTS

Firstly, I'd like to thank my advisor, prof. Eduardo Camponogara, for all the support provided throughout my academic years, from the undergraduate internship to the Ph.D. Prof. Camponogara was paramount in my development as a scholar, providing not only advice but also moral support. I'd also like to thank my co-advisor for introducing me to Echo State Network research and supporting me through the journey. I'd also like to thank Prof. Eduardo Gildin for receiving and advising me during my stay at Texas A&M University. I couldn't imagine a better reception and company during those six months. Prof. Gildin provided me with a lot of knowledge in broadening my research horizon and was crucial in making my stay in the USA more bearable while also contributing significantly to the research presented in this work.

Another important contribution to this work comes from the funding agencies. I'd like to thank both CAPES and Petrobras for funding my research during my Ph.D. years and making developing this dissertation possible.

And last but not least, I'd like to thank all my lab colleagues, both from UFSC and Texas A&M University, and my colleagues from the post-graduate program. Having to do my Ph.D. during the coronavirus pandemic showed how crucial it was to have them around, providing good laughs and company, and even exchanging knowledge. Also, I'd like to thank my friends for being there for me and my family. There is no such thing as a self-made man.

"Freedom is the freedom to say that 2+2=4. If that is granted, all else follows"
(George Orwell)

RESUMO

Redes de Estado de Eco (*Echo State Networks*, ESNs) são redes neurais consolidadas como modelos de identificação de sistemas, com treinamento por mínimos quadrados e capazes de representar diversos tipos de modelos. Porém, suas aplicações em controle são pouco estudadas em comparação com outros modelos de redes neurais. O objetivo dessa tese de doutorado é estudar aspectos de controle de sistemas dinâmicos modelados por Redes de Estado de Eco. A rede de estado de eco demanda um número alto de estados para identificar sistemas não lineares apenas treinando os pesos de saída, o que naturalmente resulta em um problema computacional quando se considera esse tipo de sistema em aplicações de controle, otimização e controle preditivo baseado em modelo. Como a rede de estado de eco é um sistema não-linear, sua prova de estabilidade em malha fechada não é trivial, o que torna interessante o desenvolvimento de métodos de análise de estabilidade para a mesma. Levando esses fatores em conta, a pesquisa segue em três frentes. A primeira consiste em aplicar experimentos com relação ao *Practical Nonlinear Model Predictive Control* (PNMPC), explorando suas capacidades em diversos tipos de aplicação, avaliando métricas tais como erro de seguimento e tempo computacional. Foram considerados diversos casos de estudo simulados, como o sistema de quatro tanques, uma plataforma de petróleo contendo dois poços e um riser, e um poço com produção por meio de uma bomba elétrica submersível. O controlador demonstrou êxito em tarefas tanto de rejeição de perturbação quanto de seguimento de referência, até mesmo em comparação com outras estratégias de controle. Para resolver o problema do alto número de estados da rede de estado de eco, essa tese estuda a possibilidade de se utilizar técnicas de redução de ordem de modelo como *Proper Orthogonal Decomposition* e *Discrete Empirical Interpolation* para reduzir o número de estados, assim tornando o modelo mais tratável para aplicações de controle ótimo e controle preditivo baseado em modelo, porém tentando manter a precisão e a representatividade. Foram feitas avaliações da capacidade de redução desses métodos perante a ESN considerando a métrica de capacidade de memória, um sistema *Nonlinear Autoregressive with Moving Average* (NARMA) de décima ordem, e a mesma plataforma com dois poços e um riser dos experimentos com o PNMPC, demonstrando uma grande eficiência em obter um modelo aproximado com um número significativamente reduzido de estados. A pesquisa desta tese também demonstra que é possível expressar o problema de controle de uma rede de estado de eco via *Linear Matrix Inequalities* (LMI) através do problema de estabilidade absoluta. A formulação é desenvolvida tanto para um controle de ganho linear, quanto para um *Nonlinear Model Predictive Control* (NMPC) com considerações conservadoras. Esses métodos são testados em uma ESN que identifica o modelo de um tanque como prova de conceito. Desenvolver e efetuar demonstrações em estudos de caso para os métodos mencionados acima, constitui contribuições ao estado da arte para controle com Redes de Estado de Eco.

Palavras-chaves: Redes de Estado de Eco. Redução de Ordem de Modelo. Análise de Estabilidade. Controle Preditivo.

RESUMO EXPANDIDO

Introdução

Redes de Estado de Eco (ESN, *Echo State Network*) são redes neurais consolidadas como modelos de identificação de sistemas, com fácil treinamento e alta representatividade de modelos, e uma gama diversificada de aplicações em identificação de sistemas e predição de séries temporais. Em contrapartida, o uso dessas redes em aplicações de controle em comparação com outros modelos de redes neurais é pouco estudado. A ESN como modelo de identificação possui potencial se aplicada como modelo de controle preditivo, tornando-se interessante desenvolver ferramentas e executar experimentos nesse contexto. Dito isso, seu número elevado de estados pode se apresentar como um empecilho computacional na execução de algoritmos de otimização. Isso se deve à própria natureza da Computação por Reservatórios, que exige um número elevado de estados por parte da ESN para poder representar uma dinâmica. Além disso, como a ESN é um sistema não-linear, ainda não há um método simples e computacionalmente aplicável de provar sua estabilidade, principalmente no contexto de controle de malha fechada e controle preditivo baseado em modelo. Um método compreensível e intuitivo de prova de estabilidade para controle com ESN seria benéfico para a interpretabilidade do modelo de identificação.

Objetivos

O objetivo dessa tese é avaliar aspectos de controle de sistemas dinâmicos modelados por Redes de Estado de Eco. A pesquisa segue em três frentes. A primeira se trata de aplicar experimentos com relação ao Practical Nonlinear Model Predictive Control (PNMPC), melhor explorando suas capacidades em diversos tipos de aplicação, avaliando métricas tais como erro de seguimento e tempo computacional, e comparando com outros métodos. Isso se referindo ao aspecto da inserção da ESN no contexto de controle preditivo. A segunda frente se trata da possibilidade de utilizar técnicas de redução de ordem de modelo em Redes de Estado de Eco para reduzir o número de estados, assim tornando o modelo ainda mais tratável para aplicações de controle ótimo e controle preditivo, porém tentando manter a precisão e a representatividade. É possível utilizar métodos de redução de ordem de modelo para reduzir o número de estados, mas mantendo um perfil dinâmico similar. Na terceira frente, se deseja obter uma prova de estabilidade para diferentes controladores, como realimentação por ganho e *Nonlinear Model Predictive Control*, utilizando redes de estado de eco como modelo de projeto, obtendo não só uma garantia de estabilidade local, mas também uma estimativa de região de atração.

Metodologia

Para a primeira frente de pesquisa, se desenvolve um algoritmo para o PNMPC com ESN como modelo de predição, e após isso o controlador desenvolvido é testado em diversos tipos de aplicações com simulação. Dentre elas, um sistema de quatro tanques, uma plataforma de petróleo contendo dois poços e um riser, e um poço de petróleo com elevação por bomba

submergida. Nesses experimentos, se avalia métricas como tempo computacional e erro com relação à referência, além da comparação do controlador com outros métodos. Para a segunda frente, se implementa o método de redução de ordem *Proper Orthogonal Decomposition* (POD) na ESN, para fins de comparação com sua rede original. Como o POD para um sistema não-linear reduz o número de estados mas não necessariamente reduz o número de computações, também se analisa a implementação do *Discrete Empirical Interpolation Method* (DEIM), que tenta diminuir o custo computacional da ESN reduzida por POD através de uma aproximação por interpolação. Como caso de estudo se utiliza a avaliação da *Memory Capacity* da ESN reduzida em comparação com a original, um sistema discreto não-linear de décima ordem com uma entrada e uma saída, e uma plataforma de produção de petróleo com dois poços e um riser. Para a terceira frente, emprega-se ferramentas de estabilidade absoluta para desenvolver *Linear Matrix Inequalities* (LMIs) que descrevem, em forma de condição suficiente, a estabilidade da ESN, tanto realimentada por um ganho linear, quanto quando utilizada como modelo por um algoritmo de *Nonlinear Model Predictive Control* (NMPC). Nesse caso, é utilizado o modelo de um tanque como prova de conceito.

Resultados e Discussões

A ESN demonstrou ser um modelo viável para o uso em controle preditivo em todas a aplicações propostas, com o quatro tanques sendo utilizado para avaliar seu desempenho geral, o sistema de dois poços e um riser demonstrando sua viabilidade em aplicações mais complexas, e o poço com elevação por bomba submergível oferecendo uma comparação entre NMPC e PNMPC, avaliando os dois controles preditivos no contexto do uso de uma ESN como modelo. Também se obteve excelente performance em comparação com o controle preditivo utilizando outros métodos de identificação, como a rede neural *Long Short-Term Memory* (LSTM) e uma rede do tipo *feedforward* com dinâmica externa. O método POD conseguiu efetuar uma redução de ordem de modelo com comportamento bastante próximo à ESN original, e ainda reduzir o tempo computacional. Para a aplicação do DEIM, para surtir algum efeito na identificação, foi preciso adicionar um algoritmo para garantia de estabilidade. Foi possível desenvolver uma LMI para análise de estabilidade de uma ESN controlada por uma lei de controle com ganho proporcional. Também foi possível obter LMIs para estabilizar NMPC com ESN, desconsiderando o efeito da variável de decisão na não-linearidade do sistema.

Considerações Finais

Ao desenvolver e efetuar demonstrações em estudos de caso para os métodos acima, a presente tese gerou contribuições ao estado da arte para controle em Redes de Estado de Eco. Além de uma implementação para PNMPC com ESN como modelo de predição, foi possível obter um modelo de ordem reduzida com comportamento equivalente a uma ESN de ordem completa, mesmo com essa possuindo um número de estados elevado por natureza. Através da obtenção de LMIs de estabilidade para uma ESN, é possível fazer inferência da estabilidade de uma rede

de estado de eco, utilizando apenas os seus pesos.

Palavras-chaves: Redes de Estado de Eco. Redução de Ordem de Modelo. Análise de Estabilidade. Controle Preditivo.

ABSTRACT

Echo State Networks (ESN) are a flavor of Recurrent Neural Networks widely applied as system identification models, as they are easy to train and have high representation capacity. However, their use in control applications is not widely studied in comparison to other neural network models. This research aims to evaluate dynamic system control aspects for ESNs. ESNs have an intrinsically high number of states, which leads to computation overhead when performing heavy computation (i.e., optimization) with these networks. Also, stability proofs for closed-loop control with ESNs are not intuitive, hence the need for an easy, systematic stability analysis method. Considering these factors, this research follows three main fronts: the first one being the experimentation of the so-called Practical Nonlinear Model Predictive Control (PNMPC) with ESNs as a model. Three case studies were considered: a four-tank system, an oil and gas platform considering two gas-lifted wells and one riser, and an ESP-lifted oil well. The resulting ESN-PNMPC performed well in both trajectory tracking and disturbance rejection in all cases compared with other controllers. The following research topic concerns employing model order reduction techniques such as Proper Orthogonal Decomposition (POD) and Discrete Empirical Interpolation in the ESN to reduce the number of states, rendering the model easier to compute for MPC and Optimal Control applications. An experiment compares the memory capacity of ESNs with their reduced counterparts, while two case studies were evaluated: a Nonlinear Autoregressive with Moving Average (NARMA) system of 10th order and the same two wells and one riser platform of the PNMPC experiment. The reduction effectively found an ESN reduced-order model, behaving closely to the original but with significantly fewer states. The third front led to the development of a Linear Matrix Inequality stability proof for feedback control with Echo State Networks, showing that the stability of a controller with an ESN in the loop can be expressed as an absolute stability problem. This result produced a stability guarantee and an attraction region estimate for a linear gain controller and an NMPC, with conservative considerations. A single tank model trained by a small ESN was considered a proof of concept. Developing such methods, exploring these three research fronts, and demonstrating their use in case studies contributes to expanding the state-of-the-art regarding Echo State Networks for control applications.

Keywords: Echo State Networks. Model Order Reduction. Stability Analysis. Model Predictive Control.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

Machine Learning (ML) and, more specifically, Deep Learning are ascending areas that have had a disruptive impact on relevant problems in both academia and industry (TORRES et al., 2021). These fields include computer vision (LIU, Y.; CHENG; WANG, W., 2018), image classification (HE, 2020), natural language processing (OTTER; MEDINA; KALITA, 2019), system identification (NELLES, 2020), and time series analysis. Those last two mentioned areas are important in many applications such as control design, prediction, and simulation, especially in the context of systems engineering (CAMACHO; BORDONS, 1999; NELLES, 2020).

There are many tools in ML for system identification. In particular, Recurrent Neural Networks (RNNs) (GOODFELLOW; BENGIO; COURVILLE, 2016) have achieved excellent results in that regard. They are a sort of bio-inspired, nonlinear dynamic system that, through a suitable weight tuning (i.e., training process), can learn to reproduce the behavior of any plant/real-life dynamic system, with arbitrary accuracy (NELLES, 2020) in principle, and without prior physical information about the system in question.

The main issue with the original RNNs involves fading gradients during RNN training. This hindrance to effective training of RNNs has led to specialized architectures aiming to overcome this issue, such as the Long Short-Term Memory (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) network and the Gated Recurrent Unit (GRU) network. Fading gradient is a numerical problem that emerges when computing the gradient of the cost function concerning the RNN's parameters (or any deep network for that matter), called Backpropagation Through Time (BPTT) (MOZER, 1989). As BPTT works by propagating the error backward through the time series using the chain rule, and as the activation function is generally the hyperbolic tangent, the back-propagated error may tend to zero as it moves back in time. LSTMs and GRUs change the neuron structure to avoid and diminish the fading gradient problem. However, this work follows an alternative approach to RNN training, called Reservoir Computing (RC), that circumvents not only the fading gradient issue but also the many local optima that happen due to the nonlinearity of conventional RNN training.

Reservoir Computing (RC) (SCHRAUWEN; VERSTRAETEN, David; VAN CAMPENHOUT, 2007) is a training method for RNNs that came about as a unifying paradigm for works developed a few years back, such as Echo State Networks (ESNs) (JAEGER, 2001) and Liquid State Machines (MAASS; MARKRAM, 2004). RC separates the RNN into a non-trainable, recurrent network, referred to as a Reservoir, and a trainable static output layer representing the output prediction as a linear combination of the reservoir states. The reason for the name "Reservoir" (TANAKA et al., 2019) is that the static reservoir must have a large number of states. A large number of recurrent neurons implies a sufficiently rich pool of dynamics for correctly computing an output trajectory corresponding to the system being identified. This

work in question focuses on Echo State Networks, as they are a simple flavor of RC suitably designed for engineering applications.

Echo State Networks have many interesting applications in the literature, such as learning complex goal-directed robot behaviors (ANTONELO, Eric A; SCHRAUWEN, 2015), grammatical structure processing (HINAUT; DOMINEY, 2012), short-term stock prediction (LIN; YANG, Z.; SONG, 2009), speech emotion recognition in Metaverse (DANESHFAR; JAMSHIDI, 2023) (working with octonions instead of real numbers), predicting stellar mass accretion (BINO et al., 2023), and noninvasive fetal detection (LUKOŠEVIČIUS; MAROZAS, 2014). Also, in (CHEN, C.; LIU, H., 2021), the ESN serves as a basis for the prediction model of a wind speed time series, where it is also coupled with tools such as reinforcement learning and real-time wavelet packet decomposition.

Control applications benefit very much from system identification (NELLES, 2020), as data-driven models can either be used as a basis to tune a controller or be implemented as part of the controller itself, such as in Model Predictive Control (MPC) (CAMACHO; BORDONS, 1999). Although only a few, there are some instances in the literature of control applications with Echo State Networks as the nominal plant model. For instance, Pan and J. Wang (2012) apply a real-time linearization in an identified ESN, which is thus in state-space form, while also relaxing the resulting Quadratic Programming (QP) problem of the Model Predictive Control. Armenio et al. (2019) work with full NMPC and ESN models, also developing a stability proof for it. There are other instances of ESN control in the literature unrelated to MPC. Jean P. Jordanou, Eric Aislan Antonelo, and Camponogara (2019), for example, employ ESNs in an online learning control, where the controller infers an inverse model of the plant as data is fed online to the network through time. Qiang Chen et al. (2018) utilize an ESN as a basis for the design of a backstepping nonlinear controller.

One interesting approach that provides an efficient MPC when combined with ESN comes from the Practical Nonlinear Model Predictive Control (PNMPC). In fact, this ESN-PNMPC approach was initially developed in a previous work (JORDANOU, Jean P. et al., 2018) and is further developed and tested in this research, considering three additional plants: a four-tank system, an oil production system consisting of two gas-lifted oil wells and one riser (JORDANOU, Jean Panaioti; ANTONELO, Eric Aislan; CAMPONOGARA, 2022), and an ESP-lifted oil well (JORDANOU, Jean P. et al., 2022). In that preliminary work, PNMPC uses an ESN as a prediction model and calculates the Jacobian according to a recursive rule that gives an analytical derivative. This research explores further the properties of the so-called ESN-PNMPC as one of the main topics, further expanding the experimental analysis regarding the MPC method at hand, such as testing the ESN-PNMPC on mode complex plants, and exploring properties related to the parameters, computation time, and comparison to other types of controllers.

Naturally, Reservoir Computing inherently needs the number of states to be far larger than the number of inputs and outputs. This fact alone directly impacts the computing time

of any optimization problem employing an ESN model compared to a less complex counterpart. Although some works in the literature seek to find the minimum complexity Echo State Network to perform a specific task (RODAN; TINO, 2010), they have not been designed for a control scenario, i.e., they do not reduce the number of states, but the number of weight connections. For MPC, a model with fewer states is desirable, as the number of states plays a sensitive part in MPC employing state-space models (mainly due to the derivative chain rule when computing gradients). Hence, there is a demand for an equivalent model to the ESN, however, with fewer states.

There is an extensive literature for model order reduction (MOR) of large-scale dynamic systems (SILJAK, 2007; FLOREZ; GILDIN, 2019; GHASEMI; IBRAHIM; GILDIN, 2014), with special mention to the Proper Orthogonal Decomposition (MIJALKOVIC, 2006), the Discrete Empirical Interpolation Method (CHATURANTABUT; SORENSEN, 2010), autoencoders (AN; CHO, 2015), and the Embed-to-Control E2C (WATTER et al., 2015) strategy. The latter was initially developed for time-series identification with data as a sequence of images (a video stream) but is also utilized in oil and gas reservoirs. The hypothesis in this research is that Echo State Networks can benefit from such MOR techniques to find an approximate ESN with fewer states. In this way, this work hopes to advance the performance of an RC model in control tasks, where not only the training of RNNs is efficient, but also the optimization problem complexity is further reduced and easier to solve.

Another point of attention that might come to mind is that there are few comprehensible, systematic proofs of stability involving ESNs, especially concerning control loops. Since the traditional activation function for an ESN (the hyperbolic tangent) has a similar behavior to the saturation function, a possibility is to find a proof that converts the ESN stability verification problem into an absolute stability problem. The implication is reducing any open and closed-loop stability proof into an LMI, which can be solved numerically through Semidefinite Programming (SDP) methods, with a formulation akin to Eugênio B. Castelan, Sophie Tarbouriech, and Queinnec (2008), who considered a nonlinear system with saturated control action for the absolute stability problem formulation.

Eugênio B. Castelan, Sophie Tarbouriech, and Queinnec (2008) also guide on performing stability analysis for a saturated controller on a system with arbitrary nonlinearity. In parallel, Cisneros and Werner (2018) provide a method to convert the process of solving a QP given its linear term, into a nonlinear operation that obeys a particular so-called sector condition. Since the ESN is already being reformulated into an absolute stability problem, an NMPC employing it as a model could be represented as a QP, with the static nonlinearity as an exogenous input, in the same vein as Cisneros and Werner (2018), separating the LPV model into a dynamic LTI and a static LPV to solve its problem. The fact that such stability proof for QP MPC can be demonstrated allows us to represent an ESN-NMPC stability as an LMI that provides a sufficient stability condition proof.

1.2 OBJECTIVES

The general objective of this research is to develop methods that advance the state-of-the-art in the subject of Echo State Networks when applied in control loops, with a focus on Model Predictive Control. With this direction in mind, this dissertation led to the following original contributions:

- Analyzed ESN-PNMPC properties regarding its computational efficiency and performance on different plants, comparing with alternative controllers such as ESN-NMPC and PNMPC with LSTM.
- Studied MOR strategies to find approximate ESNs for control applications, such as Model Predictive Control, with less computing overhead and more accuracy. Several experiments were carried out to assess the application of Proper Orthogonal Decomposition (POD) and Discrete Empirical Interpolation Method (DEIM) for model order reduction of ESNs.
- Developed a stability proof for ESNs in a loop with different controllers that can verify the closed-loop stability and estimate a region of attraction given an arbitrary initial condition. The stability conditions provide interpretability and a systematic way to obtain information about stability in control design problems such as the state feedback gain controller. This work also contributed to the initial development of a numerical procedure to calculate the stability of NMPC with ESNs as a model.

1.3 THESIS ORGANIZATION

The rest of this document is organized as follows:

- Chapter 2 is an overview of the theories that serve as a basis for this study.
- Chapter 3 introduces the ESN-PNMPC control strategy and shows the new experiments performed, which are detailed in (JORDANOU, Jean Panaioti; ANTONELO, Eric Aislan; CAMPONOGARA, 2022) and (JORDANOU, Jean P. et al., 2022).
- Chapter 4 describes the Model Order Reduction methods, which are based on a published work (JORDANOU, Jean Panaioti et al., 2023).
- Chapter 5 showcases the fundamental elements and steps to assess the stability of NMPC with ESNs as a model.
- Chapter 6 concludes the dissertation, discussing its limitations and suggesting future works.

2 THEORETICAL BACKGROUND

This chapter presents the theoretical background necessary to understand the concepts involved in this research. Every content here is applied in one way or another in the developed methods.

2.1 DYNAMIC SYSTEMS

This section gives an overview of dynamic systems. A system is dynamic if, to compute the current system output, one needs to provide prior input and output information (CHEN, C.-T., 1998). This feature sets them apart from static systems, which only depend on the present input (as in the static linear system with input vector $\mathbf{x}$ and output vector $\mathbf{y}$, $\mathbf{y} = \mathbf{A}\mathbf{x}$).

The dependency of the output with respect to a space of previous inputs and outputs is captured by the concept of a **state**, which is by definition an accumulation of all the previous inputs that were applied into the system since the beginning of time ($t = -\infty$). A dynamic system can be either continuous or discrete time. As the continuous system response has a continuous domain, differential equations depict the derivative of the state response. In discrete systems, the time domain is countable, so a function that directly maps the previous step into the next one (the so-called transition function) is available (CHEN, C.-T., 1998).

The generic mathematical representation of a continuous-time dynamic system in state equations is:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t)) \tag{1}$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t),\mathbf{u}(t)) \tag{2}$$

at any given time t, where the states are represent by $\mathbf{x} \in \mathbb{R}^n$, with n being the number of states; the inputs by $\mathbf{u} \in \mathbb{R}^m$, with m being the number of inputs; and the outputs of the system are $\mathbf{y} \in \mathbb{R}^o$, with o being the number of outputs. A dynamic system without inputs is referred to as an autonomous system (KHALIL, 2018). The dynamic system's initial condition ($\mathbf{x}_0 = \mathbf{x}(t = 0)$) must be known to fully be able to solve the differential equation (1) and obtain a system response. The vector function $\mathbf{f} : (\mathbb{R}^n \times \mathbb{R}^m) \mapsto \mathbb{R}^n$ is called the state transition function, which computes the state gradient at time t based on the input and the state. The function $\mathbf{f}$ is also referred to sometimes as the flow map, since it represents a flow vector field in a state space. The vector function $\mathbf{g} : (\mathbb{R}^n \times \mathbb{R}^m) \mapsto \mathbb{R}^o$ is a static mapping to define the output of the dynamic system. A discrete dynamic system, in turn, is represented by the following equations:

$$\mathbf{x}[k+1] = \mathbf{f}(\mathbf{x}[k],\mathbf{u}[k]) \tag{3}$$

$$\mathbf{y}[k] = \mathbf{g}(\mathbf{x}[k],\mathbf{u}[k]) \tag{4}$$

where the same notation of the continuous case is used, and $k \in \mathbb{N}$ is the discrete time step. The notation $\mathbf{x}[k]$ is equivalent to $\mathbf{x}(t = T_s k)$, with $T_s \in \mathbb{R}$ being the sampling time

of the system. Nonlinear differential equations (difference equations in the case of discrete dynamic systems) do not have a unified method of obtaining an analytical solution available, therefore the most usual way to compute their response is through numerical simulation (or direct computation of $\mathbf{x}[k+1]$ given $\mathbf{x}[k]$ and $\mathbf{u}[k]$ in the case of discrete systems). Since it is not possible to directly get analytical responses for nonlinear dynamic systems, qualitative properties play an important part in system analysis. One such property is the equilibrium point $\bar{\mathbf{x}}$, a state value where the system is stationary ($\lim_{t\mapsto\infty}\mathbf{x}(t) = \bar{\mathbf{x}}$ if $\mathbf{x}(0) = \mathbf{x}_0$). For continuous-time autonomous systems (without loss of generality as one can consider any fixed input), this means that $\mathbf{f}(\bar{\mathbf{x}}) = \mathbf{0}$, as the flow map gradient being null means the system is stationary at point $\bar{\mathbf{x}}$. As for discrete-time autonomous systems, $\mathbf{f}(\bar{\mathbf{x}}) = \bar{\mathbf{x}}$ at equilibrium, which is a fixed point.

2.1.1 Stability Analysis

Equilibrium points can be either stable or unstable. A stable equilibrium point is a state which the system converges to, from any given initial condition located at an arbitrarily small distance from it ($\lim_{t\mapsto\infty}\mathbf{x}(t) = \bar{\mathbf{x}}$ for any $\mathbf{x}(0) = \mathbf{x_0}, \|\mathbf{x_0} - \bar{\mathbf{x}}\| < \epsilon$, ϵ being a small scalar). The region that converges to a given equilibrium point is referred to as the **region of attraction** of it.

The are two main forms to evaluate the stability of a nonlinear system (KHALIL, 2018):

- Linearizing the system by calculating the Jacobian of $\mathbf{f}$ at point $\bar{\mathbf{x}}$ and some $\bar{\mathbf{u}}$ in the state equation:

$$\boldsymbol{\Delta}\dot{\mathbf{x}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\boldsymbol{\Delta}\mathbf{x}) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\boldsymbol{\Delta}\mathbf{u} \tag{5}$$

  the stability properties of the resulting system holds at least locally for the nonlinear system, except for when the matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ has any Eigenvalue λ where $\Re(\lambda) = 0$.

- The Lyapunov Stability, which was created based on energy properties of mechanical systems (KHALIL, 2018). Given an energy scalar function $V(\mathbf{x})$, the flow map of a dynamic system is the vector field that dictates the system trajectory along $V(\mathbf{x})$. The Lyapunov function $V(\mathbf{x})$ must be positive definite for all $\mathbf{x}$, except at an equilibrium point $\bar{\mathbf{x}}$. The Lyapunov stability theorem for continuous systems states that (KHALIL, 2018), given a Lyapunov function $V(\mathbf{x})$ and a dynamic system as in (1), if $\frac{dV(\mathbf{x})}{dt} < 0$, then the system is at least locally asymptotically stable. Applying the chain rule shows that the derivative of $V(\mathbf{x})$ is a directional derivative with the system trajectory as the associated vector field:

$$\frac{dV(\mathbf{x})}{dt} = \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}\dot{\mathbf{x}} = \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x},\mathbf{u}) \tag{6}$$

  The flow map $\mathbf{f}$ is a vector field that dictates the trajectory of the state $\mathbf{x}$ along the Lyapunov function V. As the function is positive definite and the derivative

is negative definite, then any trajectory sequence converges to a local minimum for V. Therefore, if one sees V as an energy function, a derivative over time that is always negative means that the system will converge to the point of minimum energy, and therefore present stability. Generally speaking, it is common practice to use the quadratic Lyapunov function (KHALIL, 2018):

$$V(\mathbf{x}) = \mathbf{x}^T\mathbf{P}\mathbf{x} \tag{7}$$

with $\mathbf{P} \succ 0$. The Lyapunov function (7) has the nice property of having an easily computable gradient:

$$\frac{dV(\mathbf{x})}{dt} = \mathbf{f}(\mathbf{x},\mathbf{u})^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\mathbf{P}\mathbf{f}(\mathbf{x},\mathbf{u}) \tag{8}$$

For any nonlinear system, if the derivative over time of the Lyapunov function is not positive definite, then it does not imply absence of asymptotic stability, and another Lyapunov function must be chosen.

Generally, the linearization of the system is the easiest path to check if the system is stable or how the system behaves near the equilibrium point. However, the Lyapunov stability approach also has its benefits in the form of being able to estimate an attraction region for a given equilibrium point (KHALIL, 2018), and being a general tool for designing nonlinear controllers. A nonlinear system may have many equilibrium points or none at all, and that information is lost when it is linearized. Also, a property called limit cycle (a state trajectory that repeats itself periodically (KHALIL, 2018), which is similar to an equilibrium point that can be stable or unstable) can be, in certain cases, detected by the Lyapunov function. Unfortunately, there is no available tool to easily detect limit cycles in a given system, and each particular system has a particular method involved.

2.1.2 Linear Dynamic Systems

A particular case of dynamic systems is of interest because it is way more thoroughly analyzed (CHEN, C.-T., 1998): linear systems. As shown in the previous sections, one way to analyze stability is to linearize the nonlinear system. This is because, for linear systems, the analytical solution of the differential equation is readily available, therefore one can easily know how the system behaves based on its form. Consider the generic linear system without direct transmission:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{9}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} \tag{10}$$

The analytical solution of this differential equation has the following form.

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{u}(\tau)d\tau \tag{11}$$

The matrix $e^{\mathbf{A}t}$ has the following form:

$$\mathbf{V}\begin{pmatrix} e^{\lambda_1 t} & \frac{te^{\lambda_1 t}}{1!} & \frac{t^2 e^{\lambda_1 t}}{2!} & \cdots & \frac{t^n e^{\lambda_1 t}}{(n-1)!} & 0 & 0 & 0 \\ 0 & e^{\lambda_1 t} & \frac{te^{\lambda_1 t}}{1!} & \cdots & \frac{t^{n-1} e^{\lambda_1 t}}{(n-2)!} & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & e^{\lambda_1 t} & te^{\lambda_1 t} & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & e^{\lambda_1 t} & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & e^{\lambda_2 t} & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & e^{\lambda_m t} \end{pmatrix}\mathbf{V}^{-1} \tag{12}$$

where $\mathbf{V}$ corresponds to the generalized Eigenvectors of $\mathbf{A}$ and the middle matrix is the inverse Laplace transform of $\mathbf{A}$ in Jordan form. Here, it is assumed without loss of generality that $\mathbf{A}$ has Eigenvalue λ_1 with multiplicity n, and has m different Eigenvalues, where $\lambda_i, i \in \{2,3,4,\ldots,m\}$ has multiplicity one. This implies that the Eigenvalues of matrix $\mathbf{A}$ hold precious information about the dynamics of the system. Any exponential e^{at} with positive a can imply a divergent trajectory when $t \mapsto \infty$, therefore a positive Eigenvalue indicates an unstable system. Along this line, the Eigenvalues also hold information about convergence time (for a negative a, $1/a$ is the time it takes for $e^{at} = e^{-1} \approx 0.37$, therefore the larger the a, the faster is the convergence time) and damping (the system has no oscillation when all the Eigenvalues are real, otherwise sinusoidal functions appear). The implication of the matrix Eigenvalue is that it is possible to tune a controller that forces the closed-loop system to have certain Eigenvalues, thus making the system behave in a certain desired way.

For the Lyapunov stability, the derivative over time $\frac{dV}{dt}$ for a quadratic Lyapunov function is:

$$\frac{dV}{dt} = \mathbf{x}^T(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A})\mathbf{x} = -\mathbf{x}^T\mathbf{Q}\mathbf{x} \tag{13}$$

where $\mathbf{Q}$ is some positive definite matrix. Contrary to the nonlinear case, when a quadratic negative definite derivative for the Lyapunov function is not found for the linear system, then it can be affirmed that the system is unstable (CHEN, C.-T., 1998). Therefore, since the Lyapunov function time derivative is negative if, and only if, the system is stable, the stability problem can be solved through a matrix equation. The matrix $\mathbf{Q}$ is given and the matrix containing the coefficients of the Lyapunov Function, $\mathbf{P}$, is found through the following equation:

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q} \tag{14}$$

If a positive definite $\mathbf{P}$ can be obtained, then the system is stable. Also, for linear systems, if $\mathbf{P}$ is not positive definite, then the system is unstable (CHEN, C.-T., 1998). This equation is referred to as the Lyapunov equation and is used as one of the fundamentals for many robust control and linear optimal control methods, since it presents itself as a way to place stability in the form of matrix constraints for optimization problems.

There are other important concepts related to linear systems, such as controlability, observability, conversion to transfer function, state-space realization, and feedback control through state feedback and output feedback (using observers), which will not be discussed in this dissertation, but more information on then is found in (CHEN, C.-T., 1998).

2.1.3 LMI-based Stability Analysis

As seen in equation (13) and (14), the linear system stability is expressed in terms of a matrix equation with $\mathbf{P}$ as the variable. The same stability condition can be expressed as:

$$\mathbf{x}^T(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A})\mathbf{x} < 0 \tag{15}$$

$$(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A}) \prec 0 \tag{16}$$

with the symbol $\prec 0$ denoting semidefineteness of a matrix. Equations such as (16) are referred to as Linear Matrix Inequalities (LMI). The variable in these equations is always a matrix, such as $\mathbf{P}$, and the equation is always linear in terms of $\mathbf{P}$. As the objective is to find $\mathbf{P}$ such that the LMI holds, and there is no unique solution for $\mathbf{P}$, the solution of the LMI is always tied to an SDP (semi-definite programming) optimization problem.

2.1.3.1 Example: Closed-loop Stabilization of a Continuous Time System

Given a system in closed loop with a linear controller:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{17a}$$

$$\mathbf{u} = \mathbf{K}\mathbf{x} \tag{17b}$$

The objective is to find $\mathbf{K}$ such that the closed loop system:

$$(\mathbf{A} + \mathbf{B}\mathbf{K})\mathbf{x} \tag{18}$$

is stable. The matrix inequality representing such system is:

$$(\mathbf{A} + \mathbf{B}\mathbf{K})^T\mathbf{P} + \mathbf{P}(\mathbf{A} + \mathbf{B}\mathbf{K}) \prec 0 \tag{19}$$

Such an equation would be classified as an LMI if $\mathbf{K}$ were given, however the gain is also included as a variable. To turn the stability equation into an LMI, the variables are substituted. Consider the expansion of the above inequality:

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{K}^T\mathbf{B}^T\mathbf{P} + \mathbf{P}\mathbf{B}\mathbf{K} \prec 0$$

The first step is to define a matrix $\mathbf{Q} = \mathbf{P}^{-1}$, and then pre and post multiply the matrix inequality as follows:

$$\mathbf{Q}\mathbf{A}^T + \mathbf{A}\mathbf{Q} + \mathbf{Q}\mathbf{K}^T\mathbf{B}^T + \mathbf{B}\mathbf{K}\mathbf{Q} \prec 0$$

The next step is to do $\mathbf{Y} = \mathbf{KQ}$, so that:

$$\mathbf{QA}^T + \mathbf{AQ} + \mathbf{Y}^T\mathbf{B}^T + \mathbf{BY} \prec 0 \tag{20}$$

The equation (20) is an LMI with variables $\mathbf{Q}$ and $\mathbf{Y}$. $\mathbf{P}$ and $\mathbf{K}$ are then directly derived, and a stabilizing gain is possible to be found.

It is possible to include extra conditions in the stability formulation, such as dynamic constraints, and robustness terms, which are extensively explored in the literature.

2.1.4 Absolute Stability

Assume an autonomous nonlinear system in the following form:

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\phi(\mathbf{y}[k]) \tag{21}$$

$$\mathbf{y}[k] = \mathbf{C}\mathbf{x}[k] \tag{22}$$

where $\phi(\mathbf{y})$ is a nonlinear map that follows a given **sector condition**.

A sector condition (TARBOURIECH, Sophie et al., 2011a) is an an inequality condition where any value of $\phi(\mathbf{y})$ is inside an envelope of linear functions $\Omega_{\min}\mathbf{y}$ and $\Omega_{\max}\mathbf{y}$, and the inequality in question can be expressed as:

$$2(\phi - \Omega_{\min}\mathbf{y})^T(\phi - \Omega_{\max}\mathbf{y}) \leq 0 \tag{23}$$

The domain of the nonlinearity input $\mathbf{y} \in \mathcal{D}$ must contain the origin (TARBOURIECH, Sophie et al., 2011a).

Given the discrete-time system described in (21), the system is considered to have **absolute stability** if it is open-loop stable assuming $\phi(\mathbf{y})$ as an external output that is bounded by the sector condition. The absolute stability conditions for a discrete-time system can be expressed in the following quadratic inequality:

$$\begin{pmatrix}\mathbf{x}\\ \phi\end{pmatrix}^T \begin{pmatrix}\mathbf{A}^T\mathbf{PA} - \mathbf{P} & \mathbf{A}^T\mathbf{PB}\\ \mathbf{B}^T\mathbf{PA} & \mathbf{B}^T\mathbf{PB}\end{pmatrix} \begin{pmatrix}\mathbf{x}\\ \phi\end{pmatrix} \leq 0 \tag{24}$$

given the Lyapunov function $\mathbf{V}[k] = \mathbf{x}[k]^T\mathbf{P}\mathbf{x}[k]$ and the stability condition that it decreases over time steps $\mathbf{V}[k+1] - \mathbf{V}[k] < 0$.

Both (23) and (24) must hold at the same time to obtain absolute stability, therefore (TARBOURIECH, Sophie et al., 2011a):

$$\begin{pmatrix}\mathbf{x}\\ \phi\end{pmatrix}^T \begin{pmatrix}\mathbf{A}^T\mathbf{PA} - \mathbf{P} & \mathbf{A}^T\mathbf{PB}\\ \mathbf{B}^T\mathbf{PA} & \mathbf{B}^T\mathbf{PB}\end{pmatrix} \begin{pmatrix}\mathbf{x}\\ \phi\end{pmatrix} + 2(\phi - \Omega_{\min}\mathbf{y})^T(\phi - \Omega_{\max}\mathbf{y}) \leq 0$$

By transforming the second term into matrix form, with $\mathbf{y} = \mathbf{Cx}$, one finally obtains the LMI that defines the absolute stability of a system in the form (21):

$$\begin{pmatrix}\mathbf{x}\\ \phi\end{pmatrix}^T \begin{pmatrix}\mathbf{A}^T\mathbf{PA} - \mathbf{P} + \mathbf{C}^T\Omega_{\min}\Omega_{\max}\mathbf{C} & \mathbf{A}^T\mathbf{PB} - \mathbf{C}^T(\Omega_{\min} + \Omega_{\max})\\ \mathbf{B}^T\mathbf{PA} - (\Omega_{\min} + \Omega_{\max})\mathbf{C} & \mathbf{B}^T\mathbf{PB} + 2\mathbf{I}\end{pmatrix} \begin{pmatrix}\mathbf{x}\\ \phi\end{pmatrix} \leq 0 \tag{25}$$

Equation (25) is an example of a L'ure stability quadratic inequality, and finding $\mathbf{P}$ that makes such inequality hold $\forall \mathbf{x}, \phi$ proves absolute stability for discrete systems in the form of (21) and any nonlinearity ϕ under sector $(\Omega_{\min}, \Omega_{\max})$. Other derivations can be made for continuous systems, and specific nonlinearities for saturation.

Absolute stability is a subject for many applications, and can even be useful for proving stability for nonlinear MPC (CISNEROS; WERNER, 2018).

2.2 ARTIFICIAL NEURAL NETWORKS (ANN)

Artificial Neural Networks (ANNs) have been coined as such from biological sciences, as an attempt to find a mathematical model of the brain (BISHOP, 2006). In engineering, they are mostly seen as a nonlinear powerful statistical model (NELLES, 2020). An ANN has the following characteristics:

- large number of simple units;
- highly parallel units;
- densely connected units;
- fault tolerance of single units.

As the aforementioned characteristics imply, these networks are heavily distributed, so the combination of each unit serves as an universal function approximator, which entails an easy hardware implementation (NELLES, 2020).

The main unit of an ANN is the "neuron". A neuron in this context is a mapping between a linear combination of inputs and a given output, which is computed through a given activation function f. In most tasks, it is usual to use activation functions such as $\tanh(\cdot)$ or the sigmoid$(\cdot)$, the latter defined as:

$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}} \tag{26}$$

Figure 1 shows the curves of a hyperbolic tangent and a sigmoid in a 2-d plot, respectively. Both the hyperbolic tangent and the sigmoid function are bounded, which makes them useful for logistic regression (BISHOP, 2006). Since both functions converge to asymptotes from both the negative and positive sides, their gradient becomes closer to zero as the function input magnitude $|z|$ increases, which is commonly referred to as a "fading gradient" that can cause numerical problems in the gradient calculation. The deep learning literature (GOODFELLOW; BENGIO; COURVILLE, 2016) proposes another activation function for the neural networks to avoid the "fading gradient": the Rectifier Linear Unit (ReLU) (GLOROT; BORDES; BENGIO, 2011), calculated as:

$$f(z) = \max(0, z) \tag{27}$$

Glorot, Bordes, and Bengio (2011) argue that the most important property of a biological neuron is the retification (the diode effect of nullifying negative stimuli). Also, biological neurons

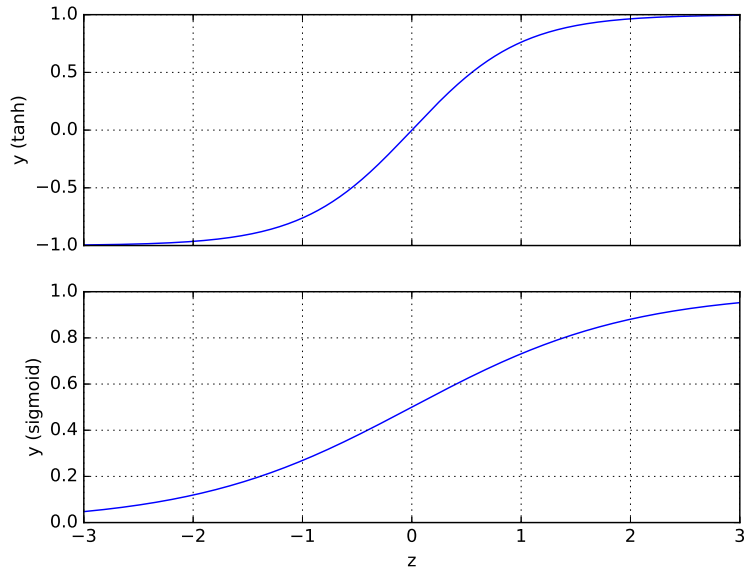


Figure 1 – Plot of a hyperbolic tangent and a sigmoid function.

rarely reach the maximum firing magnitude. Both these facts explain the proposal of ReLU, which is ubiquitous in image recognition and computer vision applications (GOODFELLOW; BENGIO; COURVILLE, 2016).

Inside a neural network, the neurons are linked through weighted connections, as each neuron receives as input a linear combination of the outputs from other neurons. The distributed composition between activation functions, alongside the weight tuning, defines a feedforward neural network as a universal approximator of static functions (CYBENKO, 1989). For regression applications, the weights are obtained by minimizing the quadratic error of some desired values, solving a nonlinear least squares problem.

ANNs, overall, are ideal in some applications where a large amount of data is available and the patterns are not trivial enough to be identified through a linear regression model. The neural network is a complex, nonlinear model, therefore prone to overfitting, which welcomes the implementation of regularization solutions. Viable strategies for regularization in an ANN include the Tikhonov Regularization (BISHOP, 2006) or the LASSO (least absolute shrinkage and selection operator) regularization (TIBSHIRANI, 1996). While Tikhonov utilizes ℓ_2-norm, LASSO utilizes ℓ_1-norm which is able to obtain networks with sparse weights, significantly reducing model complexity.

2.3 FEEDFORWARD NEURAL NETWORKS

The most basic type of neural networks are the Feedforward Neural Networks (also known as Multilayer Perceptron (MLP). The main characteristic of these networks is that, since the value of each neuron does not depend on itself in previous instants in time, they lack dynamics, hence feedforward. To ease gradient calculation and function evaluation, the

neurons are commonly organized in layers.

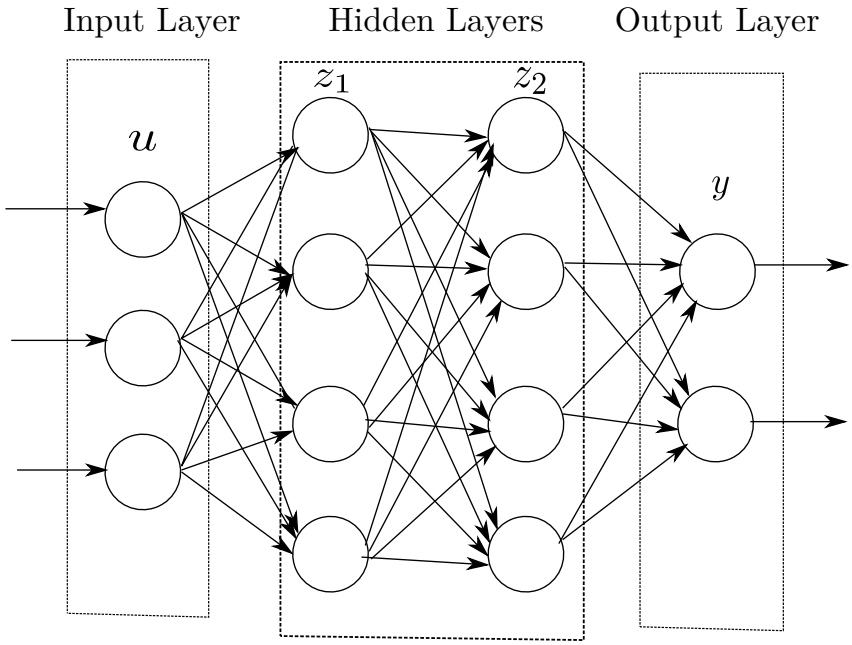


Figure 2 – Representation of a Feedforward Neural Network with 3 inputs, 2 hidden layers with 4 neurons each, and 2 outputs. Each arrow represents a weighted connection between each neuron, represented by a circle.

Figure 2 depicts a feedforward neural network, which is governed by the following equations:

$$\mathbf{y} = \mathbf{f}(\mathbf{W_{n_l}} \mathbf{z_{n_l}}) \tag{28}$$

$$\mathbf{z_n} = \mathbf{f}(\mathbf{W_{n-1}} \mathbf{z_{n-1}}) \tag{29}$$

$$\mathbf{z_1} = \mathbf{f}(\mathbf{W_0} \mathbf{u}) \tag{30}$$

where the network has n_l layers, and $\mathbf{z}_n$ is a vector where each element is a layer-n neuron. The index n_l also doubles as the index of the final layer. $\mathbf{u}$ is the input and $\mathbf{y}$ is the output vector. The matrix $\mathbf{W_0}$ defines the weights between the input layer and the first hidden layer. The matrix $\mathbf{W_n}$ gives the weights between layer n and layer $n+1$. Consequentially, $\mathbf{W_{n_l}}$ is the weight matrix of that maps the hidden neurons from the last layer $\mathbf{z_{n_l}}$ into the outputs. The vector function $\mathbf{f}(\cdot)$ is the element-wise activation function, which is unspecified for this analysis. For instance, in Figure 2, the network has $n_l = 2$, and is described by the following equations:

$$\mathbf{y} = \mathbf{f}(\mathbf{W_2} \mathbf{z_2})$$

$$\mathbf{z_2} = \mathbf{f}(\mathbf{W_1} \mathbf{z_1})$$

$$\mathbf{z_1} = \mathbf{f}(\mathbf{W_0} \mathbf{u})$$

Functions such as $\tanh(\cdot)$ or the sigmoid$(\cdot)$ as presented in Equation (26) are common activation functions. Bias can be added by concatenating the input vector $\mathbf{u}$ and each layer

vector $\mathbf{z_n}$ with 1 and adding a weight corresponding to the connection between the bias and a given neuron. A neural network that has a large n_l, that is, a large number of layers, is considered a Deep Neural Network (GOODFELLOW; BENGIO; COURVILLE, 2016).

In a neural network, the trainable weights are the elements in each matrix $\mathbf{W_n}$ where the index $n \in \{0,1,...,n_l\}$. To simplify the optimization process, consider a vector $\mathbf{w}$ that is the flattened and concatenated form of a matrix $\mathbf{W_n}$. In regression applications, the objective is to minimize the quadratic error between the desired output $\widehat{\mathbf{y}}$ and the expected output $\mathbf{y}(\mathbf{w},\mathbf{u})$ (for simplicity, assume there is only one output in the following formulation):

$$E(\mathbf{w},\mathbf{u}) = \frac{1}{2}(\widehat{y} - y(\mathbf{w},\mathbf{u}))^2 \tag{31}$$

Calculating the gradient in a neural network was an impeding task in the early years, up until the error Backpropagation algorithm was invented (NELLES, 2020). Backpropagation is essentially the chain rule calculation of the gradient, which is facilitated by the layered structure of the network. The derivative of the cost function with respect to $\mathbf{w}$ is:

$$\frac{\partial E}{\partial \mathbf{w}} = (\widehat{y} - y(\mathbf{w},\mathbf{u}))\frac{\partial y}{\partial \mathbf{w}} \tag{32}$$

The value for $\frac{\partial y}{\partial \mathbf{w}}$ depends on which layer the weights are. If the weights are at the output layer, then the answer is trivial (since y, $\mathbf{W_{n_l}}$ is a row vector):

$$\frac{\partial y}{\partial \mathbf{W_{n_l}}} = \mathbf{f'}(\mathbf{W_{n_l}}\mathbf{z_{n_l}})\mathbf{z_{n_l}}^T \tag{33}$$

For y with respect to an arbitrary hidden layer weight matrix $\mathbf{W_n}$, the derivative is:

$$\frac{\partial y}{\partial \mathbf{W_n}} = \frac{\partial y}{\partial \mathbf{a_n}}\mathbf{z_n}^T \tag{34}$$

with $\mathbf{a_n} = \mathbf{W_n}\mathbf{z_n}$. The calculation of $\frac{\partial y}{\partial \mathbf{a_n}}$ is performed recursively:

$$\frac{\partial y}{\partial \mathbf{a_n}} = \frac{\partial \mathbf{a_{n+1}}}{\partial \mathbf{a_n}}^T \frac{\partial y}{\partial \mathbf{a_{n+1}}} = \mathbf{f'}(\mathbf{a_n})\mathbf{W_{n+1}}^T \frac{\partial y}{\partial \mathbf{a_{n+1}}} \tag{35}$$

In fact, $\frac{\partial y}{\partial \mathbf{a_{n_l}}} = \mathbf{f'}(\mathbf{a_{n_l}})$, so the derivative of the activation function is used to calculate the gradient of the previous layer, so on and so forth. To initialize the algorithm, it is advisable that the weights are started randomly, as the optmization problem has many different optimal points. Starting the weights as 0 could bind the network to a bad local optimum (BISHOP, 2006).

2.4 RECURRENT NEURAL NETWORKS

RNNs are Neural Networks whose neurons depend on their own previous values in time. They are necessary to predict time series (GOODFELLOW; BENGIO; COURVILLE, 2016) and solve system identification problems (NELLES, 2020). Since these networks are dependent

on time, their training is harder than feedforward networks, however as their feedforward counterparts are universal function approximators, the RNN are universal dynamical systems approximators. The general equation for a recurrent neural network is:

$$\mathbf{x}[k+1] = \mathbf{f}(\mathbf{W}\mathbf{x}[k] + \mathbf{W_u}\mathbf{u}[k] + \mathbf{b}) \tag{36}$$

$$\widehat{\mathbf{y}}[k] = \mathbf{W_o}\mathbf{x}[k] \tag{37}$$

with $\mathbf{x}[k]$ being the vector containing the hidden neurons, $\mathbf{u}[k]$ the input vector, and $\mathbf{b}$ the bias vector. The function $\mathbf{f}$ is the activation function. It is needed to train each parameter $\mathbf{W}$, $\mathbf{W_o}$, $\mathbf{b}$ and $\mathbf{W_u}$ to minimize a quadratic error over time. This problem is solved by the Backpropagation Through Time algorithm (GOODFELLOW; BENGIO; COURVILLE, 2016), which is the Backpropagation counterpart for RNN.

As in the case of the feedforward neural networks, the optimization problem minimizes the following cost function (the output is assumed to be scalar):

$$E(\mathbf{w},\mathbf{u}) = \frac{1}{2}\sum_{k=1}^{N}(\widehat{y}[k] - y[k])^2 \tag{38}$$

where $\mathbf{w}$ is a vector containing the parameter matrices $\mathbf{W}$, $\mathbf{W_u}$, and $\mathbf{W_o}$ flattened into a column vector and further concatenated with $\mathbf{b}$. The objective of Backpropagation through time is to calculate $\frac{\partial E}{\partial \mathbf{w}}$. By using the chain rule:

$$\frac{\partial E}{\partial \mathbf{w}} = \sum_{k=1}^{N}\frac{\partial \mathbf{x}[k]}{\partial \mathbf{w}}^T\frac{\partial E}{\partial \mathbf{x}[k]} \tag{39}$$

The term $\frac{\partial E}{\partial \mathbf{x}[k]}$ represents the effect that the state of the network at a given time k has at the quadratic error function. When $k = N$, $\mathbf{x}[N]$ appears only once, so:

$$\frac{\partial E}{\partial \mathbf{x}[N]} = \frac{\partial E}{\partial \widehat{y}[N]}\frac{\partial \widehat{y}[N]}{\partial \mathbf{x}[N]} = (\widehat{y}[N] - y[N])\mathbf{W_o}^T \tag{40}$$

For an arbitrary k, as the gradient for $k = N$ is already given, it is convenient to express the gradient in terms of $\frac{\partial E}{\partial \mathbf{x}[k+1]}$. The algorithm starts evaluating from $\frac{\partial E}{\partial \mathbf{x}[N]}$ and backpropagates until $k = 1$.

$$\frac{\partial E}{\partial \mathbf{x}[k]} = \frac{\partial \mathbf{x}[k+1]}{\partial \mathbf{x}[k]}^T\frac{\partial E}{\partial \mathbf{x}[k+1]} + \frac{\partial E}{\partial \widehat{y}[k]}\frac{\partial \widehat{y}[k]}{\partial \mathbf{x}[k]} \tag{41}$$

$$\frac{\partial E}{\partial \mathbf{x}[k]} = (\widehat{y}[k] - y[k])\mathbf{W_o}^T + \mathbf{W}^T\mathbf{f}'(\mathbf{a}[k])\frac{\partial E}{\partial \mathbf{x}[k+1]} \tag{42}$$

$$\mathbf{a}[k] = \mathbf{W}\mathbf{x}[k] + \mathbf{W_u}\mathbf{u}[k] + \mathbf{b} \tag{43}$$

Only the calculation of $\frac{\partial \mathbf{x}[k]}{\partial \mathbf{w}}$ is left. Since $\mathbf{w}$ corresponds to $\mathbf{W}$, $\mathbf{W_u}$, $\mathbf{b}$ and $\mathbf{W_o}$ flattened into a column vector and concatenated, the respective gradients must be calculated

separately. First, the bias weight vector $\mathbf{b}$:

$$\frac{\partial \mathbf{x}[k]}{\partial \mathbf{b}} = \frac{\partial \mathbf{a}[k-1]}{\partial \mathbf{b}}^T \frac{\partial \mathbf{x}[k]}{\partial \mathbf{a}[k-1]} + \frac{\partial \mathbf{x}[k-1]}{\partial \mathbf{b}}^T \frac{\partial \mathbf{x}[k]}{\partial \mathbf{x}[k-1]} \tag{44}$$

$$\frac{\partial \mathbf{x}[k]}{\partial \mathbf{b}} = \mathbf{f}'(\mathbf{a}[k-1]) + \frac{\partial \mathbf{x}[k-1]}{\partial \mathbf{b}}^T \mathbf{W}^T \mathbf{f}'(\mathbf{a}[k-1]) \tag{45}$$

when $k = 2$, the second term is dropped, since $k = 1$ is defined as the initial condition in this formulation, and does not depend on $\mathbf{b}$. The value for $k = 2$ would be:

$$\frac{\partial \mathbf{x}[2]}{\partial \mathbf{b}} = \mathbf{f}'(\mathbf{a}[1]) \tag{46}$$

which is propagated forward until $\mathbf{x}[N]$.

For an arbitrary input weight row vector $\mathbf{w}_u(i)$, where i is the corresponding line in $\mathbf{W_u}$, the calculations are analogous:

$$\frac{\partial \mathbf{x}[k]}{\partial \mathbf{w}_u(i)} = \frac{\partial \mathbf{x}[k]}{\partial a_i[k-1]} \frac{\partial a_i[k]}{\partial \mathbf{w}_u(i)} + \frac{\partial \mathbf{x}[k]}{\partial \mathbf{x}[k-1]} \frac{\partial \mathbf{x}[k-1]}{\partial \mathbf{w}_u(i)} \tag{47}$$

$$\frac{\partial \mathbf{x}[k]}{\partial \mathbf{w}_u(i)} = \mathbf{f}_i'(a_i[k-1])\mathbf{u}^T[k] + \mathbf{W}^T \mathbf{f}'(\mathbf{a}[k-1]) \frac{\partial \mathbf{x}[k-1]}{\partial \mathbf{w}_u(i)} \tag{48}$$

where $\mathbf{f}_i'$ is a column vector with only row i non-zero. For an arbitrary state row vector $\mathbf{w}_x(i)$, where i is the corresponding line in $\mathbf{W}$, the following equations hold:

$$\frac{\partial \mathbf{x}[k]}{\partial \mathbf{w}_x(i)} = \frac{\partial \mathbf{x}[k]}{\partial a_i[k-1]} \frac{\partial a_i[k]}{\partial \mathbf{w}_x(i)} + \frac{\partial \mathbf{x}[k]}{\partial \mathbf{x}[k-1]} \frac{\partial \mathbf{x}[k-1]}{\partial \mathbf{w}_x(i)} \tag{49}$$

$$\frac{\partial \mathbf{x}[k]}{\partial \mathbf{w}_x(i)} = \mathbf{f}_i'(a_i[k-1])\mathbf{x}^T[k] + \mathbf{W}^T \mathbf{f}'(\mathbf{a}[k-1]) \frac{\partial \mathbf{x}[k-1]}{\partial \mathbf{w}_x(i)} \tag{50}$$

For the output weights, the calculation is trivial, as they do not depend on the state $\mathbf{x}$:

$$\sum_{k=1}^{N} (\widehat{y} - y)\mathbf{x}[k]^T \tag{51}$$

To finally obtain $\frac{\partial \mathbf{E}}{\partial \mathbf{w}}$, the weight vector and matrices are concatenated:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial \mathbf{E}}{\partial \mathbf{W_o}}^T \\ \frac{\partial \mathbf{E}}{\partial \mathbf{b}} \\ \frac{\partial \mathbf{E}}{\partial \mathbf{w_u}(1)}^T \\ \frac{\partial \mathbf{E}}{\partial \mathbf{w_u}(2)}^T \\ \vdots \\ \frac{\partial \mathbf{E}}{\partial \mathbf{w_u}(n_u)}^T \\ \frac{\partial \mathbf{E}}{\partial \mathbf{w_x}(1)}^T \\ \frac{\partial \mathbf{E}}{\partial \mathbf{w_x}(2)}^T \\ \vdots \\ \frac{\partial \mathbf{E}}{\partial \mathbf{w_x}(n_x)}^T \end{pmatrix} \tag{52}$$

where n_u is the number of inputs and n_x is the number of states in the network.

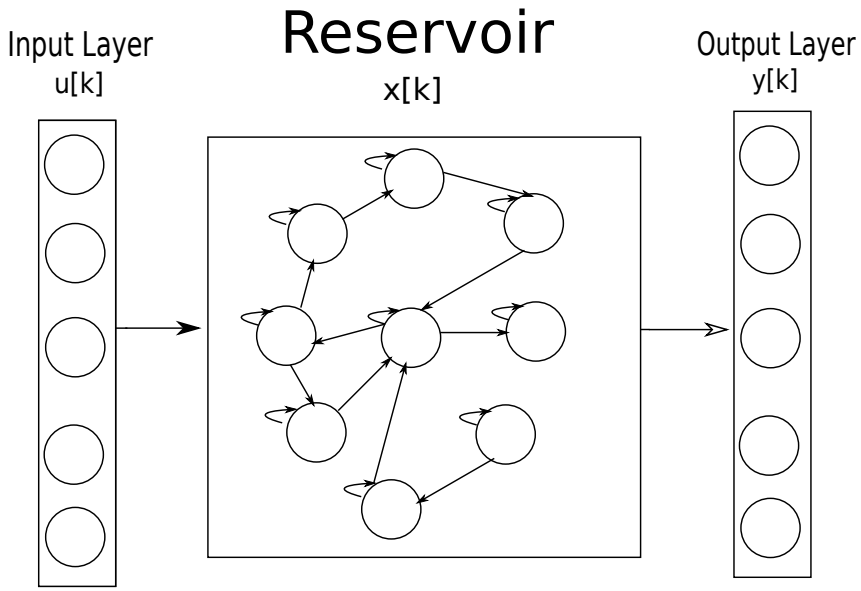


Figure 3 – Representation of an Echo State Network.

2.5 ECHO STATE NETWORKS (ESN)

An ESN is a type of recurrent neural network with useful characteristics for system identification (JAEGER et al., 2007), as it represents nonlinear dynamics well and the training consists in solving a linear least-squares problem of relatively low computational cost when compared to nonlinear optimization.

2.5.1 Model

Proposed by Jaeger and Haas (2004) and Jaeger (2001), the ESN is governed by the following discrete-time dynamic equations:

$$\begin{aligned}\mathbf{x}[k+1] &= (1-\gamma)\mathbf{x}[k] \\ &\quad + \gamma f(\mathbf{W}^{\mathbf{r}}_{\mathbf{r}}\mathbf{x}[k] + \mathbf{W}^{\mathbf{r}}_{\mathbf{i}}\mathbf{u}[k] + \mathbf{W}^{\mathbf{r}}_{\mathbf{b}} + \mathbf{W}^{\mathbf{r}}_{\mathbf{o}}\mathbf{y}[k]) \qquad (53)\\ \mathbf{y}[k+1] &= \mathbf{W}^{\mathbf{o}}_{\mathbf{r}}\mathbf{x}[k+1] \qquad (54)\end{aligned}$$

where the state of the reservoir neurons at time k is given by $\mathbf{x}[k]$; the current values of the input and output neurons are represented by $\mathbf{u}[k]$ and $\mathbf{y}[k]$, respectively; $0 < \gamma \leq 1$ is called leak rate (JAEGER et al., 2007), which governs the percentage of the current state $\mathbf{x}[k]$ that is transferred into the next state $\mathbf{x}[k+1]$. The weights are represented in the notation $\mathbf{W}^{\mathbf{to}}_{\mathbf{from}}$, with "$\mathbf{b}$", "$\mathbf{o}$", "$\mathbf{r}$", and "$\mathbf{i}$" sub and superscript meaning the bias, output, reservoir, and input neurons, respectively; and $f = \tanh(\cdot)$ is an activation function widely used in the literature, also called a base function in system identification theory (NELLES, 2020). Figure 3 depicts a standard architecture of an echo state network. However, there are other more complex and multi-layered architectures (MALIK; HUSSAIN; WU, Q. J., 2017).

The network has N neurons in the reservoir, which is the dimension of $\mathbf{x}[k]$ and is typically orders of magnitude higher than the number of network inputs. As long as regulariza-

tion is used in network training, N can be as large as needed, but at the expense of increased computation time to update the reservoir states as defined in (53). According to Jaeger (2002), the ESN with no output feedback connections (the output has no effect on the state), which is given by $\mathbf{W_o^r}$, has a memory capacity (MC) bounded by the number of neurons in the reservoir ($MC \leq N$), assuming that linear output units are used.

The recurrent reservoir should possess the so-called Echo State Property (ESP) (JAEGER, 2001), *i.e.*, a fading memory of its previous inputs, meaning that influences from past inputs on the reservoir states vanish with time. The ESP is guaranteed for reservoirs with $\tanh(\cdot)$ as the activation function, provided that the singular values of $\mathbf{W_r^r} < 1$. However, this condition limits the richness of the reservoir dynamical qualities, which discourages its use in practice. Note that all connections going to the reservoir are randomly initialized, usually according to the following steps:

1. Every weight of the network is initialized from a normal distribution $\mathcal{N}(0,1)$.
2. $\mathbf{W_r^r}$ is scaled so that its spectral radius ρ (Eigenvalue with the largest module) characterizes a regime able to create reservoirs with rich dynamical capabilities. It has been often observed that setting $\rho < 1$ in practice generates reservoirs with the ESP (JAEGER et al., 2007). However, reservoirs with $\rho > 1$ can still have the ESP since the effective spectral radius may still be lower than 1 (OZTURK; XU, D.; PRÍNCIPE, 2007; VERSTRAETEN, David; SCHRAUWEN, 2009).
3. $\mathbf{W_i^r}$ and $\mathbf{W_b^r}$ are multiplied by scaling factors f_i^r and f_b^r, respectively, affecting the magnitude of the input.

These scaling parameters, ρ, f_i^r, and f_b^r are crucial in the learning performance of the network, having an impact on the nonlinear representation and memory capacity of the reservoir (VERSTRAETEN, Davd et al., 2010). Also, low leak rates allow for higher memory capacity in reservoirs, while high leak rates should be favored for quickly varying inputs and/or outputs. The settings of these parameters should be such that the generalization performance of the network (loss on a validation set) is enhanced.

2.5.2 Training

While in standard RNNs all weights are trained iteratively using gradient descent (MOZER, 1989), ESNs restrict the training to the output layer $\mathbf{W_r^o}$. Because the echo state property is not insured with output feedback $\mathbf{W_o^r y}[k]$, this work favors reservoirs without feedback from the output (*i.e.*, $\mathbf{W_o^r} = 0$). Also, the inputs do not interfere directly in the output, as systems with direct transmission are less smooth and more sensitive to noise. To train an ESN, the input data $\mathbf{u}[k]$ is arranged in a matrix $\mathbf{U}$ and the desired output $\mathbf{d}[k]$ in vector $\mathbf{D}$ over a simulation time period, where each row $\mathbf{u}^T$ of $\mathbf{U}$ corresponds to a sample time k and its columns are related to the input units. For the sake of simplicity, assume that there are multiple inputs and only one output. The rows of $\mathbf{U}$ are input into the network reservoir

according to each sample time, thereby creating a data matrix $\mathbf{X}$ that contains the resulting sequence of states. Then, what is left is to apply the Ridge Regression algorithm (BISHOP, 2006) by using $\mathbf{X}$ as the input data matrix and $\mathbf{D}$ as the output data matrix. Ridge Regression results in solving the following linear system:

$$(\mathbf{X}^T\mathbf{X} - \lambda\mathbf{I})\mathbf{W}_\mathbf{r}^\mathbf{o} = \mathbf{X}^T\mathbf{D} \tag{55}$$

where λ is the Tikhonov regularization parameter, which serves to penalize the weight magnitude, avoiding overfitting. There are also methods to apply least squares training in an online way (NELLES, 2020), however these algorithms are not used in this work.

2.5.3 Deterministic Reservoir Computing

In ESNs, the non-trainable weights are decided randomly through probability distribution. Randomness can heavily affect the performance of a system identification. An investigation of possible solutions is proposed by Rodan and Tino (2010), where different topologies are proposed for the application.

One of the proposals of Rodan and Tino (2010) is to, instead of initializing $\mathbf{W}_\mathbf{r}^\mathbf{r}$ randomly, to initialize the weights in a way that the neurons are connected to form a circle, as in:

$$\mathbf{W}_\mathbf{r}^\mathbf{r}(i+1,i) = 1 \tag{56}$$

$$\mathbf{W}_\mathbf{r}^\mathbf{r}(1,N) = 1 \tag{57}$$

and the undefined elements of $\mathbf{W}_\mathbf{r}^\mathbf{r}$ are zero. The notation $\mathbf{A}(i,j)$ means the element in row i and column j of a generic $\mathbf{A}$ matrix. The work also proposes initializing $\mathbf{W}_\mathbf{i}^\mathbf{r}$ and $\mathbf{W}_\mathbf{b}^\mathbf{r}$ using a single value for every element in each weight, and choosing the signal randomly.

It is of notice that this structure has no internal feedback of each neuron (the information is passed cyclically). Another structure is proposed in Rodan and Tino (2010), where the non-zero elements of the initial $\mathbf{W}_\mathbf{r}^\mathbf{r}$ are:

$$\mathbf{W}_\mathbf{r}^\mathbf{r}(i+1,i) = 1 \tag{58}$$

$$\mathbf{W}_\mathbf{r}^\mathbf{r}(i,i) = 1 \tag{59}$$

$$\mathbf{W}_\mathbf{r}^\mathbf{r}(1,N) = 1 \tag{60}$$

which is basically the structure of the cyclic topology, but with internal feedback at each neuron added, as clarified in Figure 4.

In a previous work, Jean P. Jordanou, Eric Aislan Antonelo, and Camponogara (2019) implemented a two-wells one riser model to test a control structure involving Echo State Networks. This two-wells one riser platform was employed as a case study to compare the results of the conventional, random ESN with these 2 topologies. The experiment ran 5 simulation of 50000 time steps, where the 40000 first time steps are used as training data,

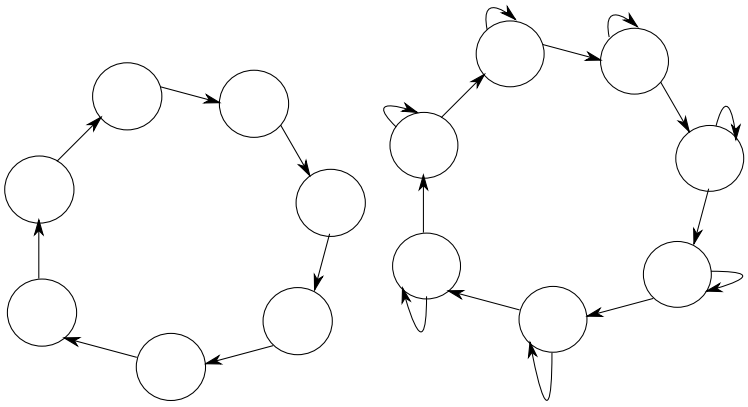


Figure 4 – Illustration of two topologies for deterministic reservoirs. On the left side, the cyclic reservoir, and on the right side, the cyclic reservoir with internal feedback.

and the remaining 10000 as test data. The error metrics in this case is defined as the mean absolute error in the validation phase. The hyper-parameters are the same for each network, being defined in Table 1. Since the purpose of this experiment was to compare the deterministic with the random approach, the parameters were selected heuristically so that both ESNs can perform system identification in the two wells and one riser plant with a small error.

Table 1 – Parameters values selected for the deterministic reservoir experiment.

Parameter	
N: Reservoir size	400
γ: Leak Rate	0.9
ρ: Spectral Radius of $\mathbf{W_r^r}$	0.999
f_i^r: Scaling Factor of $\mathbf{W_i^r}$	0.15
f_b^r: Scaling Factor of $\mathbf{W_b^r}$	0.1

The results are shown in Table 2. The performance of a random reservoir is superior in both mean and considering the mean alongside the standard deviation with respect to its deterministic counterpart, albeit by a small margin. The internal feedback has worsened the performance in comparison to the original proposal by Rodan and Tino (2010), probably due to the larger exchange of information, as there are more non-zero weights.

Table 2 – Results for the deterministic ESN experiment, with the mean absolute validation taken over 5 training runs for each network. This number of runs was selected because it was low enough to be easy for runtime, and high enough so that data is skewed to the mean, with small standard deviation.

	Validation Error	
	Mean	Std
Random Reservoir	0.019	0.0033
Cyclic Reservoir	0.022	0.001
Cyclic Reservoir (Internal Feedback)	0.031	0.00018

Another proposal is a middle ground between a deterministic and a random reservoir: Utilizing a cyclic reservoir, however with the input and bias weights decided randomly, with a normal distribution akin to the original ESN. This is coined as a semi-deterministic reservoir. Using the same experimental conditions as the experiment for the deterministic reservoirs, the results obtained with semi-deterministic networks are presented in Table 3. The result for the cyclic reservoir does not differ much, however the cyclic reservoir with internal neuron feedback has shown a better result than its deterministic counterpart. The main implication of this result is that it is viable to mix both the deterministic and a random approach to form an ESN with consistent performance.

Table 3 – Results for the semi-deterministic ESN experiment, with the mean absolute validation taken over 5 training runs for each network.

	Validation Error	
	Mean	Std
Cyclic Reservoir	0.022	0.0017
Cyclic Reservoir (Internal Feedback)	0.027	0.0026

The purpose of this evaluation was to compare a deterministic to a nondeterministic reservoir in terms of mean absolute error. For computational complexity, it is expected that deterministic reservoir will perform better using sparse matrix operations, as these reservoirs are sparse by definition. However, it might also be interesting to perform experiments evaluating computational time and number of float point operations to compare both reservoirs in those aspects.

2.6 MODEL ORDER REDUCTION

This section makes a brief overview of Model Order Reduction (MOR) methods. Model Order Reduction is a set of techniques employed to find a low order model that approximates high order systems, and thus easy to compute and apply in optimization problems.

2.6.1 Davison Method

Model Order Reduction of linear systems is a well solved problem in the literature (SIVARAMAKRISHNAN, 2013), which have many associated methods and algorithms to calculate a smaller state-space model, approximating a given higher order system. One of such methods is the simple truncation of the smallest Eigenvalues, by means of transforming the system into Jordan form, truncating the matrix, eliminating the smallest Eigenvalues, and then transforming back. This method is referred to as the Davison method.

Despite reproducing transient behavior well, the main disadvantage of the Davison method is that nothing prevents the reduced order model from having a steady state error. Another version of the Davison method (SIVARAMAKRISHNAN, 2013) proposes using a

diagonal matrix to fix the steady state error, where, assuming an univariate system:

$$d_j = \begin{cases} \frac{[\mathbf{A}^{-1}\mathbf{B}]_j}{[{\mathbf{A_r}}^{-1}\mathbf{B_r}]_j} & [{\mathbf{A_r}}^{-1}\mathbf{B_r}]_j \neq 0 \\ 1 & [{\mathbf{A_r}}^{-1}\mathbf{B_r}]_j = 0 \end{cases} \tag{61}$$

where d_j are the elements of a diagonal matrix $\mathbf{D}$ with size l, $\mathbf{A_r}$ and $\mathbf{B_r}$ are the resulting reduced order state space matrices from an arbitrary, linear state space system, and $[\cdot]_j$ is the j^{th} element of the vector inside. Also, it is assumed that the states are sorted in terms of Eigenvalue influence in the system, so that the j-element of the original system is the j-element of the reduced system. Each diagonal element d_j serves as a steady state error correction gain (SIVARAMAKRISHNAN, 2013), and the resulting system has a similar steady-state response while maintaining similar dynamic behavior.

There are, of course, other linear model order reduction variants in the literature (SIVARAMAKRISHNAN, 2013), however they either involve Eigenvalue manipulation and assessment on the frequency domain.

2.6.2 Proper Orthogonal Decomposition

The Proper Orthogonal Decomposition (POD) is a method to find a linear transformation (CHEN, C.-T., 1998) $\mathbf{T}$ for a given linear system that maps a high state space into a smaller one, namely:

$$\mathbf{x} = \mathbf{T}\mathbf{z} \tag{62}$$

where $\mathbf{x}$ is a vector of dimension n and $\mathbf{z}$ is a vector of dimension $m \ll n$. Given a linear system:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t),$$

the reduced state space system would be computed by:

$$\dot{\mathbf{z}}(t) = \mathbf{T}^T\mathbf{A}\mathbf{T}\mathbf{z}(t) + \mathbf{T}^T\mathbf{B}\mathbf{u}(t) \tag{63}$$

This transformation is akin to a similarity transformation, with the main difference is that $\mathbf{T}$ lacks an inverse matrix, however since $\mathbf{T}$ is orthonormal ($\mathbf{T}^T\mathbf{T} = \mathbf{I}$), the analogous use of a transpose is justified in this case.

A widely used method to find $\mathbf{T}$ is to gather snapshots of a given dynamical system response, akin to gathering data in a machine learning problem. The columns of the snapshot matrix $\mathbf{X}$ are the states $\mathbf{x}(t)$ at time $t = T_s k$, with T_s being the sampling time of the snapshot. The notation $\mathbf{x}[k]$ represents $\mathbf{x}(T_s k)$, thus omitting the sampling time. Then, $\mathbf{T}$ must minimize the error induced by projecting the original state onto the reduced space and back, which leads to the following error function:

$$E(\mathbf{T}) = \sum_{k=1}^{N} (\mathbf{x}[k] - \mathbf{T}\underbrace{\mathbf{T}^T\mathbf{x}[k]}_{\mathbf{z}[k]}) \tag{64}$$

where N is the number of snapshots. The second term is $\mathbf{x}$ projected onto the reduced space of $\mathbf{z}$, and then projected back. In fact, the optimal way to calculate a transformation $\mathbf{T}$ that minimizes $E(\mathbf{T})$ is through singular value decomposition (SVD) (SUN; XU, M.-h., 2017).

By applying SVD, $\mathbf{X}$ is decomposed in the following matrices:

$$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{X} \tag{65}$$

where $\mathbf{U}$ contains the left singular vectors and has dimension $n \times n$, $\mathbf{\Sigma}$ contains the singular values and has dimension $n \times n$, with only N non-zero rows. Consider that $\mathbf{\Sigma}$ is sorted from the largest to the smallest singular value. The right singular vector matrix $\mathbf{V}$ is not used for POD.

The linear transformation matrix $\mathbf{T}$ that minimizes $E(\mathbf{T})$ corresponds to the first columns of $\mathbf{U}$ associated with the largest singular vectors of the SVD. A truncation must ensure that the reduced system energy is close to the original. The following equation measures the effectiveness of the truncation in that regard:

$$\epsilon = \frac{\sum_{j=1}^{n_r} \sigma_j}{\sum_{j=1}^{N} \sigma_j} \tag{66}$$

where σ_j is the j-highest singular value, and n_r is the reduced state dimension. One then finally obtains an appropriate $\mathbf{T}$ for the space dimension reduction, which is different from the Davidson method that directly uses the Eigenvalues of a given linear system.

While this very discussion holds for the model order reduction of linear systems, Chapter 4 elaborates on POD for nonlinear systems.

2.7 SUMMARY

In this chapter lies the description of the fundamental theory behind the rest of this dissertation. The portion related to stability is important for understanding Chapter 5, and the portion related to model order reduction is related to Chapter 4.

3 ESN BASED PRACTICAL NMPC

This chapter describes the ESN-PNMPC methodology developed in (JORDANOU, Jean P. et al., 2018), and further elaborated on (JORDANOU, Jean Panaioti; ANTONELO, Eric Aislan; CAMPONOGARA, 2022) with updated results on new case studies, which serves as a basis for this work. The results shown in this chapter were also published in (JORDANOU, Jean Panaioti; ANTONELO, Eric Aislan; CAMPONOGARA, 2022).

3.1 INTRODUCTION

In the real world, the problem of controlling industrial plants without a known model is very common, and also a challenging task, demanding efficient data-driven strategies, generally involving black-box modeling.

Out of several possible control methods from the literature that can be applied to systems without known models (HOU; WANG, Z., 2013), model predictive control (MPC) is one technique that has become standard for multivariate control in industry and academia (CAMACHO; BORDONS, 1999). Since its inception in the 1970s, MPC was successfully applied in the oil and gas (JORDANOU, Jean P. et al., 2018), aerospace (EREN et al., 2017) and process industries, as well as in robotics (NASCIMENTO; DÓREA; GONÇALVES, 2018). MPC has the advantage of simplicity and intuitiveness in comparison with other approaches, as little knowledge about control theory is required to apply the method in an industrial level (CAMACHO; BORDONS, 1999). The core idea of MPC is to control a system by employing a prediction model, using it to solve an optimization problem in a receding horizon approach at every iteration. For linear models, the optimization problem to find the control action is a Quadratic Programming (QP) problem, akin to efficient linear MPC strategies such as Dynamic Matrix Control (DMC) and Generalized Predictive Control (GPC) (CAMACHO; BORDONS, 1999). On the other hand, complex industrial plants are better represented by nonlinear dynamic models, for which MPC may be challenging to apply in a rapid straightforward manner, due to the need to solve a Nonlinear Programming Problem (NLP) at each iteration.

RNNs are considered universal approximators of dynamical systems (SALEHINEJAD et al., 2017) and can serve as black-box models for nonlinear MPC. System identification with these RNNs is based on historical data, *i.e.*, on a dataset of input-output pairs of the nonlinear plant. Training RNNs to model the plant is equivalent to minimizing an error (cost) function with respect to the weights of the network. For regression, this function is usually the mean squared error between the true and predicted outputs. Traditional RNNs use the error function gradient to iteratively update the network weights, which does not guarantee global optimality. Besides, this learning procedure is prone to become disrupted by bifurcations (DOYA, 1992) and is computationally costly.

An alternative way to model dynamic systems is by Reservoir Computing (RC) (SCHRAUWEN; VERSTRAETEN, David; VAN CAMPENHOUT, 2007; JAEGER; HAAS, 2004). The basic as-

sumption of RC is to first project an input space into a high-dimensional dynamic nonlinear space, the reservoir space, and then use the resulting temporal features as new inputs in linear regression tasks. The reservoir is usually given by a nonlinear RNN with fixed weights, while a second linear readout output layer is subject to training (Fig. 3). As only the output layer of the RNN is trained, the learning is fast and has a global optimum corresponding to the least squares solution, overcoming previous limitations of gradient-based training for RNNs. When the RNN is composed of $\tanh$ units forming an analog network, the resulting network is also called an ESN (JAEGER, 2001; JAEGER; HAAS, 2004). This linear training also enables the model to be updated online with new data through algorithms such as Recursive Least Squares (RLS) (JORDANOU, Jean P.; ANTONELO, Eric Aislan; CAMPONOGARA, 2019).

Other approaches use Radial Basis Function (RBF) networks (AYALA et al., 2020) or fuzzy logic (OUALI; LADJAL, 2020) for system identification. Although RBF nets have efficient training, in order to model system's dynamics, delayed signals need to be introduced in the input layer since a RBF net has no inner dynamics as does the RC network. The benefit of RC here is to automatically take into account the nonlinear dynamics of the process, neither requiring to increase the input dimension, nor to find the right size of the time window. In general, fuzzy systems also need the same sort of delayed signals to model dynamics and, although linguistic rules can express operator knowledge, the size of the knowledge base increases exponentially with the number of inputs.

To deal with the issue of solving a NLP per iteration, the Practical Nonlinear Model Predictive Controller (PNMPC) framework (PLUCÊNIO, A. et al., 2007) employs a fully nonlinear model to compute the free response of the system, while applying a first-order Taylor expansion in the model to approximate the forced response, which is the sensitivity of the response to the control action. This expansion allows full retention of model precision in relation to the nonlinear system for the free response, unlike in (PAN; WANG, J., 2012), where the ESN is linearized as a full state-space system, therefore losing model precision along the prediction horizon. In addition, because the derivative with respect to the state has to be computed in (PAN; WANG, J., 2012), the computational and memory demands are high when compared to our proposal based on PNMPC. PNMPC was shown to perform well in several applications, such as in the control of oil and gas processes (PLUCÊNIO, Agustinho, 2013), but with one drawback: since the model gradient is either not stipulated or assumed very expensive to obtain, a finite difference method is necessary in order to compute the derivative terms involved. This incurs a high computational cost when the number of process inputs and outputs is large, due to the combinatorial nature of the finite differences computation.

In this context, the ESN-PNMPC is a Reservoir Computing (RC)-based control framework that speeds up the NMPC of processes by combining the PNMPC principle of splitting the forced and the free responses, with a trained ESN as the dynamic system model, yielding the so called ESN-PNMPC architecture. Specifically, our proposal substitutes the finite difference method of PNMPC for a fast analytical computation of the derivative in terms of the ESN.

3.1.1 Practical Nonlinear Model Predictive Control (PNMPC)

Developed by (PLUCÊNIO, A. et al., 2007), the Practical Nonlinear Model Predictive Controller (PNMPC) is a method that, through a first order Taylor expansion, separates a nonlinear dynamic model into a free response and a forced response. An advantage of this approximation strategy is that the free response has full precision in relation to the whole nonlinear system. The separation between free response and forced response is normally a characteristic of linear systems due to the homogeneity property. However, by obtaining the first order Taylor expansion of a function with respect to an input u, one can better explain the intuition behind the PNMPC:

$$y = f(u) \approx f(\bar{u}) + \frac{\partial f(\bar{u})}{\partial u}\Delta u \tag{67}$$

where $u = \bar{u} + \Delta u$, $\bar{u}$ is a fixed point, and Δu is a variation around that same point. The nonlinear function $f(u)$ is split into: (i) a zeroth-order term $f(\bar{u})$ which is analogous to the PNMPC free response as it computes the current value of f with only $\bar{u}$ as input; and (ii) a first-order term which carries the same intuition as the forced response, as it is linear over Δu. Note that here $f(\bar{u})$ will be simulated by running ((53)) without any approximation in the ESN-PNMPC method that will be presented, retaining full nonlinear precision for problems of rich dynamics.

The PNMPC has a computational advantage over a standard Nonlinear MPC because the resulting control problem to be solved per iteration is a quadratic program (QP), similar to a linear MPC, whereas in the full nonlinear case a nonlinear program (NLP) would be solved. This approach is advantageous when time constraints are in place. The PNMPC is more or less akin to performing a one-step SQP in a quadratic cost function problem using a nonlinear model. Assuming a dynamic system in the form:

$$\mathbf{x}[k+i] = \mathbf{f}(\mathbf{x}[k+i-1],\mathbf{u}[k+i-1]) \tag{68}$$

$$\mathbf{y}[k+i] = \mathbf{g}(\mathbf{x}[k+i]) \tag{69}$$

$$\mathbf{u}[k+i-1] = \mathbf{u}[k-1] + \sum_{j=0}^{i-1} \Delta\mathbf{u}[k+j] \tag{70}$$

The vectors for output predictions are collected in $\widehat{\mathbf{Y}}$[1]. Here, the control increment $\boldsymbol{\Delta}\mathbf{U}$, and free response $\mathbf{F}$ are defined in PNMPC as follows:

$$\widehat{\mathbf{Y}} = \mathbf{G} \cdot \boldsymbol{\Delta}\mathbf{U} + \mathbf{F} \tag{71}$$

$$\boldsymbol{\Delta}\mathbf{U} = \begin{pmatrix} \boldsymbol{\Delta}\mathbf{u}[k] \\ \boldsymbol{\Delta}\mathbf{u}[k+1] \\ \vdots \\ \boldsymbol{\Delta}\mathbf{u}[k+N_u-1] \end{pmatrix} \tag{72}$$

[1] $\widehat{\mathbf{Y}}$ is the notation for the model prediction in MPC theory.

$$\mathbf{F} = \begin{pmatrix} \mathbf{g}(\mathbf{f}(\mathbf{x}[k],\mathbf{u}[k-1])) \\ \mathbf{g}(\mathbf{f}(\mathbf{x}[k+1],\mathbf{u}[k-1])) \\ \vdots \\ \mathbf{g}(\mathbf{f}(\mathbf{x}[k+N_y-1],\mathbf{u}[k-1])) \end{pmatrix} \tag{73}$$

where N_y is the prediction horizon, N_u is the control horizon, and matrix $\mathbf{G}$ is the Jacobian of the state equations at the free response.

The vector $\mathbf{\Delta U}$ consists of the concatenation of each control increment applied to the calculation of the predictions up to $k = N_u$. The vector $\widehat{\mathbf{Y}}$ gives all of the predictions performed by the model from $k = 0$ to $k = N_y$. Notice that for a given time $k + i$ the free response depends on the system state at that time, $\mathbf{x}[k + i]$, and only on the input at time $k - 1$ because the control signals remain constant ($\mathbf{\Delta U} = 0$). As the vector $\mathbf{F}$ contains all the free responses, the term $\mathbf{G} \cdot \Delta\mathbf{U}$ is the forced response over the prediction horizon.

The Jacobian of the state equations is defined as:

$$\mathbf{G} = \begin{pmatrix} \frac{\partial \mathbf{y}[k+1]}{\partial \mathbf{u}[k]} & 0 & \cdots & 0 \\ \frac{\partial \mathbf{y}[k+2]}{\partial \mathbf{u}[k]} & \frac{\partial \mathbf{y}[k+2]}{\partial \mathbf{u}[k+1]} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}[k+N_y]}{\partial \mathbf{u}[k]} & \frac{\partial \mathbf{y}[k+N_y]}{\partial \mathbf{u}[k+1]} & \cdots & \frac{\partial \mathbf{y}[k+N_y]}{\partial \mathbf{u}[k+N_u-1]} \end{pmatrix}$$

The derivatives inside $\mathbf{G}$ are taken with respect to $\Delta\mathbf{u}[k + i] = 0, \forall i < N_u$, and $\mathbf{u}$ represents the manipulated variable vector. This structure is a vectorized form of the prediction along the horizon, which is similar to the vectorization of predictions in the DMC and GPC (CAMACHO; BORDONS, 1999).

The equations above derive from the first-order Taylor series expansion in relation to the manipulated variables, whereby the free-response $\mathbf{F}$ retains the nonlinearity, whereas the forced-response is linearized so that the control increment is calculated via a QP. The matrix $\mathbf{G}$ is a result of that linearization, as each line corresponds to the first order term of the Taylor series w.r.t. the control increment at a certain instant in time.

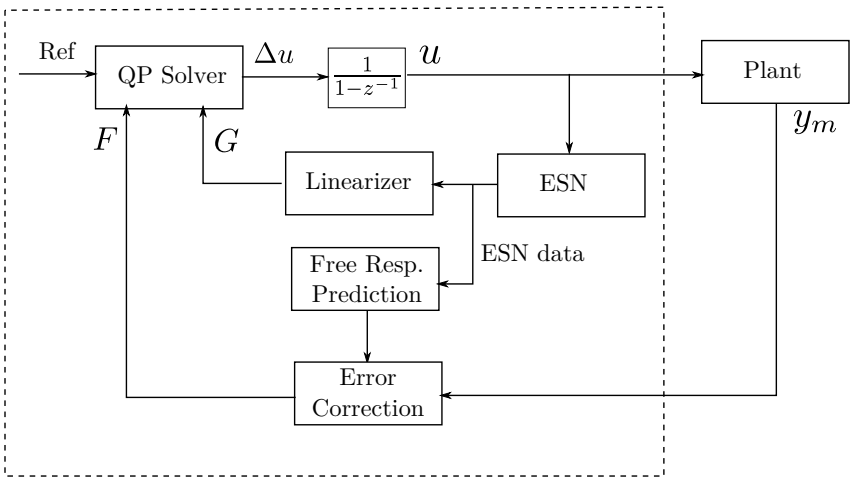
As (PLUCÊNIO, A. et al., 2007) assumes a generic nonlinear system, estimates for the derivatives are calculated with a finite-difference method, which inherently suffers from combinatorial explosion when multiple variables are involved. To this end, the following section proposes the ESN-PNMPC scheme to overcome the aforementioned limitation, enabling fast computation of linearized models on the fly.

3.1.2 ESN-PNMPC

3.1.2.1 Overview

The proposed ESN-PNMPC scheme consists of integrating an ESN as the state space model for PNMPC (see Fig. 5). The ESN model allows the analytical computation of derivatives, which drastically reduces the computation time by mitigating the limitations of finite differences.

Thus, the solution of the QP is the only computationally expensive aspect of the proposed algorithm.



ESN-PNMPC

Figure 5 – Block diagram representation of the ESN-PNMPC. The ESN block represents an ESN trained to mimic the plant, which is used by the Linearizer block to compute $\mathbf{G}$ using the ESN Jacobian, and by the Free Response Prediction block. The Error Correction block provides an integrated filter that computes the correction factor, while the QP Solver block handles the resulting optimization problem.

3.1.2.2 Linearizer – Forced Response Derivation

In order to compute the derivatives of the output $\mathbf{y}$ of the dynamical system with respect to the input $\mathbf{u}$, the chain rule is applied:

$$\frac{\partial \mathbf{y}[k+i]}{\partial \Delta \mathbf{u}[k+j]} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}[k+i]} \frac{\partial \mathbf{x}[k+i]}{\partial \Delta \mathbf{u}[k+j]} \tag{74}$$

$$\frac{\partial \mathbf{x}[k+i]}{\partial \Delta \mathbf{u}[k+j]} = \frac{\partial \mathbf{f}}{\partial \Delta \mathbf{u}[k+j]} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}[k+i-1]} \frac{\partial \mathbf{x}[k+i-1]}{\partial \Delta \mathbf{u}[k+j]} \tag{75}$$

The implication in Equation (75) is that $\mathbf{G}$ is recursively built by forward propagating from $i = 0$ to $i = N_y$. Considering that the dynamic matrix is evaluated at $\mathbf{\Delta U} = 0$, all the derivatives along the horizon can be evaluated with respect to the control input $\mathbf{u}[k-1]$. Therefore,

$$\left.\frac{\partial \mathbf{f}(\mathbf{x}[k+i])}{\partial \Delta \mathbf{u}}\right|_{\mathbf{\Delta u}=0} = \left.\frac{\partial \mathbf{f}(\mathbf{x}[k+i])}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}[k-1]}.$$

As long as $i > j$ (line i and column j of matrix $\mathbf{G}$), the control increment $\mathbf{\Delta u}[k+j]$ has influence on the output at time instant $k+i$ because the control signal was input in a previous instant. From Eq. (70), the control increment $\mathbf{\Delta u}[k+j]$ has equal influence on the state equation for state $\mathbf{x}[k+i]$ for all j, and therefore $\frac{\partial \mathbf{f}(\mathbf{x}[k+i])}{\partial \mathbf{\Delta u}[k+j]}$ has the same value regardless

of j. Therefore the notation $\mathbf{J}(i) = \frac{\partial \mathbf{f}(\mathbf{x}[k+i])}{\partial \Delta \mathbf{u}}$ can be used to represent any $\frac{\partial \mathbf{f}(\mathbf{x}[k+i])}{\partial \mathbf{\Delta u}[k+j]}$. Further, $\frac{\partial \mathbf{f}(\mathbf{x}[k+i])}{\partial \mathbf{x}[k+i]}$ is denoted as $\mathbf{S}(i)$ to simplify the developments.

By adopting the above notation into Eqs. (74)-(75), the partial derivatives are recast in a recursive form:

$$\mathbf{G}_{ij} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}[k+i]} \frac{\partial \mathbf{x}[k+i]}{\partial \mathbf{\Delta u}[k+j]} \tag{76}$$

$$\frac{\partial \mathbf{x}[k+i]}{\partial \mathbf{\Delta u}[k+j]} = \begin{cases} \mathbf{J}(i-1) + \mathbf{S}(i-1) \frac{\partial \mathbf{x}[k+i-1]}{\partial \mathbf{\Delta u}[k+j]} & i > j \\ \mathbf{J}(i-1) & i = j \\ 0 & i < j \end{cases} \tag{77}$$

where $\mathbf{G}_{ij}$ represents the block element of $\mathbf{G}$ at row i and column j. The construction of $\mathbf{G}$ starts with $i = 1$, where the initial condition $\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \mathbf{J}(0)$ is input to $\mathbf{G}_{1,1}$. As $i < j$ for the rest of the row, all terms $\mathbf{G}_{1,(j \neq 1)} = 0$. For the subsequent rows, information used to calculate the previous row is used for the next, until $i = j$, where $\mathbf{G}_{i,j} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \mathbf{J}(i-1)$ and $i < j$, where $\mathbf{G}_{i,j} = 0$. This calculation ends when $i = N_y$.

Because an ESN is trained offline to serve as the prediction model, the model derivatives are well defined (PAN; WANG, J., 2012; XIANG et al., 2016), being given as follows:

$$\begin{aligned} \frac{\partial \mathbf{g}}{\partial \mathbf{x}} &= \mathbf{W}_{\mathbf{r}}^{\mathbf{o}} \\ \mathbf{J}(i) &= \frac{\partial \mathbf{f}}{\partial \mathbf{z}_i} \mathbf{W}_{\mathbf{i}}^{\mathbf{r}} \\ \mathbf{S}(i) &= (1-\gamma)\mathbf{I} + \gamma \frac{\partial \mathbf{f}}{\partial \mathbf{z}_i} (\mathbf{W}_{\mathbf{r}}^{\mathbf{r}} + \mathbf{W}_{\mathbf{o}}^{\mathbf{r}} \mathbf{W}_{\mathbf{r}}^{\mathbf{o}}) \\ \mathbf{z}_i &= \mathbf{W}_{\mathbf{r}}^{\mathbf{r}} \mathbf{a}[k+i] + \mathbf{W}_{\mathbf{i}}^{\mathbf{r}} \mathbf{u}[k-1] + \mathbf{W}_{\mathbf{o}}^{\mathbf{r}} \mathbf{W}_{\mathbf{r}}^{\mathbf{o}} \mathbf{a}[k+i] + \mathbf{W}_{\mathbf{b}}^{\mathbf{r}} \end{aligned}$$

where the network state $\mathbf{a}[k]$ corresponds to the model state $\mathbf{x}[k]$, and $\mathbf{z}_i$ is an auxiliary variable representing the argument of the activation function in the ESN model. Since, in this work, $\mathbf{f} = \tanh(\cdot)$ is a function applied to each entry of the vector $\mathbf{z}_i$ (*i.e.*, element-wise function), $\frac{\partial \mathbf{f}}{\partial \mathbf{z}_i}$ is a diagonal matrix with all nonzero elements being $[1 - \tanh^2(\mathbf{z}_i)]$.

Summarizing, the trained ESN model is used to compute the free-response predictions and analytically calculate the Taylor approximation to formulate the QP on the fly, which is solved at every iteration. Besides errors inherent to disturbances and modeling, additional errors are incurred in the predictive model by the Taylor expansion. In (PAN; WANG, J., 2012), a supervised learning strategy is embedded in a NMPC to estimate the Taylor expansion error, whereby the actual and predicted outputs are used as information. In the PNMPC, the Taylor expansion error is considered part of the disturbance model.

3.1.2.3 Error Correction

To treat disturbances and modeling errors, (PLUCÊNIO, A. et al., 2007) advocates the use of a low-pass discrete filter on the error between the current measured output and the current prediction, which is computed as part of the free response. If the model was identical

to the plant and no disturbances were applied, the presence of the filter and the proposed closed-loop framework would be not different than an open-loop implementation. The filter serves to slow down the error response from the point of view of the controller, thus increasing robustness but sacrificing response speed according to the filter cutting frequency (CAMACHO; BORDONS, 1999). In practice, this is merely a different perspective to the problem, since the approach taken by (PAN; WANG, J., 2012) is equivalent to using a variable static gain as a filter.

Adding the filter, the free-response becomes:

$$\mathbf{F} = \begin{pmatrix} \mathbf{g}(\mathbf{f}(\mathbf{x}[k],\mathbf{u}[k-1])) \\ \mathbf{g}(\mathbf{f}(\mathbf{x}[k+1],\mathbf{u}[k-1])) \\ \vdots \\ \mathbf{g}(\mathbf{f}(\mathbf{x}[k+N_y-1],\mathbf{u}[k-1])) \end{pmatrix} + \mathbf{1}\boldsymbol{\eta}[k] \tag{78}$$

$$\Delta\boldsymbol{\eta}[k] = (1-\omega)(\widehat{\mathbf{y}}[k|k-1] - \mathbf{y_m}[k]) + \omega\Delta\boldsymbol{\eta}[k-1] \tag{79}$$

$$\boldsymbol{\eta}[k] = \boldsymbol{\eta}[k-1] + K\Delta\boldsymbol{\eta}[k] \tag{80}$$

$$\widehat{\mathbf{y}}[k|k-1] = \mathbf{g}(\mathbf{f}(\mathbf{x}[k-1],\mathbf{u}[k-1])) + \boldsymbol{\eta}[k-1] \tag{81}$$

where $\mathbf{y_m}[k]$ is the measured variable from the plant, K is the integrator gain, and ω is the leak rate of the filter, which is used to enhance the robustness capability of the controller, $\widehat{\mathbf{y}}[k|k-1]$ is the output prediction for time k calculated at time $k-1$, using $\boldsymbol{\eta}[k-1]$ as correction factor, and $\boldsymbol{\eta}[k]$ is the currently calculated correction factor yielded by the filter.

The work in (PLUCÊNIO, Agustinho, 2013) proposes a method on how to tune the filter parameters. To simplify this formulation, assume that the system being controlled has only one output, however one can either use the same filter for different outputs, or apply different filters with the same formulation. The definition of the a priori error $\epsilon[k]$ is:

$$\epsilon[k] = y_m[k] - y_{esn}[k|k-1] \tag{82}$$

where $y_{esn}[k|k-1]$[2] is the output computed by the ESN using information given at time $k-1$, and $\{y_m, y_{esn}[k|k-1]\}$ are not bold because they are assumed scalars. The a posteriori error $e[k]$ is defined as:

$$e[k] = y_m[k] - \widehat{y}[k|k-1] \tag{83}$$

A transfer function between the a priori error and the a posteriori error dictates the dynamics for the correction. To find it, the first step is to expand Eq. (83), and substitute Eq. (82) inside, obtaining:

$$e[k] = \epsilon[k] - \eta[k-1] \tag{84}$$

[2] y_{esn} is equivalent to the output y in equation ((54)).

In terms of the a posteriori error, by putting Eq. (83) in (79) the equations that define the correction factor become:

$$\Delta\eta[k] = \omega\Delta\eta[k-1] + (1-\omega)e[k] \tag{85}$$

$$\eta[k] = \eta[k-1] + K\Delta\eta[k] \tag{86}$$

Applying the z transform into Eqs. (84)-(86) results in the following equation:

$$e(z) = \epsilon(z) - z^{-1}\eta(z) \tag{87}$$

$$\eta(z) = \frac{K}{1-z^{-1}}\Delta\eta(z) \tag{88}$$

$$\Delta\eta(z) = \frac{1-\omega}{1-\omega z^{-1}}e(z) \tag{89}$$

and joining the resulting equations into one forms the transfer function for $\frac{e(z)}{\epsilon(z)}$:

$$\frac{e(z)}{\epsilon(z)} = \frac{z^2-(\omega+1)z+\omega}{z^2-(\omega+1-K(1-\omega))z+\omega} \tag{90}$$

It is noticed that, since 1 is a zero in this transfer function, the steady state error is guaranteed to be 0 for a constant a prior error (PLUCÊNIO, Agustinho, 2013). Our next step is to define K and ω so that the denominator of the transfer function is equal to a characteristic polynomial, which is defined as:

$$p(z) = (z-a)^2 = z^2 - 2az + a^2$$

where $a = [0,1)$ is the root of the polynomial, and the desired location of the poles for the transfer function. The polynomial root relates to the characteristic polynomial in the form of:

$$\omega = a^2 \tag{91}$$

$$K = \frac{a^2-2a+1}{1-a^2} \tag{92}$$

With these relations, K and ω are defined based only on the desired pole for the error correction dynamics. Without a low pass filter, or when $\omega = a = 0$, then $K = 1$.

3.1.2.4 QP Problem

If the quadratic error is used as the cost function for the NMPC, the equations in matrix form become:

$$J = (\mathbf{Y}_{ref} - \widehat{\mathbf{Y}})^T\mathbf{Q}(\mathbf{Y}_{ref} - \widehat{\mathbf{Y}}) + \mathbf{\Delta U}^T\mathbf{R}\mathbf{\Delta U}$$

in which $\mathbf{Y}_{ref}$ is the output reference over the prediction horizon, and $\mathbf{Q}$ and $\mathbf{R}$ are diagonal matrices with the output and control weighting, whose utility is to express a variable's importance in the cost function.

Since the predicted output is stated in a form akin to the GPC and DMC strategies for MPC (CAMACHO; BORDONS, 1999), the cost function is formulated as follows:

$$\begin{aligned} J &= \mathbf{\Delta U}^T \mathbf{H} \mathbf{\Delta U} + \mathbf{c}^T \mathbf{\Delta U} \\ \mathbf{H} &= \mathbf{G}^T \mathbf{Q} \mathbf{G} + \mathbf{R} \\ \mathbf{c} &= \mathbf{G}^T \mathbf{Q}^T (\mathbf{Y}_{ref} - \mathbf{F}) \end{aligned}$$

In receding horizon control problems, saturation and rate limiting constraints are typically introduced to the optimization problem to ensure a feasible operation. Such saturation constraints are formulated as follows:

$$\mathbf{1u}_{\min} - \mathbf{1u}[k-1] \leq \mathbf{T}\mathbf{\Delta U} \leq \mathbf{1u}_{\max} - \mathbf{1u}[k-1]$$

where $\mathbf{1}$ is a vector composed only of ones which matches the dimension and form of $\mathbf{\Delta U}$. If the problem was structured as a SISO (Single-Input Single-Output), $\mathbf{T}$ would be a lower triangular matrix. As our work concerns a MIMO (Multi-Input Multi-Output) system and the prediction outputs for a sample time are stacked as vectors in $\widehat{\mathbf{Y}}$, the matrix $\mathbf{T}$ is defined as:

$$\mathbf{T} = \begin{pmatrix} \mathbf{I}_{n_{in}} & \mathbf{0}_{n_{in}} & \mathbf{0}_{n_{in}} \\ \mathbf{I}_{n_{in}} & \ddots & \mathbf{0}_{n_{in}} \\ \mathbf{I}_{n_{in}} & \mathbf{I}_{n_{in}} & \mathbf{I}_{n_{in}} \end{pmatrix}$$

where $\mathbf{I}_{n_{in}}$ is a n_{in} sized identity matrix and $\mathbf{0}_{n_{in}}$ is a n_{in} sized square matrix of zeros, with n_{in} being the number of system inputs. Summarizing, $\mathbf{T}$ is a block triangular matrix of n_{in}-dimensional square matrices, where each column of the block matrix represents an instant in the prediction horizon.

Likewise, rate limiting constraints are stated as follows:

$$\mathbf{\Delta U}_{\min} \leq \mathbf{I}\mathbf{\Delta U} \leq \mathbf{\Delta U}_{\max}$$

where $\mathbf{I}$ is the identity matrix, with dimension $n_{in}N_u$.

Summarizing, the optimization problem solved per iteration is stated as follows:

$$\begin{aligned} \min_{\mathbf{\Delta U}} \; & J(\mathbf{\Delta U}) = \mathbf{\Delta U}^T \mathbf{H} \mathbf{\Delta U} + \mathbf{c}^T \mathbf{\Delta U} \\ \text{s.t. } & \mathbf{I}\mathbf{\Delta U} \leq \mathbf{\Delta U}_{\max} \\ & -\mathbf{I}\mathbf{\Delta U} \leq -\mathbf{\Delta U}_{\min} \\ & \mathbf{T}\mathbf{\Delta U} \leq \mathbf{1u}_{\max} - \mathbf{1u}[k-1] \\ & -\mathbf{T}\mathbf{\Delta U} \leq -\mathbf{1u}_{\min} + \mathbf{1u}[k-1] \end{aligned} \tag{93}$$

which is a quadratic program. As long as $\mathbf{Q}$ and $\mathbf{R}$ contain only positive values, $\mathbf{H} = \mathbf{G}^T\mathbf{Q}\mathbf{G} + \mathbf{R}$ is structurally positive definite. This guarantees that the constraints and objective function are convex and, with any other linear constraints, compose a convex QP problem, which can be easily solved.

3.2 CASE STUDIES

This section describes different case studies where the ESN-PNMPC was applied, namely the benchmark four-tank system, an oil platform consisting of two wells and one riser, and an electrical submersible pump-lifted oil well. The ESN-PNMPC was implemented using the numpy toolkit from *Python*, with *CVXOPT* as the QP problem solver. Both models were described and implemented with the help of the Modelica language.

3.2.1 Four Tanks

This section describes the Four-Tanks system and the experiments that were performed considering it as the plant to be controlled by the ESN-NMPC.

3.2.1.1 Description

The four-tank system (JOHANSSON, 2000) is widely used as a benchmark of multivariate and nonlinear control systems with coupled variables. The system consists of two pumps $j \in \{1,2\}$, two directional valves, and four tanks $i \in \{1,2,3,4\}$ with levels h_i. The four-tank system (JOHANSSON, 2000) is depicted in Figure 6. Each pump has its rotation and flow controlled by voltage u_j, with j being the index of the pump. The flow of pump 1 enters both tank 1 and tank 4, while the flow of pump 2 enters tanks 2 and 3, both distributed through directional valves. As tanks 3 and 4 are positioned above tanks 1 and 2, respectively, and each tank has a hole in its bottom, tank 2 (tank 1) is also influenced indirectly by pump 1 (pump 2). This connection between the pumps and the tanks is the source of the coupling in the system.

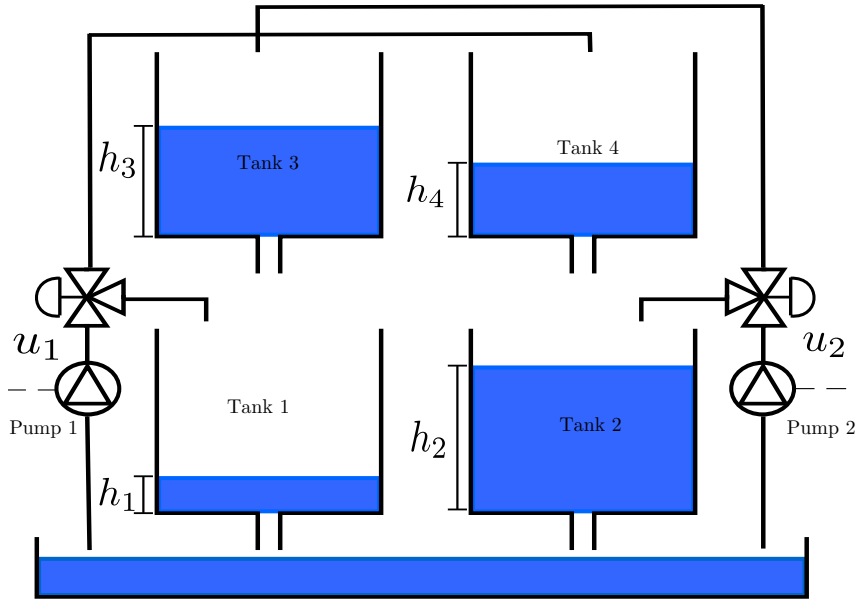


Figure 6 – Representation of a four-tank system. Adapted from (BRANDÃO et al., 2018). Description is given in the text.

The four-tank system is described by the following equations:

$$\dot{h}_1 = \frac{\gamma_1 k_1 u_1 + \omega_3 - \omega_1}{A_1} \tag{94}$$

$$\dot{h}_2 = \frac{\gamma_2 k_2 u_2 + \omega_4 - \omega_2}{A_2} \tag{95}$$

$$\dot{h}_3 = \frac{(1-\gamma_2) k_2 u_2 - \omega_3 + \omega_{dist,3}}{A_3} \tag{96}$$

$$\dot{h}_4 = \frac{(1-\gamma_1) k_1 u_1 - \omega_4}{A_4} \tag{97}$$

where h_i is the level and A_i is the area of the transverse section of tank i; k_j is how much voltage is converted to volumetric flow rate in pump j; γ_j is the opening of the directional valve accompanying pump j related to the lower tanks; and ω_i is the outflow of tank i, calculated as:

$$\omega_i = a_i \sqrt{2gh_i} \tag{98}$$

where a_i is the bottom orifice area for tank i. The values of each parameter, as well as the initial conditions to each state in the problem formulation, originate from (JOHANSSON, 2000), in a configuration that induces a non-minimum phase zero to the initial operating point. The disturbance $\omega_{dist,3}$ is a flow disturbance applied in tank 3 at a given time, which is used to test the disturbance rejection aspects of the controller.

The cost function for the predictive control problem in the four-tank system is as follows:

$$J = \sum_{j=1}^{N_y} (q_1 e_1^2[k+j|k] + q_2 e_2^2[k+j|k]) + \sum_{j=1}^{N_u - 1} (r_1 \Delta u_1^2[k+i|k] + r_2 \Delta u_2^2[k+i|k]) \tag{99}$$

where q_1 (q_2) corresponds to the weight of the reference error e_1 (e_2); and r_1 (r_2) is the control variation penalty for Δu_1 (Δu_2). The errors are defined as follows:

$$e_1[k+j|k] = h_{1,ref}[k+j|k] - h_1[k+j|k] \tag{100}$$

$$e_2[k+j|k] = h_{2,ref}[k+j|k] - h_2[k+j|k] \tag{101}$$

with $h_{i,ref}$ being the reference for h_i, $i \in \{1,2\}$. The cost function in (99) is the widely used quadratic penalization of the set point error and the control increment (CAMACHO; BORDONS, 1999), and, if the system is linear in the input, constitutes a QP.

The optimization problem also imposes the following constraints in the system:

$$0V \leq u_1, u_2 \leq 5V \tag{102}$$

$$|\Delta u_1|, |\Delta u_2| \leq \Delta u_{\max} \tag{103}$$

$$h_{i,\min} \leq h_i \leq h_{i,\max},\ i \in \{3,4\} \tag{104}$$

where $h_{i,\min}$ ($h_{i,\max}$) is the minimum (maximum) value that the upper tank levels ($i \in \{3,4\}$) can reach. Realistically, it might correspond to the tank height ($h_{i,\max}$) or the safe maximum

of the liquid level to prevent pump damage. $\Delta u_{\max}$ is the maximum control increment possible for each control action. For the PNMPC, it might be convenient to set this parameter at low values, such as $\Delta u_{\max} = 1.0$, as the function utilized is an approximation of the full nonlinear ESN.

Since certain steady-state constraints in the tanks are not controlled, some combinations of setpoints may not be reachable. For this reason, the experiments are performed in two different optimization problems:

- One where $h_{3,\min} = h_{4,\min} = 0.5$ cm and $h_{3,\max} = h_{4,\max} = 12$ cm. In this case, the constraints are innocuous to the extent that the constraints are not binding.
- The other where $h_{3,\min} = h_{4,\min} = 0.5$ cm and $h_{3,\max} = h_{4,\max} = 9$ cm. Now, the possible output state is more tight, which can lead to infeasible set-point combinations.

The first case is employed for ESN identification and hyperparameter analysis. The second case is used in a disturbance rejection and tracking test, using the ESN with the configuration that yields the lowest control error obtained in the first case.

3.2.1.2 System Identification

The identification of the model is based on the training of the readout output layer of the ESN according to ((55)), which finds the weights $\mathbf{W_r^o}$ connecting the reservoir layer to the output layer by solving the linear system in ((55)). $\mathbf{A}$ and $\mathbf{D}$ are found as follows. After collecting input-output pairs $(\mathbf{u}[k], \mathbf{d}[k])$, for $k = 1,...,N$, where N is the number of samples from the plant, ((53)) is applied using $\mathbf{u}[k]$ for all available timesteps, with the resulting reservoir states being collected into a matrix $\mathbf{A}$. Note that a warm-up drop of the first 100 initial states in $\mathbf{a}[k]$ is applied, eliminating the starting transient of the reservoir state. The corresponding desired output $\mathbf{d}[k]$ is also collected in a matrix $\mathbf{D}$.

For every randomly initialized ESN, defined by random matrices $\mathbf{W_i^r}$, $\mathbf{W_r^r}$, $\mathbf{W_b^r}$, the regularization parameter λ is found using cross-validation. Choosing suitable scalings for these random matrices is necessary in order to optimize ESN prediction performance as well as control performance. In this context, should the tuning of these scalings to achieve the best control performance be based on ESN prediction performance? These experiments show that there is a better, but more expensive alternative metric than the ESN prediction performance.

For system identification and validation purposes with respect to the four-tank system, the input signal $\mathbf{u}[k]$, representing the pumps, consists of an Amplitude Modulated Pseudo-Random Bit Sequence (APRBS) (NELLES, 2020) with range $u_i \in [0.1,5.0]$, $i \in \{1,2\}$ (before normalization) generated for $50{,}000$ timesteps. Under a sampling time of 10 seconds, and applying $\mathbf{u}[k]$ as input to the plant, a desired output $\mathbf{d}[k]$ is built by measuring the plant's response, that is, the level of the four tanks. Both $\mathbf{u}[k]$ and $\mathbf{d}[k]$ are normalized such that they lie on the interval $[0,1]$. The first $40{,}000$ timesteps were used for training the ESN and for five-fold cross-validation (5-fold CV) (BISHOP, 2006), while the latter $10{,}000$ timesteps

were used to measure the ESN's test prediction performance using the Integral Absolute Error (IAE) metric:

$$IAE = \sum_{k=1}^{N_{test}} |\mathbf{d}[k] - \mathbf{y_{esn}}[k]| \tag{105}$$

Control performance is computed in terms of IAE and using a randomly generated reference signal $h_{i,ref} \in [5,15], i \in \{1,2\}$, for $N = 200$ timesteps to form $\mathbf{h_{ref}}$, shown in Fig. 7. The IAE equation for the control experiment is:

$$IAE = \sum_{k=1}^{N} |\mathbf{y_m}[k] - \mathbf{y_{ref}}[k]| \tag{106}$$

The 5-fold CV can be performed in the ESN even though the whole model is dynamic because CV is being performed with respect to the state-to-output least squares. Although the relation between input and output is dynamic (data would be a sequence), this is not the case for the output computation from the ESN states, which is a static relation, therefore there is no time dependence.

Although IAE weights all instances in time equally, it serves the purpose of providing a sensitive comparison metric between different controllers, evluating both steady state and transients equally, hence it is a decent metric to evaluate ESN-PNMPC in the context of this work.

Our first experiment is explained as follows: each possible parameter setting for leak rate and spectral radius (from a predefined list of values) is evaluated ten times. The input scaling is fixed empirically at 0.1 and the number of units in the reservoir is set to 100 neurons. Each trial considers a different randomly initialized ESN, and, thus, will perform a 5-fold CV for finding the best regularization parameter value. The mean IAE of these ten runs are computed for the test set and shown in Fig. 8, while the mean IAE for the control task, while applying the ESN-PNMPC on the plant, is shown in Fig. 9. Here, the correction filter is not used, and thus, the control is in open-loop. This is because the experiment aims to evaluate the capacity of the ESN in helping the control performance, without the *help* of the error correction filter. The configuration of the spectral radius and leak rate that yields the minimum IAE for the ESN's prediction (Fig. 8) is given by the values of 0.4 and 0.2, respectively. Besides, in the majority of the plot, there is a smooth transition from white to black when walking through the parameter space. On the other hand, in Fig. 9, that configuration is 1.1 and 0.3 for spectral radius and leak rate, respectively. This preliminary experiment shows that the control performance does not match exactly to the ESN's prediction performance, in terms of the IAE surface dependent on the leak rate and spectral radius. Thus, it seems that the best coupling of these hyperparameters should be found by looking at the control (tracking) error instead of the ESN model's error. The drawback is that evaluating MPC control is many orders of magnitude more computationally expensive than just model evaluation.

The same type of grid search analysis was performed for reservoirs containing 400 units. The IAE surface of the ESN's prediction is somewhat different from what is shown in Figures 8

and 9, being more flat, that is, the black area where the error is lowest is wider. The following results section considers the 400-unit reservoir with parameter configuration that yields the lowest MPC tracking error, *i.e.*, spectral radius of 0.99 and leak rate of 0.3, properly regularized with a 5-CV for finding the regularization parameter. The choice of a larger reservoir is due to the observation of less oscillations in control when compared to the 100-unit reservoir.

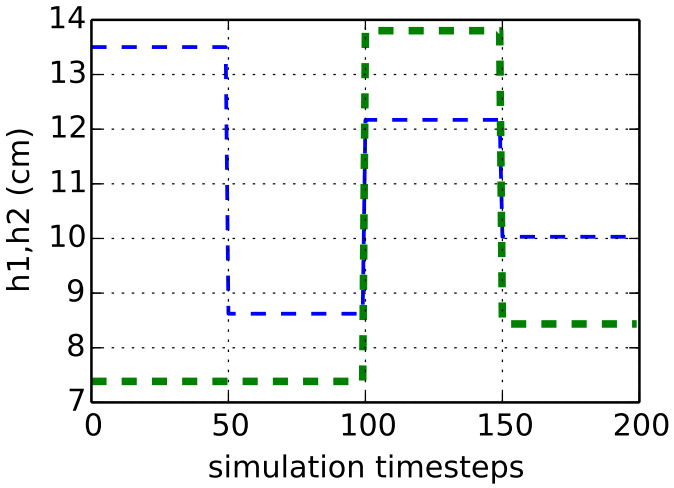


Figure 7 – Randomly generated reference signal $\mathbf{h_{ref}}$ for the level of tanks h_1 and h_2 to be used for control performance evaluation) with a duration of 200 timesteps.

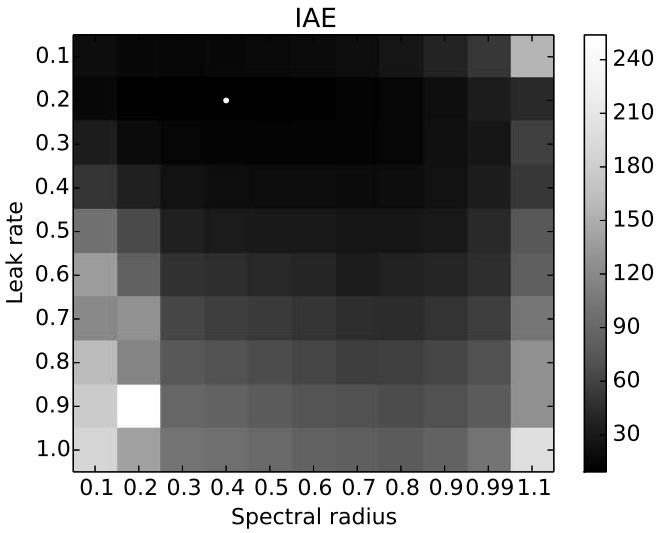


Figure 8 – IAE as a function of spectral radius and leak rate, evaluated for an ESN model with 100 units in the reservoir. A white point localized in the middle of the darkest cell gives the minimum IAE. Test IAE between ESN's prediction and reference signal for 10,000 timesteps.

Another experiment evaluates the test error (RMSE) as a function of the reservoir size and the size of the training set (Fig. 10). This is practically relevant for control of real-world plants for which data collection time should be as short as possible. As expected from machine

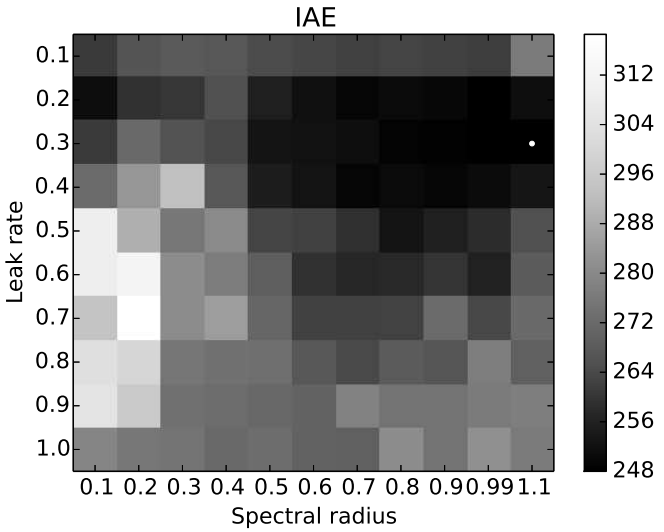


Figure 9 – IAE as a function of spectral radius and leak rate, evaluated for an ESN model with 100 units in the reservoir. A white point localized in the middle of the darkest cell gives the minimum IAE. IAE between the plant's measured response during a MPC task and a randomly generated test reference signal for 200 timesteps from Fig 7.

learning literature, the error on the test set decreases as the training set increases. The same is valid for bigger reservoirs whose training is always regularized. In the plot, each point represents the mean of the test RMSE for ten runs with randomly generated reservoirs, where each run includes a 5-CV for the search of the regularization parameter. Note that this is the error on the model prediction and that the MPC control error will not necessarily follow the same pattern.

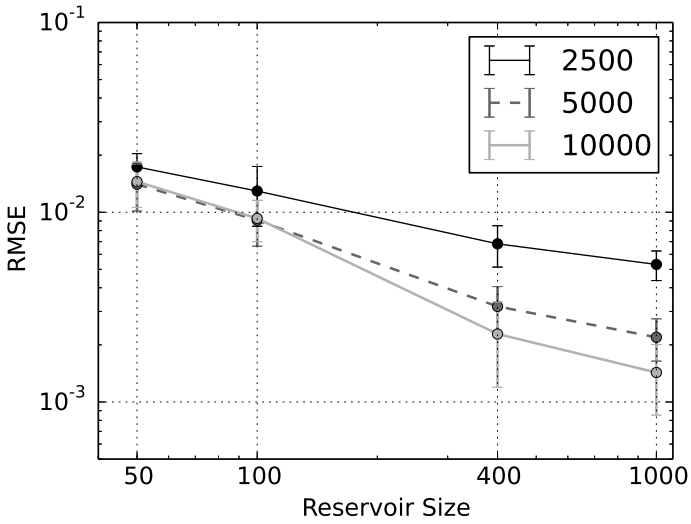


Figure 10 – Test RMSE as a function of reservoir size (50, 100, 400, 1000) and training set (2500, 5000 and 10,000 instances).

3.2.1.3 Control Experiments

An open-loop step test in the four-tank system showed that the slowest step response of the system was of 35 time steps. This test led to a prediction horizon of 50 time steps (500 seconds) for the PMNPC, so that the steady state is captured in one future prediction and taken into account during optimization. The control horizon used was of 5 time steps, a sufficiently small size to reduce the computational complexity of the execution, since the control horizon directly dictates the number of decision variables optimized in the QP. The filter parameters K and ω were tuned so that the pole of the a priori-posteriori error dynamics is $a = 0.5$ for each variable, which means that the a posteriori error shall converge to 0 at 70 seconds, once a bias appears in the a priori error. The identity matrix was used for both error weights $\mathbf{Q}$ and control variation weights $\mathbf{R}$, so the errors and the variations are equally penalized (the system is normalized in the PNMPC point of view).

A tracking experiment is performed in this section where the system is more constrained ($h_3 = h_4 = 9$ cm), using the 400-unit regularized ESN with best parameters obtained in Section 3.2.1.2. Note that the results on this section were obtained with completely new unseen data, as the reference signal for control is randomly generated once again.

The reference setpoint signal was set as an APRBS signal with range $h_{i,ref} \in [5,15]$, $i \in \{1,2\}$ over $3{,}000$ time steps, and the step-type flow disturbance in Tank 3 ($\omega_{dist,3}$) was injected in the system halfway the simulation to showcase disturbance rejection and robustness to an abrupt change in the model. When the disturbance is applied, the ESN is expected to not perform well, as the model has changed during the simulation run.

Fig. 11 is divided in three subplots: the topmost shows the level of the lower tanks (h_1 and h_2) and each associated reference signal, the second one shows each pump voltage, and the third one shows the upper tank levels (h_3 and h_4) alongside the upper and lower bounds imposed on them. The black dashed vertical line flags when the Tank 3 flow disturbance $\omega_{dist,3} = 1.8\ kg/s$ is injected into the system.

Although the presence of the disturbance has considerably diminished the prediction accuracy of the ESN due to a parametric change in the plant model, the system still managed to achieve reference tracking along all the 1500 time steps. In the third plot of Fig. 11, The constraints are taken into consideration and are overall not violated despite: the disturbance compromising the prediction capacity; the ESN being a proxy of the actual model; and the constraints calculated by the predictive controller not matching perfectly the actual system constraints. The reason of this success is shown in Fig. 12, where the a prior relative error $e_{pre,\%}$ is compared to the a posteriori error $e_{post,\%}$. In accordance with the PNMPC framework, the correction is able to filter the integration error and compensate for the bias in the prediction induced by the disturbance. Even with a severe modeling error provoked by the unmeasured disturbance, tracking is still possible. As the level constraints are modeled as "hard" (instead of appearing as a penalty in the objective function, imposed as a mathematical bound on the function domain), the controller takes priority into obeying the constraints, therefore tracking

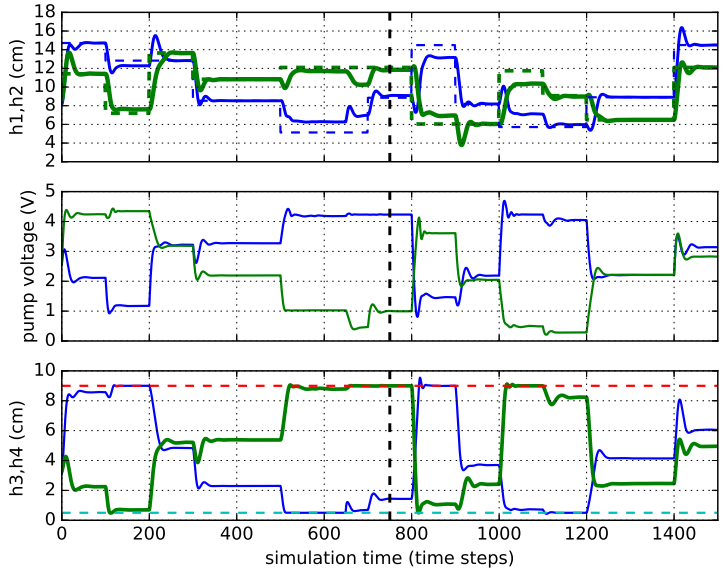


Figure 11 – Experimental results for the case where $h_{3,\max} = h_{4,\max} = 9$ in a 3000 time steps run where a disturbance is applied at $k = 750$. From upwards to downwards: the first plot consists of the tracking response of tank levels h_1 (blue) and h_2 (green) as solid lines, alongside their respective reference signals (dashed lines, matching colors and thickness); the second plot contains the voltage signal for pump 1 (blue) and 2 (green); and the third plot showcases the upper levels (h_3 and h_4) over time, alongside their upper (9 cm) and lower (0.5 cm) bounds. The dashed vertical line shows when the disturbance $\omega_{dist,3} = 1.8\ cm^3/s$ is input to the system, at time step number 750.

is sacrificed, which is the reason why at some points the reference is not tracked, as shown in Fig. 11.

3.2.1.4 Comparison with LSTM-PNMPC

This experiment compares the ESN-PNMPC to a LSTM-PNMPC controller, where an LSTM (HOCHREITER; SCHMIDHUBER, 1997) (Long Short-Term Memory) is used instead of an ESN. The LSTM hyperparameters are the number of cells, and the hyperparameters related to the algorithm that performs nonlinear optimization to train the network weights, such as the learning rate, the batch size, the regularization coefficients, number of epochs, the dropout rate among others. The ESN, in comparison, has more structural hyperparameters, but because of the nonlinear optimization of the LSTM, the number of hyperparameters both networks have to consider when tuning is roughly the same. In the case of the LSTM, the hyperparameters were selected through grid search. The LSTM network was trained for 40 epochs with the ADAM (KINGMA; BA, 2014) optimization algorithm using exactly the same dataset as the ESN ($40{,}000$ samples, see Section 3.2.1.2), and implemented with the *pytorch* library. Its hidden layer has 20 cells (other configurations, *e.g.*, 5 or 80 neurons, showed higher validation error). The built-in automatic differentiation of *pytorch* (PASZKE et al., 2017) is used to calculate

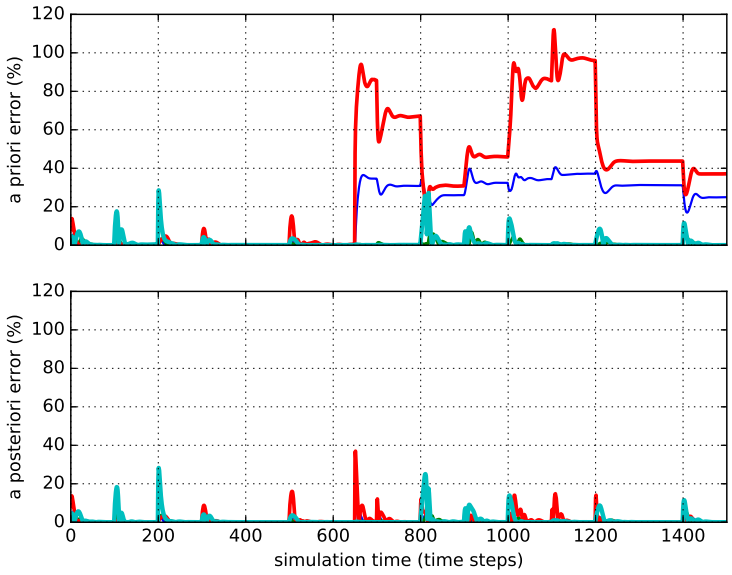


Figure 12 – Effect of the error correction filter for the case where $h_{3,\max} = h_{4,\max} = 9$ in a 3000 time steps run where a disturbance is applied at $k = 750$. The topmost subplot gives the a priori ESN prediction errors of the four tank levels h_1,h_2,h_3,h_4 (sorted by line thickness, with h_1 as the thinnest), and the bottom-most plot shows the relative error of the level predictions after correction is applied.

the Jacobian matrix $\mathbf{G}$. Note that the training time for an LSTM is orders of magnitude higher that of an ESN, since it is based on (iterative) gradient descent. Fig. 13 shows the result of the comparison, where the "unconstrained" case is considered ($h_{3,\max} = h_{4,\max} = 12$ cm) and the same reference signal is provided to both controllers for 300 timesteps. Both controllers achieved smooth responses. Although both cases solve exactly the same optimization problem, the ESN-PNMPC had a faster response than the LSTM-PNMPC. The ESN also achieved superior performance over the LSTM in terms of prediction error for the controller run.

Table 4 summarizes the results, showing the IAE for h_1 and h_2, the reference signal, and the time spent computing $\mathbf{G}$, the QP, and the total time to compute a control action. Not only the LSTM-PNMPC controller is performing worse in terms of IAE (around 92% worse) than ESN-PNMPC, but also in terms of computation time—about 500 times on average slower than the proposed ESN-PNPMC scheme. The latter only took 0.662 seconds to calculate $\mathbf{G}$ in the worst case (timestep requiring more computation time) during the 300 timesteps experiment, while the autograd (PASZKE et al., 2017) routine for computing the Jacobian of the LSTM took 170 seconds in the best case, proving unfit to control the four-tank system, which has a sampling time of 10 seconds. This is is surprising since the LSTM has only 20 cells (40 states), and the ESN has 400 neurons, which naturally makes it have more weights connecting each neuron. The time for computing $\mathbf{G}$ dominates the total time for calculating the control action. Thus, the ESN-PNMPC scheme, which simplifies the Jacobian calculation due to the analytical and recursive formulation, is very well suited to real time applications in general, unlike the

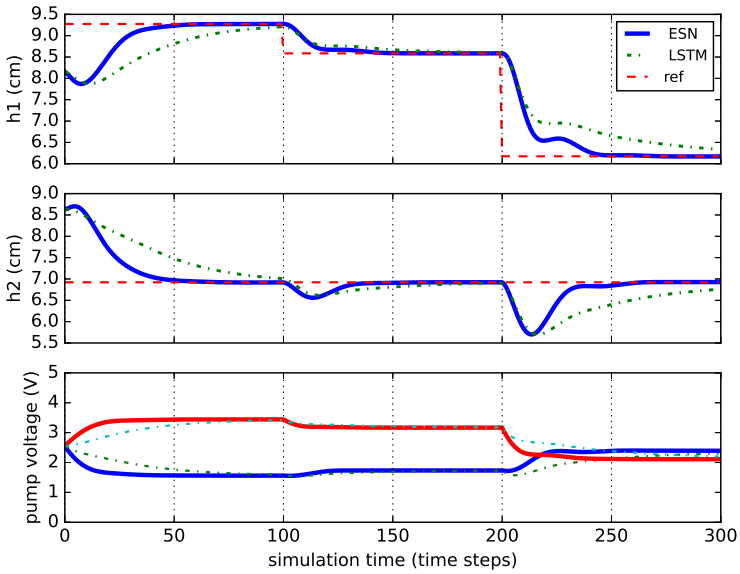


Figure 13 – Comparison between the ESN-PNMPC and LSTM-PNMPC for the control of the four-tank system, for 300 simulation time steps. The first and second plots correspond to the levels of tank 1 and 2, respectively, while the third plot shows the voltage for each pump.

Table 4 – Results for PNMPC with ESN or LSTM in the four-tank system. Computation times are per time step.

	Controller	
	ESN-PNMPC	LSTM-PNMPC
IAE (h_1)	**73.84**	139.4
IAE (h_2)	**69.82**	137.0
mean $\mathbf{G}$ calc. time (s)	**0.34**	174.8
best $\mathbf{G}$ calc. time (s)	**0.247**	170.9
worst $\mathbf{G}$ calc. time (s)	**0.662**	197.8
mean QP time (s)	5.36×10^{-3}	$\mathbf{1.98} \times 10^{-3}$
best QP time (s)	2.9×10^{-3}	$\mathbf{1.77} \times 10^{-3}$
worst QP time (s)	$\mathbf{3.85} \times 10^{-2}$	4.025×10^{-2}
mean total time (s)	**0.35**	174.8
best total time (s)	**0.25**	170.9
worst total time (s)	**0.671**	197.80

currently implemented LSTM. Using our available hardware (Intel Core i5, 3.10 GHz, 4 cores, 8 GC RAM, ran on Python), the ESN-PNMPC with its current configuration (reservoir size; number of inputs and outputs; prediction and control horizon) is fit for systems of at least 1 second sampling, as the worst-case computation time of the controller is 0.671 s. Time as a metric, serves its purpose when comparing runtimes of different controllers and prediction models in the same machine, however it does not serve as a comparison metric for different machines, or to evaluate the performance of ESN-PNMPC when employed in a microcontroler

or other types of embedded systems.

3.2.1.5 Comparison with PI controller and Linear MPC

Here, this experiment evaluates ESN-PNMPC in relation to a Proportional-Integral (PI) controller and a Linear MPC, specifically the Dynamic Matrix Control (DMC) strategy (CAMACHO; BORDONS, 1999), described as follows:

- A decentralized pair of PI controllers. The first PI controls the level of tank 1 (h_1) manipulating the voltage of pump 2 (u_2), while the second one controls h_2 using u_1. This topology was chosen by inspecting each transfer function steady-state gain (JOHANSSON, 2000). Each discrete PI has the form:

$$\frac{U(z)}{E(z)} = K\frac{z - z_0}{z - 1}, \tag{107}$$

  where the pair (K,z_0) was chosen for both PI as $(0.02,0.85)$ for conservativeness, as the system is highly coupled and very prone to non-minimum phase transmission zeros (JOHANSSON, 2000), which is a property of the system as a whole, and not of each individual transfer function.

- Dynamic Matrix Control (DMC) (CAMACHO; BORDONS, 1999). It consists of a linear MPC around the operation point of $u_1 = u_2 = 2.5$ V, chosen for being at the middle of the input range, and thus being a good step response model for the whole operating range of the four-tank system, as the system equilibrium point grows monotonically in function of positive pump voltages. Also, the four-tank system is well-behaved dynamically, therefore the step response of one operating point is a close estimate to other step responses. The DMC was obtained by applying a step of magnitude 0.2 at the aforementioned operation point and recording N_y samples of the step response to place it directly in the DMC $\mathbf{G}$ matrix, according to the procedure in (CAMACHO; BORDONS, 1999). The optimization problem solved in the DMC iteration is exactly the same as in the PNMPC, with the same normalized variables.

Both controller were chosen because they are standard in the literature, performing well without being much complex. Also, DMC has the advantage over GPC in that DMC assumes any kind step response (while assuming the system to be stable), while GPC assumes a so called CARIMA (CAMACHO; BORDONS, 1999) model.

Fig. 14 shows that the DMC was slightly slower than the ESN-PNMPC, which was more aggressive for larger changes in reference, such as at $k = 400$. Table 5 shows the IAE for the control task from Fig. 14, where ESN-PNMPC achieves the lowest tracking error, both in total, and for each controlled variable. Also, the ESN-PNMPC had a lot more ease at adjusting to the same set-point, parting from exactly the same initial condition, as seen at the beginning of the plot. This result is expected theoretically, as the DMC depends on the step-response of one

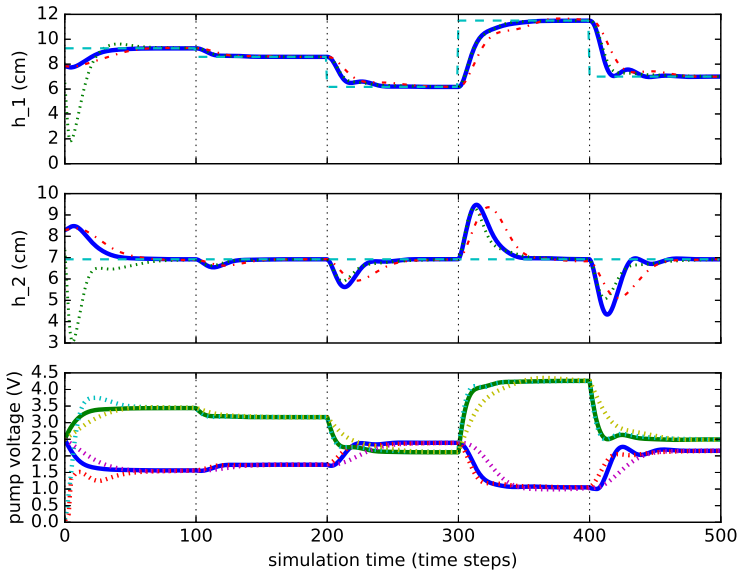


Figure 14 – Comparison between the ESN-PNMPC (solid blue line), a DMC controller (dotted green line), and a PI controller (dotted-and-dashed red line) for the four-tank problem for 500 time steps. The reference signal is given by the dashed cyan line. The first two upper plots correspond to the levels of tank 1 and 2, respectively, while the third plot shows the corresponding pair of control actions for each controller, with PNMPC in solid curve style.

Table 5 – Tracking error in relation to PI controller and Linear MPC (four-tank).

	ESN-PNMPC	PI	DMC
IAE	**375.56**	540.45	473
IAE (h_1)	**203.50**	321.04	290.76
IAE (h_2)	**172.06**	219.41	182.24

operating point (thus having the matrix $\mathbf{G}$ fixed (CAMACHO; BORDONS, 1999)), whereas ESN-PNMPC updates these matrices at each time step. An unconstrained DMC is equivalent to a PID controller (CAMACHO; BORDONS, 1999), just as an unconstrained PNMPC can be seen as an adaptive controller (because $\mathbf{G}$ changes over time). Systems with more expressive nonlinearities better expose the advantage of employing nonlinear controllers (for instance, a Continuous Stirred Tank Reactor (CSTR) (CHEN, H.; KREMLING; ALLGÖWER, 1995)), but this result from the four-tank system is a valid example. Also, the DMC performs well, almost close to the PNMPC, due to the strategical operating point utilized, as mentioned earlier.

As for the PI controller, since it is linear by nature, the controller needs to be designed regarding a (family of) linear system(s). In turn, a nonlinear controller such as the PNMPC can ideally adapt to any operating point, which is a reason for the performance of the ESN-PNMPC being superior.

3.2.2 Two Wells - One Riser

3.2.2.1 Description

The considered platform application consists of two wells and one riser, being the same one that was used in our previous work (JORDANOU, Jean P.; ANTONELO, Eric Aislan; CAMPONOGARA, 2019). Another work that uses ESNs in oil and gas applications is (ANTONELO, Eric A.; CAMPONOGARA; FOSS, 2017), which presents a soft sensor for remote estimation in offshore production platforms.

This experiment a compositional model of two gas-lifted oil wells and one riser, connected by a manifold. Figure 15 depicts the platform, whose components have the following properties:

- **Wells**: Each well is modeled as the reduced order model in (JAHANSHAHI; SKOGESTAD; HANSEN, 2012). The model considers two fluid phases (gas and liquid) and two control volumes: the annulus, containing the gas-lift; and the tubing, containing the production fluid. These define the three states of the well model: the gas in the annulus $\dot{m}_{G,a}$, the gas in the tubing $\dot{m}_{G,t}$ and the liquid in the tubing

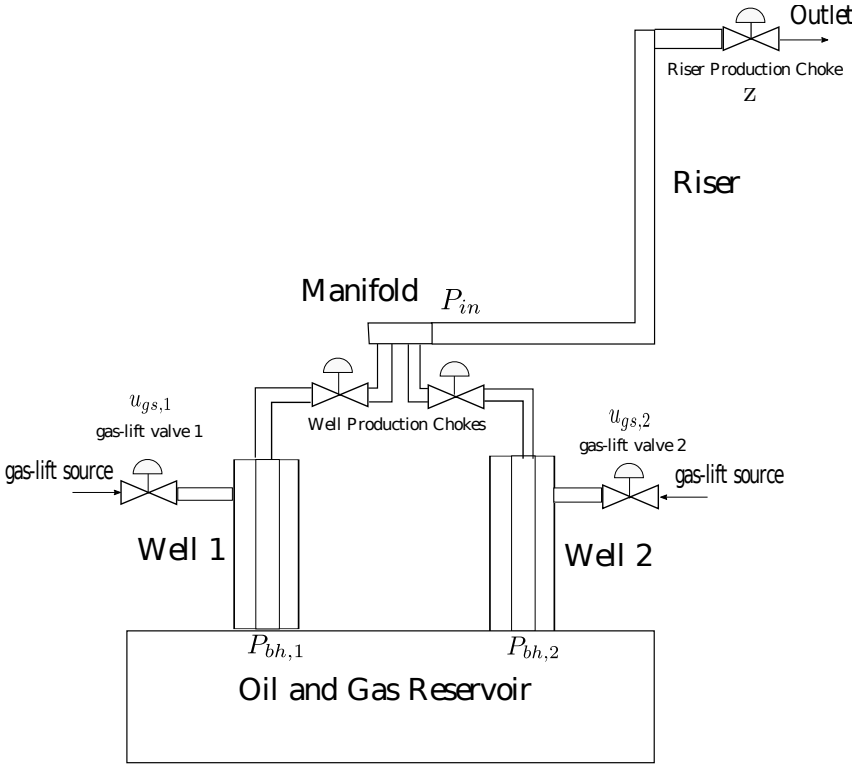


Figure 15 – Representation of an oil platform containing two wells and one riser. From (JORDANOU, Jean P.; ANTONELO, Eric Aislan; CAMPONOGARA, 2019).

$\dot{m}_{L,t}$; each state is calculated by the following mass balance equations:

$$\dot{m}_{G,a} = \omega_{G,in} - \omega_{G,inj} \tag{108}$$

$$\dot{m}_{G,t} = \omega_{G,inj} + \omega_{G,res} - \omega_{G,out} \tag{109}$$

$$\dot{m}_{L,t} = \omega_{L,res} - \omega_{L,out} \tag{110}$$

The gas flow $\omega_{G,in}$ is injected into the annulus, and $\omega_{G,inj}$ is the gas flow injected from the annulus to the main tubing. The flow defined by ω_{res} has its source on the oil reservoir attached to the well, and ω_{out} defines the outlet flows. Each of these flows is calculated through the Bernoulli orifice equation, which is a function of the pressure at certain points in the well and the choke valves involved, which is the gas-lift choke valve u_{gs} for $\omega_{G,in}$ and the wellhead choke valve u_{ch} for the outlet flow ω_{out}. The well model has 42 algebraic variables, 3 state variables, 2 input variables, and 3 boundary conditions, which are the pressure at the gas-lift source P_{gs}, the oil reservoir pressure P_{res}, and the well outlet pressure, which is connected to the rest of the system through the manifold. As the dynamic model of the wells are rather complex to be presented here, refer to (JAHANSHAHI; SKOGESTAD; HANSEN, 2012) for more details on the well model.

- **Riser**: The riser is modeled as a reduced order model developed in (JAHANSHAHI; SKOGESTAD, 2011). In the same vein as the well model, two control volumes are considered: a horizontal and a vertical pipeline. The states of the system are the gas and liquid mass that are present in each control volume. The system is calculated using the modeling logic as the well: each state is modeled as a mass balance of an input and an output flow. The input flow is a boundary condition, which in this case has its source on the two wells below, as shown in Figure 15. The output flow is the production of the whole system, being regulated by the riser production choke valve opening z and the output pressure, which is also a boundary condition and generally corresponds to the pressure at a separator of an FPSO (Floating Production Storage and Offloading) vessel. The riser model possesses 4 state variables, 36 algebraic variables, one input variable, and three boundary conditions.
- **Manifold**: The manifold is modeled as proposed in (JORDANOU, Jean P.; ANTONELO, Eric Aislan; CAMPONOGARA, 2019). In terms of modeling and equations, it corresponds to equating the riser inlet pressure to each well output pressure, and ensuring the riser inlet flow to be the sum of both well flows, while disregarding any load loss due to friction.

Overall, the whole system in Figure 15 has 120 algebraic variables, 10 state variables, 5 input variables, and exactly 5 boundary conditions: the reservoir pressure $P_{res,i}, i \in \mathcal{W} = \{1,2\}$ of each well i, the gas-lift inlet pressure $P_{gs,i}$ of each well i, and the riser output pressure. The exact same parameters as (JORDANOU, Jean P.; ANTONELO, Eric Aislan; CAMPONOGARA,

2019) are used, where the boundary conditions are $P_{gs,1} = P_{gs,2} = 200$ bar and the output pressure is $P_o = 50$ bar.

The problem for this case study is akin to a control problem addressed in (JORDANOU, Jean P.; ANTONELO, Eric Aislan; CAMPONOGARA, 2019). The problem consists in manipulating the choke valves of well 1 and well 2 ($u_{ch,1}, u_{ch,2}$) to track a setpoint signal in each well bottom-hole pressure (P_{bh}). It is important to reiterate that any problem with a quadratic cost function and linear constraints is solvable by the ESN-PNMPC, which also includes econometric formulations (*e.g.*, maximize the net present value). This tracking problem considers the following quadratic cost function in the context of predictive control:

$$J = \sum_{j=1}^{N_y} (q_1 e_1^2[k+j|k] + q_2 e_2^2[k+j|k]) + \sum_{j=1}^{N_u-1} (r_1 \Delta u_{ch1,}^2 + r_2 \Delta u_{ch,2}^2) \tag{111}$$

where the errors are defined as:

$$e_1[k+j|k] = \widetilde{P}_{bh,1}[k+j|k] - P_{bh,1}[k+j|k] \tag{112}$$

$$e_2[k+j|k] = \widetilde{P}_{bh,2}[k+j|k] - P_{bh,2}[k+j|k] \tag{113}$$

and $\widetilde{P}_{bh,i}$ is the setpoint for the corresponding bottom-hole pressure $P_{bh,i}$ of well i.

In this specific case, there are no output constraints. However, constraints related to valve limitation and rate limiting are added to the formulation:

$$0.01 \leq u_{ch,1}, u_{ch,2} \leq 1 \tag{114}$$

$$-\Delta u_{\max} \leq \Delta u_{ch,1}, \Delta u_{ch,2} \leq \Delta u_{\max} \tag{115}$$

By default, just as in the four-tank experiment, $\Delta u_{\max} = 0.2$. The ESN and the PNMPC consider a sampling of 1 min for the platform, which is suitable for such applications.

3.2.2.2 System Identification and Controller Setup

The datasets are generated by exciting the system with the APRBS (NELLES, 2020) for both inputs $u_{ch,1}$,$u_{ch,2}$. The training dataset is composed of $10{,}000$ instances in the form of input-output pairs $\{((u_{ch,1}, u_{ch,2}), (P_{bh,1}, P_{bh,2}))\}$. A validation set of $10{,}000$ instances is employed to empirically find the hyperparameter values for a 400-unit ESN with the lowest RMSE validation error, yielding: a leak rate of 0.7, an input and bias scaling of 0.1, and a spectral radius of 0.999. This parameter setting was not critical for this task, so a refined grid search was not necessary. All the output variables were scaled from the interval $[170{,}220]$ to $[0{,}1]$ before training. The prediction performance of the ESN was evaluated on $10{,}000$ test instances, presenting a RMSE of $(0.031, 0.035)$ for both output variables. Note that the real test of this trained model is done within the control task of ESN-PNMPC.

After training the ESN model, comes the tuning of the PNMPC for this application. The prediction horizon was decided through the execution of an open-loop test which showed

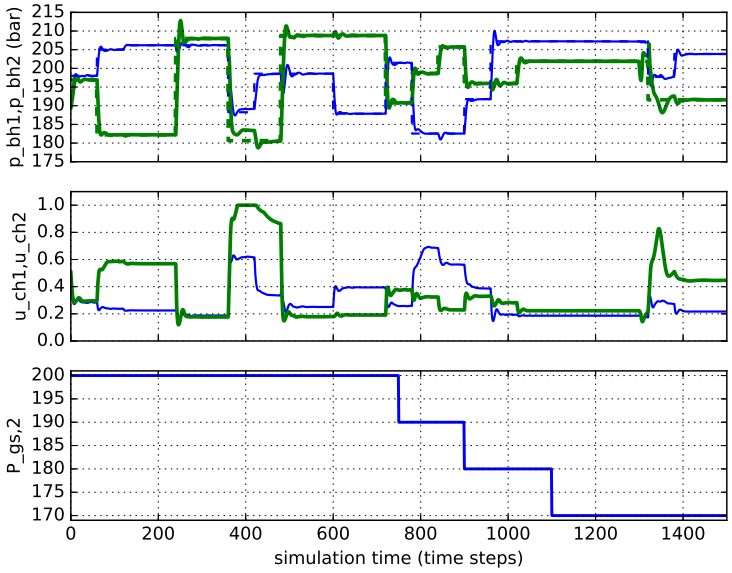


Figure 16 – Results for a 1500 time steps run where the gas-lift source pressure of the second well ($P_{gs,2}$) is depleting 10 bar at times $k = \{750,900,1100\}$ as disturbance. The topmost subplot depicts the tracking experiment, where each well bottom-hole pressure ($P_{bh,1}$ as a blue, solid and thin line, and $P_{bh,2}$ as a green, solid and thick line) is plotted together with their set-points (dashed lines of matching color and thickness) over time; the second subplot contains the control action of each well choke valve ($u_{ch,1}$: blue and thin, $u_{ch,2}$: green and thick); and the third plot represents the disturbance at the gas-lift source pressure $P_{gs,2}$ over time.

that the system reaches steady-state after 30 time steps (minutes). For each prediction window to include a portion of steady-state time, the prediction horizon was set to 40 time steps. The control horizon has length 5, chosen as a sixth of the prediction horizon (CAMACHO; BORDONS, 1999). The filter parameters K and ω tuning induced the pole $a = 0.3$ for each variable, as a smaller value for a means a quicker error response while maintaining a certain degree of robustness. As with the four-tank application, $\mathbf{Q}$ and $\mathbf{R}$ are the identity matrix.

3.2.2.3 Control Experiments

The executed identification of the ESN considers that the gas-lift source pressure of the second well $P_{gs,2}$ remains static at 200 bar for all training samples. Note that this pressure constancy does not happen in real-world oil platforms. Here, the controller has to track an APRBS reference signal, besides rejecting disturbances that occurs in $P_{gs,2}$. This disturbance is simulated as pressure drops of 10 bar that happen at times $k \in \{750,900,1100\}$, which changes the model drastically (and nonlinearly (JAHANSHAHI; SKOGESTAD; HANSEN, 2012)) when compared to the four-tank system, where a disturbance is directly added to the state equation. The simulation run lasts 1500 time steps.

In Fig. 16, the first plot shows the tracking results of the experiment, while the second one shows the a priori and a posteriori errors of each well bottom-hole pressure. The model

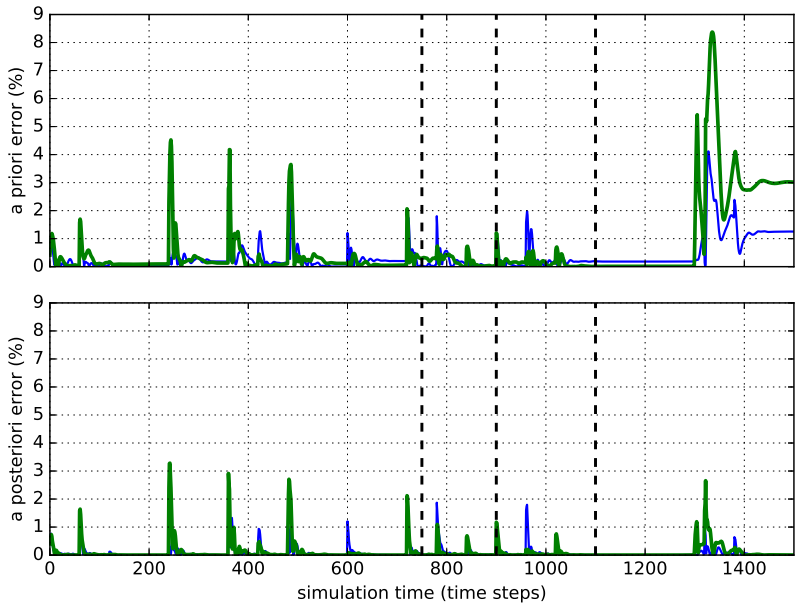


Figure 17 – Correction filter effect for a 1500 time steps run where the gas-lift source pressure of the second well ($P_{gs,2}$) is depleting 10 bar at times $k = \{750,900,1100\}$ as disturbance. The topmost subplot contains the a priori ESN prediction errors of each well bottom-hole pressure ($P_{bh,1}$,$P_{bh,2}$), and the bottom-most plot contains the relative error of the pressure predictions after correction is applied. The vertical dashed lines mark each moment when $P_{gs,2}$ changes value.

changes more drastically as the pressure keeps dropping. This parametric change affects our a priori prediction less than in the four-tank case. Even then, the prediction filter manages to decrease the prediction error to less than 4% through the whole simulation. Note that the third pressure drop at $k = 1100$ results in a large perturbation manifested later (at setpoint change) in the a priori error plot. In this scenario, the filter efficiently corrects the model predictions, reducing the a posteriori error. Additionally, the dynamics of the two wells during tracking is well behaved, despite the influence of the disturbance. For the simulation in Fig. 16, the ESN-PNMPC achieved a total IAE of 1918.83, which results from the IAE of $(757.68, 1161.15)$ for each well bottom-hole pressure.

The ESN-PNMPC was compared to a DMC for the control of the two wells in the platform, designed with the same strategy of obtaining a step response at the operating point $u_{ch,1} = u_{ch,2} = 0.5$, at the middle of the control range, and solving the same optimization problem for both controllers. Both controllers were simulated under the exact same conditions (reference signal and disturbance). The total IAE achieved by the DMC was 1961.4, corresponding to the IAE of $(682.9, 1278.5)$ for each well, which was slightly better for $P_{bh,1}$ and slightly worse for $P_{bh,2}$ compared to our approach. Fig. 18 shows the results of both controllers between time steps 700 and 1100. This region is depicted because it is exactly when the changes in $P_{gs,2}$ happen, providing a disturbance to the controller. The first 50 time steps are a good sample of what generally happens in the simulation, where disturbances are not present. Even though the DMC is faster, it is also more oscillatory and aggressive, therefore more prone

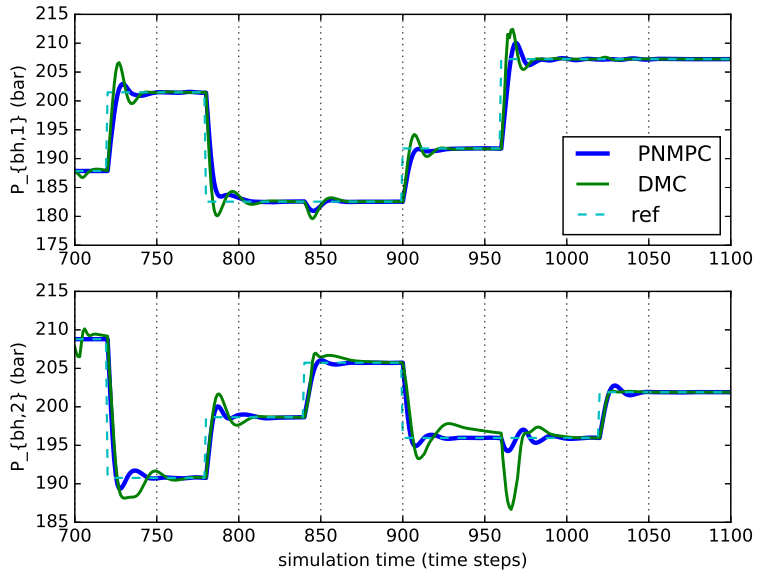


Figure 18 – Comparison between the ESN-PNMPC, and a DMC-type MPC for the oil production platform during pressure disturbances applied to the gas-lift supply, between time steps 700 and 1100 of Fig. 16. The reference signal is given by the dashed cyan line for the bottom-hole pressures $P_{bh,1}$ and $P_{bh,2}$ in the upper and lower plots, respectively. The IAE was $(\mathbf{271.22}, \mathbf{210.64})$ (total: **481.86**) for the ESN-PNMPC, and $(289.68, 429.16)$ (total: 718.84) for the DMC.

to overshooting. This cannot be expressed with IAE alone. The rest of the plot, which is when the disturbances happen, showcase how our approach can reject disturbances more efficiently, as $P_{gs,2}$ affects $P_{bh,2}$ more intensely. The qualitative differences in disturbance rejection between both controllers can be seen after $k = 950$ in Fig. 18. The theoretical explanation for this improvement in ESN-PNMPC is that the DMC uses a single step response of a single operating point as a model, while ESN-PNMPC updates $\mathbf{G}$ at each time step according to the free response trajectory, better adapting to changes. Also, the presence of the first order error correction filter helps in rejecting any bias from disturbance, while this implementation of DMC (CAMACHO; BORDONS, 1999) uses only the current modeling error to correct the free response.

3.2.3 Electrical Submersible Pump-lifted Oil Well

The results shown in this section were also published in (JORDANOU, Jean P. et al., 2022). The Electrical Submersible Pump (ESP) oil well besides being an interesting practical problem for the ESN-PNMPC, also served as a case study to compare the performance of ESN-PNMPC with a filtered NMPC using an ESN as a model (JORDANOU, Jean P. et al., 2022).

3.2.3.1 Description

The model of the ESP-lifted Oil Well employed for this work (depicted in Figure 19) was developed by (PAVLOV, A. et al., 2014), with some improvements by (BINDER, Benjamin J. T.; PAVLOV, Alexey; JOHANSEN, 2015). It employs the average flow of a mixture as state, instead of taking into account each phase flow, which makes the model simpler to compute and use in control applications.

The ESP-lifted well is characterized by the following dynamic equations:

$$\dot{p}_{bh} = \frac{V_1}{\beta_1}(q_r - q) \tag{116a}$$

$$\dot{p}_{wh} = \frac{V_2}{\beta_2}(q - q_c) \tag{116b}$$

$$\dot{q} = \frac{1}{M}(p_{bh} - p_{wh} - \rho g h_w - \Delta p_f + \Delta P_p) \tag{116c}$$

where p_{bh} is the well bottom-hole pressure, p_{wh} is the wellhead pressure, q is the average liquid flow rate in the well, q_r is the input flow rate originated from the reservoir, q_c is the flow rate at the production choke, ρ is the fluid density, g is the gravity acceleration, and h_w is the well height. The volumes V_1 and V_2 are the pipe volumes below and above the ESP, respectively, while β_1 and β_2 are the bulk modulus, which are fluid dynamics parameter that model the fluid resistance to compression. The parameter M is the fluid inertia. The quantity Δp_f is the pressure loss due to friction, and ΔP_p is the pressure increase due to pump dynamics.

The input flow rate q_r and the output flow rate q_c are calculated as follows:

$$q_r = PI(p_r - p_{bh}) \tag{117a}$$

$$q_c = C_c z \sqrt{p_{wh} - p_m} \tag{117b}$$

where PI is the production index calculated for the reservoir, p_r is the pressure in the reservoir, $z \in [0,1]$ is the production choke opening, C_c is the choke valve constant, and p_m is the manifold pressure.

The load loss due to friction is calculated as:

$$\Delta p_f = F_1 + F_2 \tag{118a}$$

$$F_i = 0.158 \frac{\rho L_i q^2}{D_i A_i^2} \left(\frac{\mu}{\rho D_i q} \right)^{\frac{1}{4}} \tag{118b}$$

where L corresponds to length, D to diameter, and A to the pipe cross-section area. The index 1 corresponds to the section between the reservoir and the ESP, whereas index 2 corresponds to the section between the ESP and choke. The parameter μ is the viscosity of the fluid.

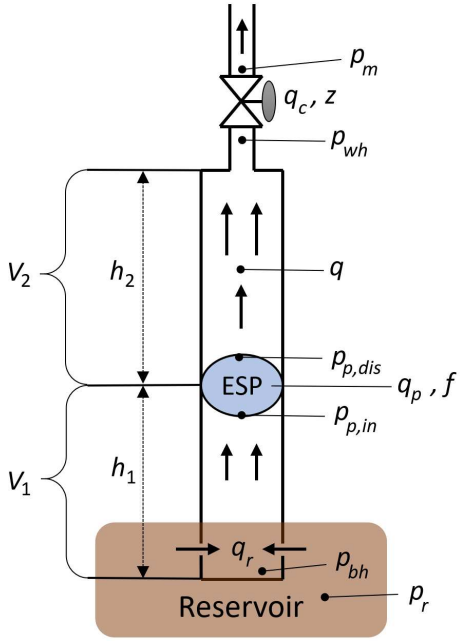


Figure 19 – Schematic representation of an ESP lifted well, adapted from (BINDER, B. J. T. et al., 2014), and obtained directly from (JORDANOU, Jean P. et al., 2022).

The ESP pressure increase is calculated as:

$$\Delta P_p = \rho g H \tag{119a}$$

$$H = C_H(\mu)\left(c_0 + c_1\left(\frac{q}{C_Q(\mu)}\frac{f_0}{f}\right) - c_2\left(\frac{q}{C_Q(\mu)}\frac{f_0}{f}\right)^2\left(\frac{f}{f_0}\right)^2\right) \tag{119b}$$

where c_0,c_1 and c_2 are pump constants, C_H and C_Q are constants dependent on the viscosity, f_0 is the nominal frequency of the pump, and f is the pump rotation frequency.

The control inputs for this system are the production choke opening z and the ESP frequency f. Table 6 presents the model parameters of the ESP-lifted oil well.

The model described by the system (116) is a lumped model for the ESP used as a reference in this work. Models related to oil and gas production that accurately predicts the dynamics of a system are hard to compute and may be lacking in accuracy (JAHN; COOK; GRAHAM, 2008), however models such as (116) can capture the fundamental behavior of this type of process.

The first step is to identify an ESN model that captures the behavior of (116). For the ESP-lifted well case, the ESN must identify the manipulated variables of the well as inputs to the model, to which end one must gather data from the pump rotation frequency f and the valve choke opening z. The ESN outputs are the system states: the bottom-hole pressure p_{bh}, the wellhead pressure p_{wh}, and the average flow rate of the fluid q.

The second step is to employ the trained ESN as the model for the MPC problem with a quadratic tracking objective function. Due to physical limitations, the choke valve opening z is constrained between 0% and 100% and the ESP frequency must range from 35 Hz to 65 Hz (PAVLOV, A. et al., 2014). The output constraints consist of bounds on the bottom-hole

Table 6 – Well dimensions and other known constants

Symbol	Name	Value	Unit
g	Gravitational acceleration constant	9.81	m/s^2
C_c	Choke valve constant	$2 \cdot 10^{-5}$	*
A_1	Cross-section area of pipe below ESP	0.008107	m^2
A_2	Cross-section area of pipe above ESP	0.008107	m^2
D_1	Pipe diameter below ESP	0.1016	m
D_2	Pipe diameter above ESP	0.1016	m
h_1	Height from reservoir to ESP	200	m
h_w	Total vertical distance in well	1000	m
L_1	Length from reservoir to ESP	500	m
L_2	Length from ESP to choke	1200	m
V_1	Pipe volume below ESP	4.054	m^3
V_2	Pipe volume above ESP	9.729	m^3
f_0	ESP characteristics reference freq.	60	Hz
I_{np}	ESP motor nameplate current	65	A
P_{np}	ESP motor nameplate power	$1.625 \cdot 10^5$	W
β_1	Bulk modulus below ESP	$1.5 \cdot 10^9$	Pa
β_2	Bulk modulus below ESP	$1.5 \cdot 10^9$	Pa
M	Fluid inertia parameter	$1.992 \cdot 10^8$	kg/m^4
ρ	Density of produced fluid	950	kg/m^3
P_r	Reservoir pressure	$1.26 \cdot 10^7$	Pa
PI	Well productivity index	$2.32 \cdot 10^{-9}$	m^3/s/Pa
μ	Viscosity of produced fluid	0.025	$Pa \cdot s$
P_m	Manifold pressure	20	Pa

pressure, well-head pressure, and liquid flow, respectively $0 \leq p_{bh}$, 1 bar $\leq p_{wh} \leq 60$ bar, and $30\ m^3/h \leq q \leq 50\ m^3/h$. The NMPC optimization problem to be solved at time k is:

$$\min_{\mathbf{\Delta U}} \quad J(\mathbf{y}[k],\mathbf{u}[k],\mathbf{\Delta U}) \tag{120a}$$

$$\text{s.t.} : \mathbf{1}_{N_u} \otimes (\mathbf{u}_{\min} - \mathbf{u}[k]) \leq (\mathbf{T}_{N_u} \otimes \mathbf{I}_m)\mathbf{\Delta U} \leq \mathbf{1}_{N_u} \otimes (\mathbf{u}_{\max} - \mathbf{u}[k]) \tag{120b}$$

$$\mathbf{1}_{N_y} \otimes \mathbf{y}_{\min} \leq \mathbf{Y} \leq \mathbf{1}_{N_y} \otimes \mathbf{y}_{\max} \tag{120c}$$

where $\otimes$ is the Kronecker product, which is used in this formulation as a broadcast operator, namely $(\mathbf{1}_n \otimes \mathbf{a})$ means that the resulting vector is $\mathbf{a}$ replicated n times, which corresponds to the number of rows in $\mathbf{1}_n$. The same holds for the Kronecker product $(\mathbf{T}_{N_u} \otimes \mathbf{I}_m)$. As $\mathbf{T}_{N_u}$ is a lower triangular matrix of size N_u filled with ones, the resulting matrix from the Kronecker product is a block-triangular matrix with non-zero block-elements being equal to $\mathbf{I}_m$, with $m = 3$ being the number of manipulated variables. In the formulation above, $y[k]$ is the system output at the current time k and $u[k]$ is the initial control signal. The actual control signal to be applied over the control horizon is a function of $u[k]$ and the vector $\mathbf{\Delta U}$ of control increments. Notice that $\mathbf{\Delta U}$ is the vector of decision variables.

The remainder of the functions and variables in problem (120) are:

$$\boldsymbol{\Delta}\mathbf{U} = \begin{pmatrix} \boldsymbol{\Delta}\mathbf{u}[k] \\ \boldsymbol{\Delta}\mathbf{u}[k+1] \\ \boldsymbol{\Delta}\mathbf{u}[k+2] \\ \vdots \\ \boldsymbol{\Delta}\mathbf{u}[k+N_u-1] \end{pmatrix}, \ \mathbf{Y} = \begin{pmatrix} \mathbf{y}[k+1] = \mathbf{g}(\mathbf{x}[k+1]) \\ \mathbf{y}[k+2] = \mathbf{g}(\mathbf{x}[k+2]) \\ \mathbf{y}[k+3] = \mathbf{g}(\mathbf{x}[k+3]) \\ \vdots \\ \mathbf{y}[k+N_y] = \mathbf{g}(\mathbf{x}[k+N_y]) \end{pmatrix}, \tag{121a}$$

$$\mathbf{u}[k+1] = \mathbf{u}[k] + \boldsymbol{\Delta}\mathbf{u}[k], \ \mathbf{u} = \begin{pmatrix} z \\ f \end{pmatrix}, \ \mathbf{u}_{\min} = \begin{pmatrix} 0 \\ 35 \end{pmatrix}, \mathbf{u}_{\max} = \begin{pmatrix} 100 \\ 65 \end{pmatrix} \tag{121b}$$

$$\mathbf{y} = \begin{pmatrix} p_{bh} \\ p_{wh} \\ q \end{pmatrix}, \mathbf{y}_{\min} = \begin{pmatrix} 0 \\ 1 \\ 30 \end{pmatrix}, \mathbf{y}_{\max} = \begin{pmatrix} \infty \\ 60 \\ 50 \end{pmatrix} \tag{121c}$$

which lead to the objective function

$$J(\mathbf{y}[k],\mathbf{u}[k],\boldsymbol{\Delta}\mathbf{U}) = (\mathbf{Y} - \mathbf{Y}_{ref})^T(\mathbf{I}_{N_y} \otimes \mathbf{Q})(\mathbf{Y} - \mathbf{Y}_{ref}) + \boldsymbol{\Delta}\mathbf{U}^T(\mathbf{I}_{N_u} \otimes \mathbf{R})\boldsymbol{\Delta}\mathbf{U} \tag{121d}$$

with $\mathbf{Q}$ being weights for the reference tracking error and $\mathbf{R}$ weights imposed on the control variation, to prevent abrupt changes in the control action. Vector $\mathbf{x}$ refers to the system state, and $\mathbf{g}(\cdot)$ is a static function that maps the states to the output. If the system defined in (116) is used as the prediction model, then $\mathbf{x} = \mathbf{y}$, in which case $\mathbf{g}(\cdot)$ is the identity function. However, this is not always the case, as a reliable prediction is not always available.

For the remainder of this case study, the ESN-PNMPC is compared with an NMPC controller that uses ESN as a model, and employs the prediction filter in the exact same way as the PMPC, which was implemented with the Single Shooting method (JORDANOU, Jean P.; ANTONELO, Eric Aislan; CAMPONOGARA, 2019).

The ESP-lifted oil well was implemented using CasADi (ANDERSSON et al., 2019), whereas the Python libraries *numpy* and *scipy* were used to implement both the Echo State Networks and controllers (NMPC and PNMPC). NMPC was implemented with CasADi and solved with IPOPT (MEHREZ, 2019), whereas PNMPC was coded in native Python and solved with CVXOPT (DAHL; VANDENBERGHE, 2020). The results were displayed using the *Matplotlib* library. A sampling time of $T_s = 1/12$ s was used for the application, based on a simple step-test of the system. The experiment was done entirely in a personal desktop possessing 8 GB RAM, with an *AMD Ryzen 5 1400 Quad-Core Processor*, that operates in 3.2 GHz. No GPU was used in the simulation and/or training.

Alongside other previously mentioned metrics, the experiments with the simulated ESP-lifted well also employs the NRMSE (Normalized RMSE) metric:

$$NRMSE = \sqrt{\frac{1}{N}\sum_{k=0}^{N}\left\|\frac{\mathbf{y}[k] - \mathbf{y}_{\mathsf{ESN}}[k]}{\mathbf{y}_{\max} - \mathbf{y}_{\min}}\right\|^2} \tag{122}$$

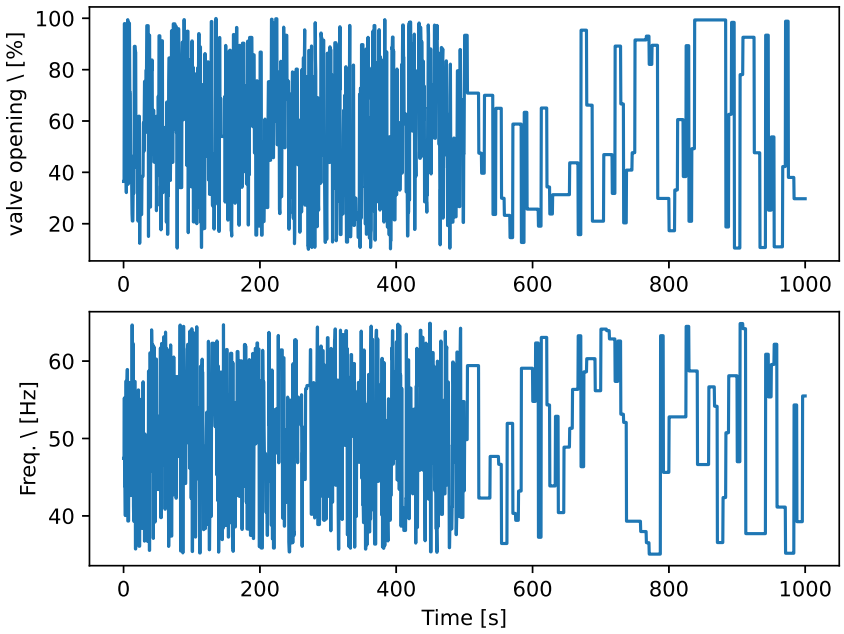


Figure 20 – APRBS excitation signal utilized to generate the dataset by inputting it into the ESP-lifted well. Valve opening presented as a percentage instead of in absolute value form.

where the real output $\mathbf{y}$ is compared to the ESN output $\mathbf{y}_{\text{ESN}}$, and the total control variation metric:

$$\Delta u_{\text{tot}} = \sum_{k=0}^{N} |\Delta u[k]| \tag{123}$$

3.2.3.2 System Identification

The APRBS signal input to the system as excitation has a period of 5 time steps ($5/12$ s) at the first half of the dataset, and a slower minimum period of 40 ($40/12$ s) steps for the rest of the simulation. The first half of the simulation is focused on identifying transients and responses to subtle variations in the control action, while the second half focuses on lower frequencies and the steady state.

Figure 20 depicts the APRBS excitation signal which generates the dataset. The signal has $12{,}000$ time steps ($1{,}000$ s) duration, ranging from $(z_{\min}, f_{\min}) = (0.1, 35)$ to $(z_{\max}, f_{\max}) = (1.0, 65)$ for the choke opening and pump frequency values, respectively. The system initial condition for the data gathering $p_{\text{bh}} = 70$ bar, $p_{\text{wh}} = 20$ bar, and $q = 36\ m^3/h$. The output signal for p_{bh}, p_{wh} and q is obtained by simulating the well model with the input sequence in Figure 20. An APRBS signal of 300 time steps was used as an excitation signal for a separate validation set, distinct from the training dataset shown in Figure 20.

Table 7 showcases the results from an experiment that aimed to assess the impact of the

Table 7 – NRMSE on the validation set for varying reservoir sizes of the ESN

Neurons	NRMSE	Average Runtime
50	0.129	0.39 s
100	0.116	0.40 s
150	0.112	0.45 s
200	0.109	0.51 s
250	0.107	0.60 s
300	0.106	0.63 s
350	0.107	0.69 s
400	0.106	0.73 s

number of neurons on ESN performance in terms of NRMSE. The performance improvement achieved in ESNs with more than 300 neurons was not significant, while the computation time increased almost linearly, which led to the selection of ESNs with 300 reservoir states, which were selected by grid search.

Table 8 – Hyperparameters used in the ESN

Parameter	Value
Leak rate (γ)	0.14
Size of the reservoir (N)	300
Spectral radius (ρ)	0.99
Input scaling (f_i^r)	0.1
Bias scale (f_b^r)	0.1
Warmup drop	200

The other parameters besides the number of neurons were decided using a grid search procedure separately, with each configuration being simulated once, leading up to the values for hyperparameters shown in Table 8. The leak rate γ is expected to be small because of the system settling time is longer when compared to the sampling time. The warm-up drop parameter is the number of training points, at the beginning of the dataset, that are dropped so that the ESN least squares does not calculate the weights using transient states. A regularization parameter of $\lambda = 10^{-8}$ was found via a 10-fold cross-validation (CV) using the training set.

Also, a 128-unit Long Short-Term Memory (LSTM) and a 64 Gated Recurrent Unit (GRU) network were trained using the same training data as the ESN, to compare their performance against the ESN in identifying the ESP-lifted oil well. Both networks were trained with the Adam algorithm, with a batch size of only one timestep (where BPTT takes into account the derivative calculation in the previous batch), and a learning rate of 0.001. These parameters were the ones that performed best in the series of experimental tests done with both networks. The stop criteria was the loss function (mean squared error) not decreasing for 10 epochs, indicating that the network weights had reached a local minimum in the optimization problem.

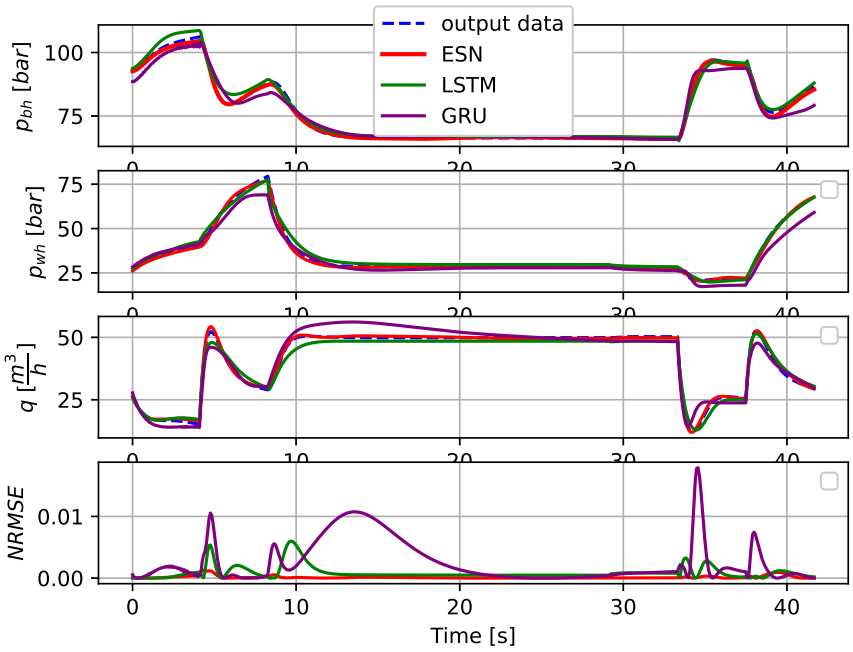


Figure 21 – Prediction of the ESN, LSTM, and GRU on test data in comparison to the ESP plant (blue dashed line) given the test excitation signal. The first plot presents the bottom-hole pressure p_{bh}, the second plot, the wellhead pressure p_{wh}, and the third plot, the flow. The fourth plot is the NRMSE at the given time-step.

The test data consists of 1000 time steps. Figure 21 regards simulations on this test data, showing both the solution of the differential equations that govern the ESP-lifted oil well (desired output data) and the three identified models. The simulations show that the ESN captures best the system dynamics, which therefore can be used as a surrogate model for the MPC strategies. Notice that the ESN response is the closest signal to the output data, as illustrated by the NRMSE curves depicted in Figure 21. Also, Table 9 compares the three networks in terms of execution time, NRMSE and epochs needed to conclude training. The results there showcase that the ESN is a suitable choice as the training is faster (linear least squares solution vs stochastic gradient descent for nonlinear least squares problem), and it performs better in terms of NRMSE.

Table 9 – Comparison of ESN model with LSTM and GRU for ESP-lifted well identification on test data.

	NRMSE	Training Epochs	Training Time (s)
ESN	**0.0135**	-	**106.89**
LSTM	0.0295	20	1438.11
GRU	0.0508	33	1649.82

3.2.3.3 Experiments Description

This experiment is about solving the following control problems using the ESN-PNMPC and the previously mentioned NMPC with an ESN model and the PNMPC prediction filter:

1. With the choke fully open ($z = 1$), the pump frequency f serves as a manipulated variable to track reference signals for the well bottom-hole pressure (p_{bh}), while adhering to the constraints in (93). Also, this experiment considers a variation in the reservoir pressure of $\Delta p_r = -10$ bar.
2. Both the choke opening z and pump frequency f serve as manipulated variables. The controller tracks a reference signal for p_{bh}, while maximizing the production flow q. Instead of including the maximization of the flow q directly into the cost function, the maximization appears in the form of a constant reference signal for q at an infeasible value for the problem in steady state. This reduces the production maximization to a reference tracking MPC problem such as in (93).

Notice that, when the controller must track the reference for a given variable, the constraint associated with that variable does not hold for the problem at hand.

3.2.3.4 Bottom-hole pressure tracking with pump frequency manipulation

For the first problem, the PNMPC (and NMPC) parameters set were:

$$\mathbf{Q} = 0.7 \tag{124a}$$

$$\mathbf{R} = 5 \tag{124b}$$

$$K = 1/3 \tag{124c}$$

$$\omega = 0.25 \tag{124d}$$

$$\Delta U_{\max} = -\Delta U_{\min} = 0.2 \tag{124e}$$

Figure 22 presents the results associated with the tracking of the bottom-hole pressure by manipulating the pump frequency. It shows the response of both controllers given a p_{bh} reference signal. Although the pure Single-Shooting NMPC shows slightly better performance in terms of settling time, the PNMPC is more conservative in its control signal. For small variations of the control action, both controllers generate similar outcomes, which can be explained by the fact that the PNMPC can be seen as a quadratic approximation of the NMPC solution. Also, the PNMPC requires considerably less time to run, which is remarkable given that the standard NMPC relies on advanced software with automatic differentiation and an NLP solver, whereas PNMPC uses just a QP solver and the simple recursive algorithm for derivative calculation.

Table 10 better illustrates the results in terms of IAE, the total control variation, the mean execution time of the control law calculation at a given time step, and the worst execution

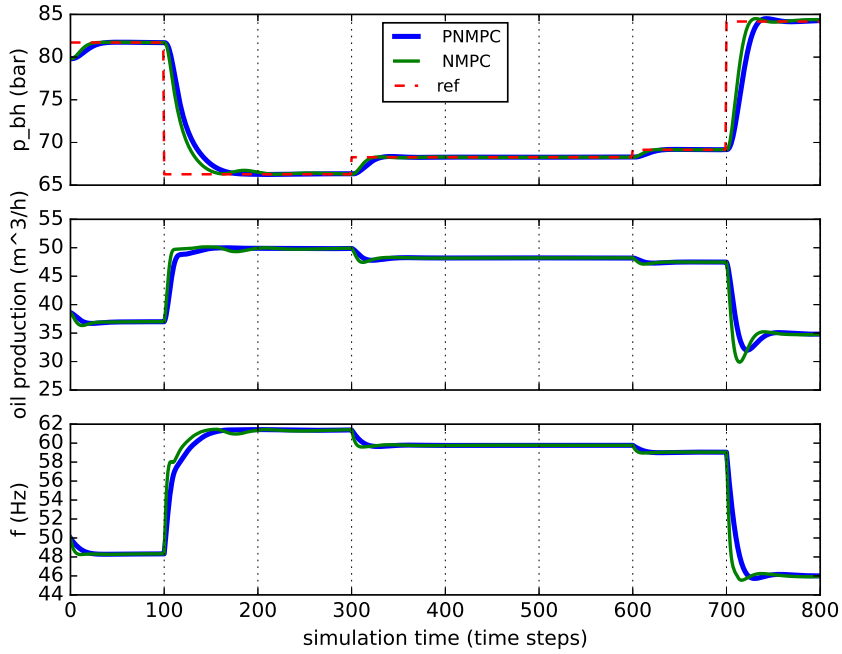


Figure 22 – Results of the experiment for the bottom-hole pressure tracking induced by both the SS ESN-NMPC and the ESN-PNMPC. The first plot regards the bottom-hole pressure, the second plot shows the oil flow q, and the third plot presents the manipulated variable: the pump frequency f.

Table 10 – Metrics for each controller in the bottom-hole pressure tracking problem.

	ESN-NMPC	ESN-PNMPC
IAE (p_{bh}) [bar]	**553.97**	712.46
Δu_{tot} [Hz]	34.29	**31.72**
Mean execution time [s]	1.19	**0.35**
Worst execution time [s]	1.73	**0.87**

time of a time step inside the simulation. The NMPC displays superior IAE, while the PNMPC is more conservative (Δu_{tot}) and more efficient in terms of execution time, by a factor of more than 3 on average. Seeing the PNMPC as an approximation to the NMPC, the relative error between the control action of each controller RE_{MPC} (the solution to their respective optimization problems) can serve as metric of how close the control action of both controllers are:

$$RE_{MPC} = \sum_{k=1}^{N} \left| \frac{u_{PNMPC}[k] - u_{NMPC}[k]}{u_{NMPC}[k]} \right| = 0.06, \tag{125}$$

where N is the total number of time steps. Such a value for RE_{MPC} demonstrates that the trajectory linearization of the PNMPC serves well as an approximation to the NMPC for ESNs, considering this specific type of MPC problem where constraints are not being violated. Notice that the decision variable in the PNMPC is the control increment, while in the NMPC it is

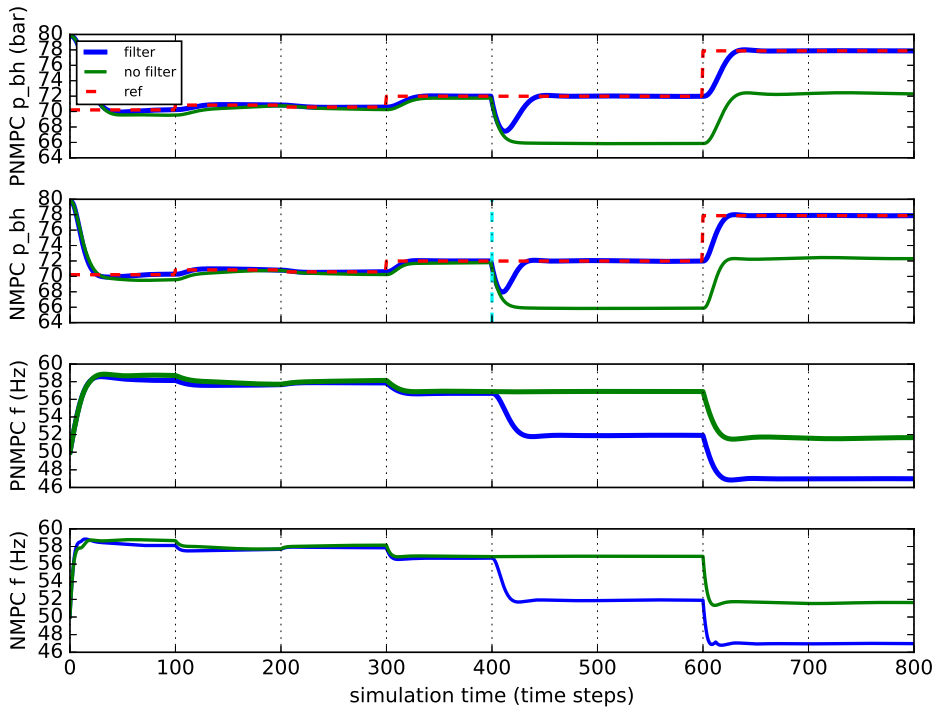


Figure 23 – Results of the experiment for the bottom-hole pressure tracking with disturbance rejection induced by both the SS ESN-NMPC and ESN-PNMPC. The first and the second plots show the bottom-hole pressure tracking by PNMPC and NMPC, respectively. Both plots have a blue, thick line representing the controller with a filter, and a green line for the controller without a filter. The third and fourth plots are the pump frequency for PNMPC and NMPC respectively. The time instant where disturbance on the reservoir pressure p_r takes place is shown as a vertical cyan dashed line at timestep 400.

the control action itself. Therefore, the NMPC calculates a new control action at each time step, and the PNMPC approximates the problem given the previous control action and outputs, computing an incremental approximation to the whole NMPC.

This experiment also considers a disturbance for this control problem, by injecting a disturbance Δp_r that decreases the reservoir pressure in 10 bar midway in the simulation. Each controller performs reference tracking under the same reference signal as the no disturbance simulation. Two cases were considered: both controllers with and without the presence of the error correction filter.

Figure 23 demonstrates the result of said experiment, where the effect of the prediction filter in the control loop is clearly shown. The -10 bar disturbance in the reservoir pressure induces a modeling bias in the ESN, which is why a constant error is present in the non-filtered counterpart of the controllers. Meanwhile, both controllers with the filter managed to bring the bottom-hole pressure to the desired reference, with a slightly lower peak for the NMPC. A slight reference error is also present for the non-filtered controllers before the disturbance is applied, as the ESN is not an exact model of the ESP-lifted oil well.

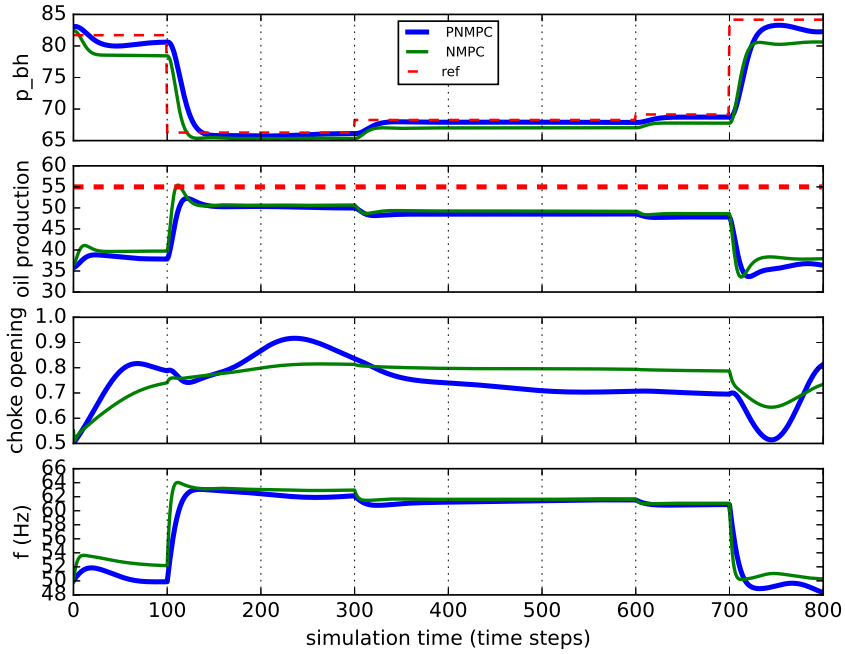


Figure 24 – Results for the bottom-hole pressure tracking while maximizing oil production for both the ESN-NMPC and the ESN-PNMPC. The first plot shows the controlled bottom-hole pressure. The second plot shows the oil flow q with the target oil production given by the horizontal dashed red line. The bottom plots present the manipulated variables choke opening z, and pump frequency f.

3.2.3.5 Bottom-hole pressure tracking with target oil production

For the second proposed problem, where the controller tracks a reference signal for the bottom-hole pressure while approaching an unattainable production flow target to maximize oil production, the parameters utilized are:

$$\mathbf{Q} = \begin{pmatrix} 0.7 & 0 \\ 0 & 0.1 \end{pmatrix} \tag{126a}$$

$$\mathbf{R} = \begin{pmatrix} 3 & 0 \\ 0 & 5 \end{pmatrix} \tag{126b}$$

$$K = 1/3 \tag{126c}$$

$$\omega = 0.25 \tag{126d}$$

$$\Delta U_{\max} = -\Delta U_{\min} = 0.2\mathbf{1} \tag{126e}$$

Notice that in this formulation, the parameters related to f and the error for p_{bh} are the same as in the previous problem. In this formulation, $\mathbf{u} = (z,f)^T$ and $\mathbf{y_{ref}} = (p_{bh}^{ref},q^{ref})^T$.

Figure 24 showcases the results of the experiments. The p_{bh} tracking is hindered by the need to bring the flow as close as possible to $q^{ref} = 55\ m^3/h$ (red horizontal dashed line). Besides, for this experiment, the control action of the controllers differs from each other more significantly, *i.e.*, PNMPC control action varies more than NMPC one. The NMPC seems to

prioritize production more than the bottom-hole pressure, which explains the conservativeness of the generated control action. Table 11 better illustrates this phenomenon with the results, where PNMPC performed better with respect to the p_{bh} tracking, while NMPC produced slightly more oil since it showed lower IAE for q.

Table 11 – Metrics for each controller in the problem of pressure tracking with target production.

	ESN-NMPC	ESN-PNMPC
IAE (p_{bh}) [bar]	1724.65	$\mathbf{870.78}$
IAE (q) [m^3/h]	$\mathbf{7222.01}$	8106.24
Δu_{tot} (z)	$\mathbf{0.37}$	0.95
Δu_{tot} (f) [Hz]	$\mathbf{21.45}$	25.34
Mean execution time [s]	1.07	**0.45** s
Worst execution time [s]	1.23	**1.11** s

The total relative error of the choke valve opening between each controller was $RE_{MPC}(z) = 47.12$, while for the pump frequency it was $RE_{MPC}(f) = 11.04$. There is a significant difference between each controller because the system is being driven into a point at the limit of its own constraints, therefore provoking entirely different reactions from each controller. From a qualitative point of view, the PNMPC calculates the control action based on the current operating point of the optimization problem, defined by the initial free response trajectory, while the NMPC prediction model must compose a whole trajectory from scratch, which changes how both controllers handle constraint limits. The PNMPC model is linear in terms of the control increment, therefore it will seek greedy solutions to handle the constraints, whereas the NMPC handles the control trajectory more precisely. This explains why the PNMPC prioritizes the bottom-hole pressure, since minimizing its tracking error is easier from the controller's point of view.

3.3 SUMMARY

This chapter presented the ESN-PNMPC, and report results from experiments using the four-tank system, the two-wells one-riser oil and gas production platform system, and the ESP-lifted well to experiment with various aspects of the controller application. There are some remarks about future experimental work:

- All the parameters were decided through some form of grid search, however it is also possible to utilize Bayesian optimization tools such as the Tree-Structured Paizen Estimator (WATANABE, 2023).
- Using computation time as a time metric is flawed in the aspect that it is sensitive to the machine that the simulation is being run. A proposal for a metric that can be used instead in future works is the number of floating point operations, as it is machine-independent.

- The total control variation for each experiment with the exception of the ESP-lifted well were omitted as the discussion for those cases was purely focused on the tracking and disturbance rejection aspects. It is an important measure in directly evaluating conservativeness of a controller, and it is recommended to be addressed in future works.
- Other metrics instead of IAE could be employed, such as the integral time-weighted absolute error, that penalizes transients more, or any type of squared error, which were used for minimization but not as a metric. An integral squared error metric is naturally more interesting for optimization than the IAE because of the easier derivative calculation.

Although the ESN-PNMPC is shown to perform well for all the proposed applications, as each model is able to be replicated with and ESN and there was data abundance, the method still presents a few theoretical limitations:

- The ESN-PNMPC does not have an established stability analysis method. The experiments were successful, but there is no way to analytically prove that an ESN-PNMPC controller would be stable for a given control problem. Also, the industry relies on simple, yet fast and reliable controllers such as the PI and DMC. Alongside a systematic hyperparameter tuning method, methods to reduce the ESN complexity might turn the ESN-PNMPC more indutry-friendly.
- Reservoir Computing requires by definition that the number of states in an ESN be magnitudes larger than the problem dimension. This might cause computational overhead when solving optimization and optimal control problems using ESN models.

The next two chapters are dedicated to contributing on resolving each issue raised.

4 INVESTIGATION OF POD METHODS FOR ECHO STATE NETWORKS

This chapter describes the development of model order reduction methods for ESNs. It discusses POD, the Energy Contribution metric, DEIM, and the stability issues regarding DEIM. The chapter showcases several experiments regarding the application of these tools in an ESN identifying different kinds of systems.

4.1 OVERVIEW

In Reservoir Computing, the large number of dynamic states in the reservoir is an essential characteristic, as the output, being a linear combination of them, can represent a more extensive repertoire of dynamics. However, using ESNs as dynamic models for problems such as optimization and MPC (Model Predictive Control) (CAMACHO; BORDONS, 1999) may pose a problem, as the higher the number of states in the ESN is, the larger the optimization problem. Because the number of states in the ESN inherently dominates the number of inputs and outputs in such applications, a large reservoir size renders the optimization problem inherently larger and harder to solve.

As ESNs are high-dimensional, model order reduction methods can find approximate ESN models with a considerably smaller number of states but which still keep the properties and performance of the original high-dimensional ESN. To that end, this problem counts on Proper Orthogonal Decomposition (POD) (CHATURANTABUT; SORENSEN, 2010) as a possible solution, which applies Singular Value Decomposition (SVD) to find an optimal linear transformation that represents the state space of a large dynamical system in a more compact form. POD is already widely used to reduce the number of states of large dynamical models, especially phenomenological models such as a gas reservoir simulator (WANG, Yi; YU; WANG, Ye, 2018) with tens of thousands of variables. However, POD has one disadvantage concerning nonlinear systems: although the method can reduce the number of states, it does not reduce the computation number of nonlinear functions. There are developments of interpolation methods, such as the Discrete Empirical Interpolation Method (DEIM) (CHATURANTABUT; SORENSEN, 2010), to mitigate the issue by pivoting and approximating the nonlinear portion of the given model computation. Both POD and DEIM can find lower-dimensional networks that are equivalent to the original ESN and, thus, have the potential to alleviate the computational burden of simulations that depend on the size of the trained ESN.

The main purpose of this chapter is the experimentation of POD and DEIM so that one can obtain a reduced-order model equivalent for an already-trained ESN. To such end, the ESN reduction through POD is employed in three different contexts: a Memory Capacity (MC) (JAEGER, 2002) evaluation experiment; a NARMA10 difference equation (SAKEMI et al., 2020); and a simulated oil platform containing two gas-lifted oil wells and one riser (JORDANOU, Jean P.; ANTONELO, Eric Aislan; CAMPONOGARA, 2019). Additionally, this chapter shows results using DEIM-based reduction for the ESN in the first and last experiments

mentioned above. This chapter compares the performance of the reduced ESN to the original (non-reduced) ESN in the three experiments and another ESN with the same size as the reduced ESN in the MC and NARMA experiments.

The main contributions depicted in this chapter are two-fold: (1) the development of efficient computational frameworks for implementing large echo-state network models in a variety of applications, achieved via model-order reduction (MOR) techniques; and (2) assessing the trade-offs between low-complexity reservoir models, obtained from the application of model order reduction (MOR), and the large baseline model in terms of numerical accuracy. The low-complexity models, despite their relatively small state-space dimensions, demonstrate comparable representation power to the large baseline model. As such, this chapter showcases contributions to this nascent field of applications of MOR strategies to reservoir computing, which can potentially improve computational performance in modeling, control, and optimization. Specifically, the findings of the work depicted in this chapter are the following:

- The memory capacity of an ESN reduced by POD is generally higher than that of a non-reduced ESN of equivalent size. This difference in memory capacity is more significant as the desired ESN gets smaller in size.
- Given two echo state networks with the same number of states, the ESN obtained from POD reduction is likelier to perform better in a given task. This property is more evident and relevant when the desired reservoir is small.
- By employing a MOR method on ESNs, this work shows that small ESNs are robust and performant, improving their suitability for real-time or embedded applications with memory limitations.
- DEIM reduction alone for ESNs does not achieve satisfactory results compared to pure POD reductions.

In broader terms, the main implication of these findings is that a smaller version of an ESN, obtained by model order reduction, can achieve nearly equivalent behavior to the original (and larger) ESN, thus making dynamic reservoirs more compact. The new model can serve as a proxy model in optimization and predictive control, as an observer, and in other related tasks, addressing the issue of computational cost in a reservoir consisting of a large number of internal states (reservoir size), which can be orders of magnitude larger than the number of inputs and outputs.

This chapter is organized as follows: Section 4.2 contains related works, Section 4.3 describes POD and DEIM, Section 4.4 reports on the case studies and experimental testing for the reduced ESN, and Section 4.6 concludes the chapter.

The theory and results shown in this chapter are directly from (JORDANOU, Jean Panaioti et al., 2023), a publication resulting from the research undertaken in this dissertation.

4.2 RELATED WORK

The following works address the issue of reducing the model size in reservoir computing. One of them is (SAKEMI et al., 2020), where they propose reducing the number of states by considering the output as a linear combination of the states at different instants in time, comparing to an original ESN through the Information Processing Capacity (IPC) metric, and also applying the proposal to a NARMA system and the generalized Hénon-map. The solution raises the effective number of states as a multiple of the delay or "drift-state" number utilized. The architecture is hardware-friendly, easing the computation compared to a standard ESN.

Another example is the work (WHITEAKER; GERSTOFT, 2022), where they propose to employ the controllability matrix of the ESN as a means to find a so-called minimal ESN, which would be the ESN with the smallest reservoir that could reproduce the task at hand. They train the ESN for a particular task, obtain the controllability matrix at given points, and define its rank as a new candidate reservoir size. An extensive search procedure is then performed to find the optimal ESN at that size; however, there is no direct connection between the larger and the smaller ESN. In summary, the method in (WHITEAKER; GERSTOFT, 2022) proposes a useful way of finding a minimal reservoir for a task. In comparison, our work follows a different direction: reducing the size of the network through POD. Another work (LIU, W. et al., 2022) proposes a different approach to reducing reservoir size, which calculates the correlation between each neuron and eliminates the reservoir neurons with the highest correlation.

The necessary large number of reservoir states in an ESN implies a complex computational model, therefore works such as (YANG, C.; WU, Z., 2022) employ methods of so-called "network size reduction," which perform multi-objective optimization on the output weights and minimize not only the least-square error but also the number of non-zero elements in the output weights. Enforcing sparseness is ideal for simplifying computations with the ESN. Another work that follows this line of reasoning is (RODAN; TINO, 2010), where they enforce a minimum complexity ESN by forcing the ESN reservoir to follow a deterministic form (*i.e.*, a circular reservoir).

In (LØKSE; BIANCHI; JENSSEN, 2017), they propose to add the reservoir dimensionality reduction into the architecture via Principal Component Analysis (PCA) and calculate the output layer based on the PCA output instead of the reservoir states. They affirm that this enhances the dynamic properties of the resulting ESN concerning the system identified and improves the network generalization capabilities. Also, applying dimensionality reduction in the states renders the ESN a tool for dynamic system analysis. In this sense, our POD-ESN method is similar to PCA regarding obtaining the new state space but goes beyond (LØKSE; BIANCHI; JENSSEN, 2017) by embedding the reduction achieved in the reservoir's state update equation. In other words, the reservoir recurrent simulation is executed in the reduced state space with POD-ESN, which does not happen in (LØKSE; BIANCHI; JENSSEN, 2017).

Another approach of reduction in reservoir computing, not involving POD, is proposed in

(HALUSZCZYNSKI et al., 2020). Their idea involves procedurally removing neurons according to the output weight value, which they curiously discovered that the network performance improves (given the Lorentz system as an application) by removing the neurons associated with large output weights. They thoroughly analyze the effect of removing different types of nodes in the ESN.

4.3 MODEL ORDER REDUCTION

This section proposes Model Order Reduction (MOR) methods for reducing the reservoir dimensionality in ESNs, specifically the Proper Orthogonal Decomposition (POD) and the Discrete Empirical Interpolation Method (DEIM).

4.3.1 Proper Orthogonal Decomposition

The Proper Orthogonal Decomposition is a method to find a linear transformation (CHEN, C.-T., 1998) $\mathbf{T}$ for a given system that maps a high-dimensional state space into a reduced one, namely:

$$\mathbf{x} = \mathbf{T}\mathbf{z} \tag{127}$$

where $\mathbf{x}$ is a vector of dimension n and $\mathbf{z}$ is a vector of dimension $m \ll n$, so that $\mathbf{T} \in \mathbb{R}^{n \times m}$.

The transformation itself is akin to a similarity transformation, with the main difference being that $\mathbf{T}$ lacks an inverse for not being a square matrix. However, the $\mathbf{T}$ resulting from POD is orthonormal ($\mathbf{T}^T\mathbf{T} = \mathbf{I}$), so the transpose is used in place of an inverse.

Finding $\mathbf{T}$ requires gathering snapshots of the states in a given dynamical system response, akin to gathering data in a machine learning problem. The columns of the snapshot matrix $\mathbf{X} \in \mathbb{R}^{n \times N}$ are the states $\mathbf{x}[k] \in \mathbb{R}^n$, where N is the number of snapshots such that $N \geq n$. Then, one wishes to minimize the error induced by projecting the original state onto the reduced space and back, which leads to the following error function:

$$E(\mathbf{T}) = \sum_{k=1}^{N} \Big(\mathbf{x}[k] - \mathbf{T}\underbrace{\mathbf{T}^T\mathbf{x}[k]}_{\mathbf{z}[k]}\Big)^2 \tag{128}$$

The second term is $\mathbf{x}$ *projected* onto the reduced space of $\mathbf{z}$, and then *lifted* back. The optimal $\mathbf{T}$ is obtained through singular value decomposition (SVD) (SUN; XU, M.-h., 2017), decomposing $\mathbf{X}$ in the following form:

$$\mathbf{U_{svd}}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{X} \tag{129}$$

where $\mathbf{U_{svd}}$ contains the left singular vectors and has dimension $n \times n$, $\mathbf{\Sigma}$ contains the singular values and has dimension $n \times N$, with only n non-zero columns, and $\mathbf{V}$ contains the right singular vectors and has dimension $N \times N$. Consider that $\mathbf{\Sigma}$ is sorted from the largest to the smallest singular value. POD does not use the right singular vector matrix $\mathbf{V}$.

The transformation $\mathbf{T}$ that minimizes $E(\mathbf{T})$ is found by concatenating the columns with the m largest corresponding singular values from $\mathbf{U_{svd}}$. One seeks a truncation so that the reduced system energy is close to the original, measured by:

$$\epsilon = \sum_{j=1}^{m} \epsilon_j \qquad\qquad \epsilon_j = \sigma_j / \sum_{i=1}^{n} \sigma_i \tag{130}$$

where ϵ is the total energy contribution of the singular values maintained in the reduced-order model, σ_j is the j^{th} highest singular value, ϵ_j is the energy contribution of that given singular value, and m is the reduced state dimension. The energy contribution of the remaining singular values in the reduction is a metric on how close the reduced-order model is to the original system regarding information. This work considers measuring the energy contribution of each singular value of the original signal and truncate $\mathbf{U_{svd}}$ to obtain $\mathbf{T}$ so that ϵ reaches a desired energy contribution value (*e.g.*, $\epsilon = 0.95$, so that the reduced system has 95% of the original system's energy). In other words, the reduced-order model carries ϵ information of the original system. After obtaining $\mathbf{T}$ for the dimension reduction through the process above, the reduced ESN dynamics can be expressed as follows:

$$\begin{aligned}\mathbf{z}[k+1] &= (1-\gamma)\mathbf{z}[k] \\ &\quad + \gamma \mathbf{T}^T \mathbf{f}(\mathbf{W_r^r T z}[k] + \mathbf{W_i^r u}[k] + \mathbf{W_b^r}) \qquad (131)\text{a} \\ \mathbf{y}[k+1] &= \mathbf{W_r^o T z}[k+1], \qquad (131)\text{b}\end{aligned}$$

Observe, from the operation $\mathbf{T}^T\mathbf{f}(\cdot)$, that the reduced-order ESN does not reduce the number of computations by only performing POD on it. In fact, to compute the element-wise $\tanh$, $\mathbf{T}$ brings the dimension back to the original state-space size, which is to be reduced again with $\mathbf{T}^T$, increasing the number of computations. This computational increase is inherent in POD for nonlinear systems and will be dealt with by the method described in the next section.

4.3.2 Discrete Empirical Interpolation

The Discrete Empirical Interpolation Method (DEIM) is an approximation method to circumvent the POD computation issue (CHATURANTABUT; SORENSEN, 2010), which consists of state projection and lifting operations to compute state transitions in the reduced-order model. The core idea of DEIM is to approximate the nonlinear term of a dynamic system as a polynomial interpolation that resembles the strategy employed in POD. Given the following discrete-time nonlinear system:

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{f}(\mathbf{x}[k]), \tag{132}$$

where the nonlinear function is elementwise, meaning that

$$\mathbf{f} = \Big(f(\mathbf{x}), f(\mathbf{x}), \ldots, f(\mathbf{x})\Big) \tag{133}$$

for a given function f such as $\tanh$. Notice that the system is divided into linear and nonlinear portions. Applying the POD ($\mathbf{x} = \mathbf{Tz}$) into such a system yields:

$$\mathbf{z}[k+1] = \mathbf{T}^T\mathbf{A}\mathbf{T}\mathbf{z}[k] + \mathbf{T}^T\mathbf{f}(\mathbf{T}\mathbf{z}[k]) \tag{134}$$

The nonlinear mapping $\mathbf{f}$ of the dynamic system can be approximated as follows:

$$\mathbf{P}^T\mathbf{f}(\mathbf{T}\mathbf{z}[k]) \approx \mathbf{P}^T\mathbf{U}\mathbf{c}[k] \tag{135}$$

where $\mathbf{U} \in \mathbb{R}^{n\times m}$, which is obtained from the same POD as $\mathbf{T}$, however with a different number m of singular vectors, with n being the number of states, and $\mathbf{P}$ being a a pivoting matrix of the same dimension as $\mathbf{U}$. DEIM interprets that a linear combination, with basis $\mathbf{U}$ and the elements $\mathbf{c}[k]$ as function coefficients, approximates the elementwise function $\mathbf{f}$.

After obtaining $\mathbf{U}$ from $\mathbf{U_{svd}}$, one can then obtain $\mathbf{P}$ with the following procedure (CHATURANTABUT; SORENSEN, 2010):

1. The index and value of the largest element of the first left-singular vector is stored in a list. $\mathbf{P}$ starts as a column matrix with the only non-zero element being the value 1 at the row corresponding to this index.
2. For each column $l \geq 2$ of the POD left-singular vectors (where $\widetilde{\mathbf{U}}_l$ is a matrix with the first $l-1$ columns of $\mathbf{U}$):
    a) find $\mathbf{c}$ where $(\mathbf{P}^T\widetilde{\mathbf{U}}_l)\mathbf{c} = \mathbf{P}^T\mathbf{u}_l$, where $\mathbf{u}_l$ is the left-singular vector corresponding to the l^{th} column of $\mathbf{U}$.
    b) Calculate $\mathbf{r} = \mathbf{u}_l - \widetilde{\mathbf{U}}_l\mathbf{c}$ and store the maximum absolute value and index of $\mathbf{r}$ in a list. Add a new column to $\mathbf{P}$ according to the obtained index.
3. Output: Pivoting matrix $\mathbf{P}$ according to the order dictated by the index list obtained.

This procedure guarantees that $\mathbf{P}^T\widetilde{\mathbf{U}}_l$ is always nonsingular; thus $\mathbf{c}$ is the unique solution to the linear system in step 2 (CHATURANTABUT; SORENSEN, 2010). Letting $\mathbf{U}$ be the matrix of left singular values obtained from the procedure, it follows from (135) that:

$$\mathbf{c}[k] = (\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{f}(\mathbf{T}\mathbf{z}[k]) \tag{136}$$

The result from (136) leads to the DEIM function interpolation:

$$\hat{\mathbf{f}}(\mathbf{T}\mathbf{z}[k]) \approx \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{f}(\mathbf{T}\mathbf{z}[k]) \tag{137}$$

This function approximation has an ℓ_2 error bound of the following form (CHATURANTABUT; SORENSEN, 2010):

$$e_{\ell_2}(\mathbf{f}) \leq \|(\mathbf{P}^T\mathbf{U})\|_2\|(\mathbf{I} - \mathbf{U}\mathbf{U}^T)\mathbf{f}(\mathbf{T}\mathbf{z}[k])\| \tag{138}$$

where, in turn:

$$\|(\mathbf{P}^T\mathbf{U})\|_2 \leq (1+\sqrt{2n})^{m-1}\|\mathbf{u}_1\|_\infty^{-1} \tag{139}$$

with $\mathbf{u}_1$ being the first column of $\mathbf{U}$ and n being the number of original states.

The main advantage of DEIM is that, as $\mathbf{f}$ is an element-wise nonlinear function, the following equality holds:

$$\underbrace{\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T}_{\mathbf{T}_1\in\mathbb{R}^{n\times n}}\underbrace{\mathbf{f}(\mathbf{T}\mathbf{z}[k])}_{\mathbf{f}:\mathbb{R}^n\to\mathbb{R}^n} = \underbrace{\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}}_{\mathbf{T}_2\in\mathbb{R}^{n\times m}}\underbrace{\mathbf{f}(\mathbf{P}^T\mathbf{T}\mathbf{z}[k])}_{\mathbf{f}:\mathbb{R}^m\to\mathbb{R}^m} \tag{140}$$

The difference between the right-hand side and left-hand side of this equation is better seen in a compact form,

$$\mathbf{T}_1\mathbf{f}(\mathbf{T}\mathbf{z}[k]) = \mathbf{T}_2\mathbf{f}(\mathbf{P}^T\mathbf{T}\mathbf{z}[k])$$

where $\mathbf{T_1}$ has n columns, which yields the same computation problem as the original Galerkin projection, whereas $\mathbf{T_2}$ has m columns, which is the reduced state space. This simple difference grants huge computational savings since the online calculations would be performed in terms of the reduced dimension m, $m \ll n$, which mitigates the computation issues regarding the POD method.

The DEIM and POD model order reduction for the ESN is obtained by applying DEIM from Eq. (140) into the already reduced POD-ESN at (131):

$$\mathbf{z}[k+1] = (1-\gamma)\mathbf{z}[k] + \gamma\mathbf{T}^T\mathbf{T_2}\mathbf{f}\left(\mathbf{P}^T\mathbf{W_r^r}\mathbf{T}\mathbf{z}[k] + \mathbf{P}^T\mathbf{W_i^r}\mathbf{u}[k] + \mathbf{P}^T\mathbf{W_b^r}\right) \tag{141a}$$

$$\mathbf{y}[k+1] = \mathbf{W_r^o}\mathbf{T}\mathbf{z}[k+1], \tag{141b}$$

The property $\mathbf{P}^T\mathbf{f}(\cdot) = \mathbf{f}(\mathbf{P}^T)$ holds for elementwise operations, which justify the matrix placement in the DEIM reduced-order ESN.

4.3.3 Stability Loss in DEIM

According to (SELGA; LOHMANN; EID, 2012), a contractive linear system is guaranteed to retain stability when applying POD for model order reduction; therefore, if the ESN is contractive, the POD-ESN is guaranteed to retain stability. However, DEIM has no such property. Assume an equilibrium point $\mathbf{x_{eq}}$ of the ESN, and a fixed input $\mathbf{u}$,

$$\mathbf{x_{eq}} = \mathbf{f}(\mathbf{W_r^r}\mathbf{x_{eq}} + \mathbf{W_i^r}\mathbf{u} + \mathbf{W_b^r}) \tag{142}$$

and its reduced mapping $\mathbf{z_{eq}} = \mathbf{T}^T\mathbf{x_{eq}}$. The Jacobian of the full and reduced order model are:

$$J(\mathbf{x_{eq}}) = (1-\gamma)\mathbf{I} + \gamma\mathbf{f}'(\mathbf{g}(\mathbf{x_{eq}}))\mathbf{W_r^r} \tag{143}$$

$$J(\mathbf{z_{eq}}) = (1-\gamma)\mathbf{I} + \gamma\mathbf{T}^T\mathbf{f}'(\mathbf{g}(\mathbf{T}\mathbf{z_{eq}}))\mathbf{W_r^r}\mathbf{T} \tag{144}$$

where:

$$\mathbf{g}(\mathbf{x}) = \mathbf{W_r^r}\mathbf{x} + \mathbf{W_i^r}\mathbf{u} + \mathbf{W_b^r} \tag{145}$$

Since $\mathbf{f}'$ is a diagonal matrix where each element belongs to the interval $(0,1]$ for being the elementwise derivative of the $\tanh$ function, the stability of the ESN in both cases is

governed by $\mathbf{W_r^r}$ at an equilibrium point. Also, as per (SELGA; LOHMANN; EID, 2012), the POD reduction retains the stability of the ESN. Summing up, the original and reduced-order ESNs are stable provided that the spectral radius of $\mathbf{W_r^r}$ is smaller than 1.

With DEIM, however, the stability is not retained, as shown by calculating the Jacobian of an ESN reduced by both POD and DEIM:

$$\mathbf{J_{DEIM}}(\mathbf{z}) = (1-\gamma)\mathbf{I} + \gamma\mathbf{T}^T\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{f}'(\mathbf{P}^T\mathbf{g}(\mathbf{Tz}))\mathbf{P}^T\mathbf{W_r^rT} \tag{146}$$

Notice that, when compared to the Jacobian of the POD-ESN, the term $(\mathbf{P}^T\mathbf{U})^{-1}$ can amplify the Jacobian to the point that the ESN dynamic system has an unstable Eigenvalue, despite POD-ESN being stable. This term represents the pivoting of the truncated singular vectors associated with DEIM. This stability discussion implies that pure DEIM may not apply to ESN.

4.3.3.1 Stabilizing DEIM on POD-ESN

A method to stabilize DEIM was developed in (HOCHMAN; BOND; WHITE, 2011). A similar and independent work on MOR for ESNs (WANG, H.; LONG; LIU, X.-X., 2022) applies this very method to the case where ESNs are unbiased.

The idea behind the stabilization is straightforward. As the previous discussion shows, DEIM amplifies the Jacobian of the interpolating function. The solution in (HOCHMAN; BOND; WHITE, 2011) involves decomposing the function so that the interpolated terms are forced to have a very small Jacobian near a given equilibrium point $\mathbf{x_0}$.

Given the dynamic system:

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{f}(\mathbf{x}[k]) \tag{147}$$

Decompose $\mathbf{f}$ as:

$$\mathbf{f}(\mathbf{x}[k]) = \mathbf{f_p}(\mathbf{x}[k]) + \mathbf{J}(\mathbf{x_0})(\mathbf{x}[k] - \mathbf{x_0}) \tag{148}$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{f}$ and:

$$\mathbf{f_p}(\mathbf{x}[k]) = \mathbf{f}(\mathbf{x}[k]) - \mathbf{J}(\mathbf{x_0})(\mathbf{x}[k] - \mathbf{x_0}) \tag{149}$$

Note that the Jacobian of $\mathbf{f_p}$ near $\mathbf{x_0}$ is very close to zero, because $\mathbf{f_p}$ has the Jacobian of $\mathbf{f}$ being canceled by the Jacobian of the second term (which is the Jacobian of $\mathbf{f}$ at the equilibrium). Thus, if DEIM interpolation is performed for $\mathbf{f_p}$ (with $\mathbf{J}(\mathbf{x_0})(\mathbf{x} - \mathbf{x_0})$ treated as a linear portion), the stability effect of DEIM would be minimal, as the method is being performed over a portion of the dynamic system that has an almost insignificant contribution to the Jacobian.

By decomposing the $\mathbf{f}$ in POD-ESN (131) to obtain $\mathbf{f_p}$, and performing DEIM on $\mathbf{f_p}$ instead of $\mathbf{f}$ the result over POD-ESN is as follows:

$$\begin{aligned}\mathbf{z}[k+1] = (1-\gamma)\mathbf{z}[k] + \gamma(\mathbf{T}^T - \mathbf{T}^T\mathbf{T_2}\mathbf{P}^T)\mathbf{f}'(\mathbf{g}(\mathbf{Tz_0}))\mathbf{W_r^rT}(\mathbf{z}[k] - \mathbf{z_0}) + \\ \gamma\mathbf{T}^T\mathbf{T_2}\mathbf{f}\left(\mathbf{P}^T\mathbf{W_r^rTz}[k] + \mathbf{P}^T\mathbf{W_i^r}\mathbf{u}[k] + \mathbf{P}^T\mathbf{W_b^r}\right)\end{aligned} \tag{150}$$

Since (WANG, H.; LONG; LIU, X.-X., 2022) assumes the ESN to be unbiased, it is easy to note that, for that specific version of the ESN, $\mathbf{x_0} = \mathbf{0}$ and $\mathbf{J}(\mathbf{x_0}) = \mathbf{W_r^r}$, making the formulation of this solution for the ESN trivial, having the following form:

$$\mathbf{z}[k+1] = (1-\gamma)\mathbf{z}[k] + \gamma(\mathbf{T}^T - \mathbf{T}^T\mathbf{T_2}\mathbf{P}^T)\mathbf{W_r^r}\mathbf{T}\mathbf{z}[k] + \gamma\mathbf{T}^T\mathbf{T_2}\mathbf{f}\left(\mathbf{P}^T\mathbf{W_r^r}\mathbf{T}\mathbf{z}[k] + \mathbf{P}^T\mathbf{W_i^r}\mathbf{u}[k]\right) \quad (151)$$

However, in case there is bias present in the ESN, $\mathbf{x_0}$ would be the solution to the equation:

$$\mathbf{x_0} = \mathbf{f}(\mathbf{W_r^r}\mathbf{x_0} + \mathbf{W_i^r}\mathbf{u} + \mathbf{W_b^r}), \quad (152)$$

or, alternatively, the ESN state at a steady state in a simulation. In comparison to the zero bias result, the Jacobian value would be multiplied by $\mathbf{f}'(\mathbf{x_0})$, which is a diagonal matrix with elements $\in (0,1)$ as $\mathbf{f} = \tanh(\cdot)$.

Naturally, since $\mathbf{x_0}$ is the equilibrium state,

$$\mathbf{x_0} = \mathbf{T}\mathbf{z_0} \quad (153)$$

4.4 APPLICATIONS

This section presents results from experiments with reduced-order ESNs for three case studies, along with a preliminary analysis on the singular values of the ESN snapshots.

4.4.1 Preliminary Study: Energy contribution distribution in Echo State Networks

POD and DEIM originate from applying SVD into the ESN state response matrix, obtained from exciting the ESN's reservoir with an input signal. Thus, the SVD does not depend on the output layer. To test the influence of input signals into the singular values of the state snapshots, this experiment considers initializing 20 different single-input ESN reservoirs and apply SVD into the snapshots of the response obtained from the reservoir, given as inputs with $10{,}000$ timesteps:

- A white noise following the normal distribution $\mathcal{N}(0,1)$.
- Four different APRBS (Amplitude-modulated Pseudo-Random Binary Signal) random stair signals, defined by their minimum period, *i.e.*, 10 timesteps, 100 timesteps, 500 timesteps, and $1{,}000$ timesteps.
- A concatenation in time of all the signals above.

The input signals for the experiments are shown in Figure 25. Note that this discussion concerns only the state dynamics of the reservoir; therefore, it is neither dependent on the identified system nor on the output weights.

After exciting the ESN with the signals mentioned above, one at a time, the next step is to perform SVD of the resulting ESN state response snapshots and plot the energy

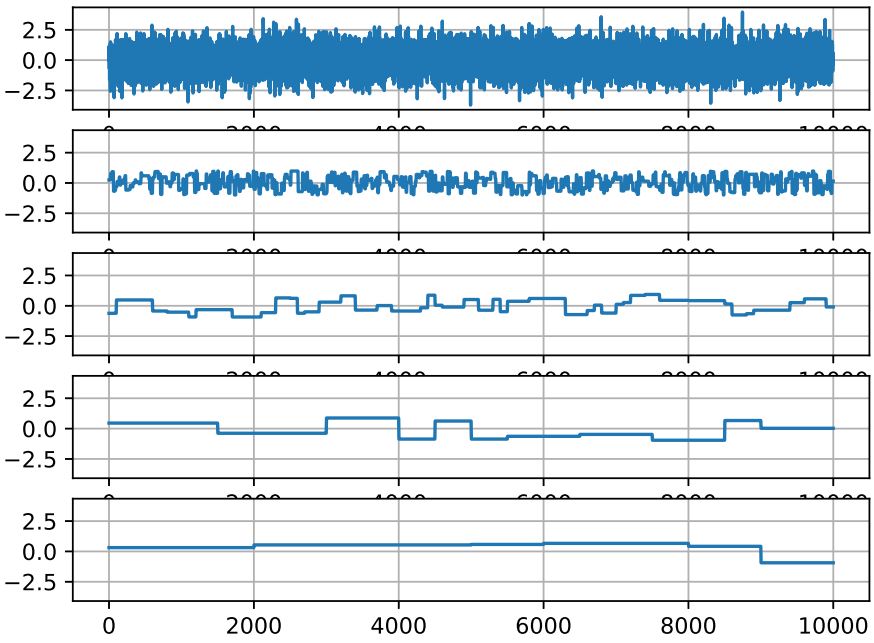


Figure 25 – One-dimensional input signals for the reservoir energy contribution distribution experiment. White noise (top), APRBS signals (usually used in identification tasks): with a minimum period of 10, 100, 500, and 1,000 timesteps, respectively, from second topmost plot to bottom.

contribution ϵ_j associated with each singular value, sorted from highest to lowest according to Eq. (130). All the reservoirs employed for this experiment are fully leaked ($\gamma = 1$), have 500 neurons, a spectral radius $\rho = 0.99$, and a value 0.1 for both input scaling and bias scaling. Since there is actually no system being identified, the parameters were chosen heuristically so that the ESN at least is stable and has echo state property, which are the requirements for this discussion to hold.

Figure 26 showcases the mean and standard deviation of the energy contribution of the 10 highest singular values for each state snapshot considering 20 randomly initialized reservoirs. The implication of this result is that the singular values become more evenly distributed the higher the frequencies of the input signal are. As the white noise is a signal with heavy high-frequency information, one expects the ESN state response to have a more even energy contribution distribution among the singular values.

Meanwhile, the lower frequency signals have the energy contribution concentrated about the highest magnitude singular value. In fact, real-life dynamic systems work as low pass filters (CHEN, C.-T., 1998) and, therefore, they are expected to have lower frequency information. The slower the system dynamics are, the larger the minimum period of an APRBS signal needs to be, which directly affects the singular value profile of the model order reduction.

This experiment implies that, since the distribution of the energy contribution depends

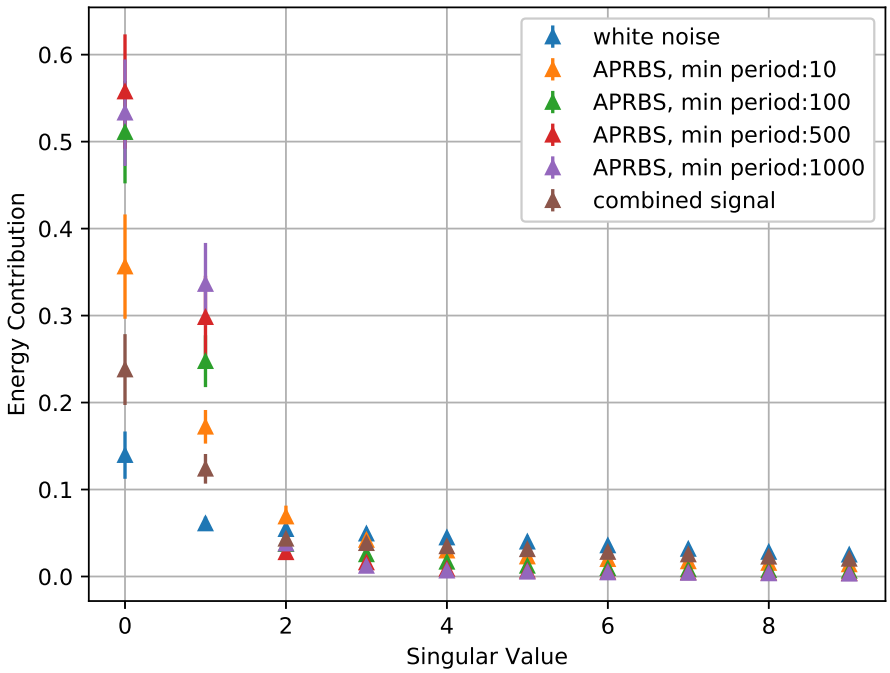


Figure 26 – Mean and Standard deviation of the first ordered 10 singular values (with 0 corresponding to the highest and 9 to the lowest) obtained from the snapshots of 20 different ESN reservoirs. Each color corresponds to a different input signal fed to the ESN reservoir, shown in Fig. 25.

entirely on the input signal frequency, the number of states pruned by MOR is higher for cases with low-frequency dynamics. After all, since the energy contribution is more concentrated on the first singular values, the number of columns pruned is higher than when the singular values are more evenly distributed (as in the case of high-frequency signals like white noise). As an easy example, the highest energy contribution singular value for the APRBS signal with a minimum period of $1{,}000$ timesteps contributes more to the total energy of the snapshots than the sum of the 10 highest singular values for the white noise shown in the plot.

4.4.2 Memory Capacity Evaluation

Short-term Memory Capacity (MC) is a well-known metric for Echo State Networks (JAEGER, 2002) that measures how well an ESN can remember past inputs and general dynamic storage capacity. MC serves as a performance measurement for ESN reservoirs which is obtained from the following procedure:

- For an arbitrary n, train a single-input, single-output Echo State Network so that the input is a given white noise $\eta[k]$, and the output is the same white noise delayed n timesteps $\eta[k-n]$. In layman's terms, the ESN is supposed to "memorize" the input from n timesteps ago.

- Obtain the correlation coefficient R_n for the training with an arbitrary n,

$$R_n = \frac{cov(y_{esn},\eta[k-n])}{var(y_{esn})var(\eta[k-n])} \tag{154}$$

  where $cov(\cdot)$ is the covariance operator, y_{esn} is the single ESN output, $var(\cdot)$ is the variance operator, and, therefore, R_n is merely the determination coefficient for a given delay n.

- The memory capacity is calculated, in theory, as:

$$MC = \sum_{n=1}^{\infty} R_n \tag{155}$$

The MC of an ESN was mathematically proven to have an upper bound in its number of neurons N (JAEGER, 2002), which means that it is directly related to the number of network neurons.

The point of this experiment is to compare the memory capacity of the reduced-order model of the ESN, and the original ESN, since the number of neurons is the upper bound for MC. Because it is impossible to run infinite training experiments, the memory capacity computation for this experiment is limited as follows:

$$MC = \sum_{n=1}^{N_{MC}} R_n \tag{156}$$

where $N_{MC} = 100$ is a sufficiently large number to measure the memory capacity of the network. As preliminary tests show, after a given n, the determination coefficient converges to a low value. Therefore, the information regarding memory capacity is more concentrated in the lower n spectrum, endorsing the limited number of experiments ($N_{MC} = 100$) for comparison purposes.

4.4.2.1 POD Reduction

The memory capacity experiment runs considering different numbers of neurons ($N = \{400,600,800,1000,1200,1400,1600,1800,2000,2200\}$) with an Energy Cutoff (EC) of 1%, 5%, and 10%. After initializing the ESN reservoir at random, MOR is applied in 12 different reservoirs for each configuration. Then, this experiment measures the mean and standard deviation for the memory capacity of these twelve runs while also obtaining the range of the reduced dimension for a given energy cutoff. This analysis allows us to measure the memory capacity drop for the model order reduction and assess how reservoir-dependent the order-reduction procedure is.

All reservoirs analyzed are fully leaked ($\gamma = 1.0$) and have input and bias scaling at 0.1. Also, the reservoir spectral radius is $\rho = 0.99$.

Figure 27 showcases the results of the Memory Capacity experiments when performing MOR at the tested ESNs given different energy cutoffs, depicting both mean and standard deviation of the 12 runs.

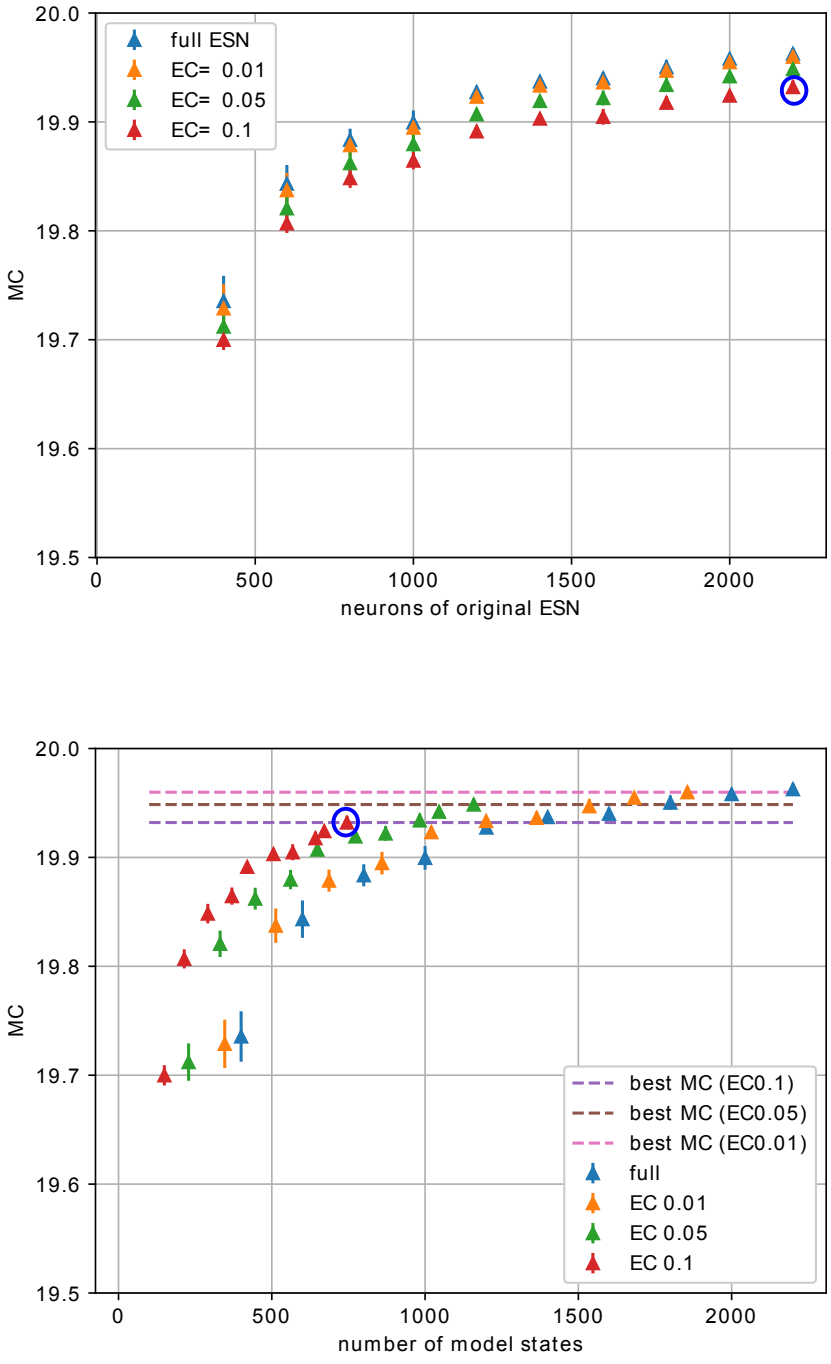


Figure 27 – Plot of the memory capacity as a function of the number of neurons of the original network (upper plot), and as a function of the number of states (lower plot). Each point is colored according to the energy cutoff of the POD-ESN that obtained the MC shown (points in blue are the MC obtained from full ESNs). EC means the energy cutoff of the applied POD.

The first plot depicts the number of ESN neurons before applying POD to a given network. It shows the expected drop in MC resulting from applying MOR with more energy cutoff.

Meanwhile, the second plot portrays the MC as a function of a given network's exact number of states after performing MOR through POD. As MC progresses monotonically, given the number of states, either in an ESN or in a given MOR of that ESN, it becomes easy to map a point of the second plot into the first one: for example, the last red point (from left to right) of both plots (marked within a blue circle) have the same memory capacity since they correspond to the same network/EC configuration. Thus, the MOR of an ESN with 2200 neurons (first plot) has roughly 750 states (second plot) at 1% energy cutoff.

As per the previous section, since this experiment traditionally employs a white noise signal, the drop in the number of reduced states is not very significant; however, the drop in MC is still small, given that a large number of states were still cut off (even in the case of 10% energy cutoff for the 2200 neuron network, the number of states was reduced to almost a third). In fact, the second plot shows that a POD-reduced network ends up being more powerful in terms of MC than a full (non-reduced) ESN with the same number of states: when one compares an ESN with a given reservoir size to a POD-reduced network from a larger ESN with the same number of states as that ESN reservoir size, the POD-reduced ESN consistently achieves a higher MC. Of course, the better performance is justifiable because a POD-reduced ESN is still more structurally complex (originated from a larger ESN) than an ESN (randomly generated) with the same number of neurons as the reduced network.

4.4.2.2 DEIM Reduction

DEIM is performed for each POD-reduced ESN to further reduce the number of $\tanh$ in the computations and evaluate the drop in MC compared to the POD-reduced ESN. Four different energy cutoff configurations were considered for the DEIM: $\{1\%,5\%,10\%,20\%\}$. This choice of four values is justified because they represent distinct magnitudes of energy cutoff, testing how the DEIM behaves on four different approximation precision requirements.

Table 12 shows the results of applying these DEIM configurations into each POD for three original reservoir sizes $N = \{800,1400,2000\}$ (from the topmost table to the bottom-most one, respectively). It presents the results for the DEIM reduction, where the memory capacity is evaluated for each configuration in energy cutoff for both POD and DEIM. The number in parenthesis is the actual dimension resulting from the reduction. Each column corresponds to a different energy cutoff configuration for DEIM, evaluated in the first row. In contrast, each row represents a different energy cutoff configuration for POD, evaluated in the first column. For instance, the MC of an ESN with a 1% energy cutoff POD (yielding $1{,}119$ states when $N = 1{,}400$) and a 5% energy cutoff DEIM (yielding 748 $\tanh$ function evaluations when $N = 1{,}400$) is 0.099, 0.03 and 0.02 for $N = 800, 1400, 2000$ respectively. Notice that there was no POD reduction for the first row of each table and no DEIM reduction for the

Table 12 – Memory capacity evaluated for different energy cutoffs used in POD and DEIM. Each table considers an original ESN with a different size N, to be reduced.

$N = 800$	Energy Cutoff (EC) for DEIM				
EC (POD)	0%	$1\%(678)$	$5\%(430)$	$10\%(279)$	$20\%(128)$
$0\%(800)$	19.88 ± 0.01	–	–	–	–
$1\%(686)$	19.87 ± 0.01	0.44 ± 0.20	0.099 ± 0.01	0.08 ± 0.04	0.54 ± 0.18
$5\%(445)$	19.86 ± 0.01	16.48 ± 2.21	0.059 ± 0.026	0.096 ± 0.02	0.55 ± 0.17
$10\%(291)$	19.84 ± 0.008	19.68 ± 0.03	0.99 ± 0.25	0.097 ± 0.02	0.54 ± 0.18

$N = 1{,}400$	Energy Cutoff (EC) for DEIM				
EC (POD)	0%	$1\%(1{,}186)$	$5\%(748)$	$10\%(484)$	$20\%(226)$
$0\%(1{,}400)$	19.93 ± 0.003	–	–	–	–
$1\%(1{,}119)$	19.93 ± 0.003	0.11 ± 0.03	0.03 ± 0.03	0.04 ± 0.02	0.17 ± 0.05
$5\%(772)$	19.91 ± 0.003	3.189 ± 1.09	0.03 ± 0.02	0.04 ± 0.02	0.17 ± 0.04
$10\%(505)$	19.90 ± 0.003	19.18 ± 0.50	0.19 ± 0.02	0.025 ± 0.02	0.17 ± 0.05

$N = 2{,}000$	Energy Cutoff (EC) for DEIM				
EC (POD)	0%	$1\%(1{,}835)$	$5\%(1{,}122)$	$10\%(713)$	$20\%(333)$
$0\%(2{,}000)$	19.96 ± 0.002	–	–	–	–
$1\%(1{,}682)$	19.95 ± 0.002	0.06 ± 0.01	0.02 ± 0.02	0.03 ± 0.01	0.03 ± 0.03
$5\%(1{,}045)$	19.94 ± 0.002	1.2 ± 0.4	0.01 ± 0.02	0.02 ± 0.02	0.08 ± 0.03
$10\%(671)$	19.92 ± 0.002	18.37 ± 0.90	0.09 ± 0.03	0.04 ± 0.01	0.07 ± 0.03

first column of each table. The empty cells indicate that DEIM can not be employed without first applying the POD reduction.

The only time DEIM achieved an MC close to the MOR was when there was a 1% energy cutoff for DEIM considering 10% energy cutoff for POD. That is, DEIM is performed for smaller reduced-order models. Regarding the experiments, performance is generally mildly better whenever DEIM has a higher number of states ratio than the POD states. For this experiment, DEIM did not perform well as expected since the white noise signal does not allow for a significant reduction of states, as it is a highly heavy information signal.

4.4.3 NARMA System

As an initial case study for the POD reduction of the ESN, this work considers a so-called NARMA (Nonlinear Autoregressive Moving Average) difference equation system (SAKEMI et al., 2020), equated as follows:

$$y[k] = 0.3y[k-1] + 0.05y[k-1]\sum_{i=1}^{m} y[k-i] + 1.5u[k-m+1]u[k] + 0.1 \tag{157}$$

where $m = 10$ is the order of the system.

As in (SAKEMI et al., 2020), the excitation signal applied in (157) is drawn from the random uniform distribution with a value range of $0 \leq u[k] \leq 0.05$. A simulation performs

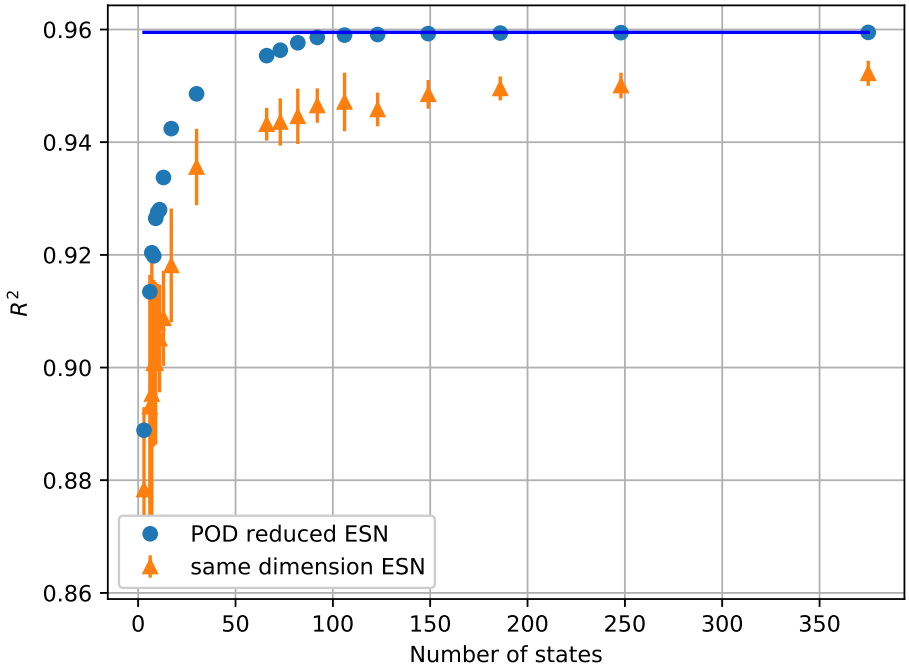


Figure 28 – Experiment comparing a POD-reduced ESN (blue dots) with an ESN of equivalent size (to the reduced ESN) (orange triangles) for the 10th-order NARMA task. The POD reduction is applied on an ESN with $1{,}400$ units in the reservoir. The horizontal axis is the number of states (units) of the reduced (full) network, while the vertical axis is the R^2 metric on the test set. The plot's blue horizontal line corresponds to the R^2 of the $1{,}400$ units ESN.

$5{,}000$ time steps where the first 2,000 samples are labeled as training data and the rest is labeled as test data. This work employs the R^2 metric to measure network performance:

$$R^2 = 1 - \frac{\sum_{k=1}^{N} e^2[k]}{(y[k] - \overline{y}[k])^2} \tag{158}$$

where $e[k]$ is the residual between the prediction model and the real system, and $\overline{y}$ is the mean of the data, while $y[k]$ is the observed data at time k.

With the dataset mentioned above, the trained ESN has the following configuration: $1{,}400$ neurons in the reservoir layer, high enough to show the MOR potential at work; a leak rate of $\gamma = 0.7$; scaling of 0.1 for both bias and input connections; and spectral radius of $\rho = 0.99$. In terms of R^2, the network had a performance of 0.95949337 for the NARMA model output. These parameters were chosen heuristically as for this study only an ESN that performs sufficiently well for the NARMA is needed, and the objective of this experiment is to assess the reduction properties of the POD in relation to the problem.

Figure 28 showcases the experiment regarding applying POD reduction so that the number of states of the POD-reduced ESN appears in the x axis (blue dots). For comparison, the plot shows the R^2 for the same NARMA experiment with 10 runs of full (non-reduced)

ESNs with the same reservoir size as the networks that underwent POD reduction (orange triangles). The POD-ESN reduction generally achieved superior performance over the full ESN at the same reservoir size, which is understandable, as the POD-reduced ESN is not only supposed to be an emulation of a larger ESN behavior but also more complex in structure. The NARMA experiment also shows that the R^2 metric for ESNs reduced to at least 50 states is very similar to the metric achieved by the original 1,400 units ESN, *i.e.*, the blue dots are very close to the horizontal blue line in the plot of Figure 28 when the number of states is higher than 50.

4.4.3.1 DEIM Stabilization

This experiment tests the stabilization solution for DEIM. The first test considers a biasless ESN, so that the solution in (WANG, H.; LONG; LIU, X.-X., 2022) can be applied.

The experiment measured the mean absolute error between the model (ESN, POD-ESN,POD-ESN with DEIM (DEIM-ESN) and a DEIM-ESN stabilized by the method above (stable DEIM-ESN)), and employed an energy cutoff for the POD at 1% (0.5% for the DEIM). Since the ESN has random initialization, the experiment was done under the same dataset for 10 runs, measuring the mean and the standard deviation of the Mean Absolute Error for each model. The result is depicted at Table 13.

Table 13 – Mean absolute error for the NARMA experiment obtained from the $3{,}000$ test time steps.

	Mean Absolute Error	St. Dev.
ESN (size=1400)	0.0497	0.0019
POD-ESN (ec=1%)	0.0636	0.00084
DEIM-ESN (dc=0.5%)	24	24
stable DEIM-ESN (dc=0.5%)	0.0887	0.0206

The method for DEIM stabilization manages to correct the inherent flaw in DEIM involving stability loss, making the resulting ESN being able to coherently identify the NARMA10. An energy cutoff of 0.5% for DEIM means that the system was interpolated around 200 points, which means a very significant reduction in computation size in relation to the POD-ESN, with small performance degradation.

Other experiments proved that even when bias is present, using the nonbiased solution as an heuristic is more effective than calculating the equilibrium point and the associated Jacobian. This is because the equilibrium point and Jacobian in the biased version depends on the control action. In both cases, one considers the control action to be 0 when applying the Jacobian, however since the Jacobian in the biased case depends on the control action, the potential for error (and thus an $\mathbf{f_p}$ Jacobian not close to zero in the interpolation) is larger.

4.4.4 Two Wells and One Riser Platform

This experiment tests the MOR over the ESN for a physical problem: an oil production platform consisting of two gas-lifted oil wells and one riser, as illustrated in Figure 15. The model for this experiment was exactly the same one used in Chapter 3.

The experiment with the two-well production platform depicts how to achieve MOR with ESNs from a system identification standpoint. First, one must train an ESN model for the two-well one-riser platform. To that end, generate $50{,}000$ timesteps of data from numerical simulation of the platform model, yielding a dataset where the 2-dimensional input to the ESN is composed of both well-production chokes $u_{ch,1}$ and $u_{ch,2}$. Further, the desired 2-dimensional output of the network corresponds to each well bottom-hole pressure: $P_{bh,1}$, $P_{bh,2}$. The training dataset consists of the first $10{,}000$ timesteps, while the segment from $k = 20{,}000$ to $k = 30{,}000$ serves as a validation set, and the rest ($k > 30{,}000$) as a test set. The described dataset is employed to train an ESN with $1{,}400$ reservoir units (chosen this high for the sake of demonstrating the MOR potential at work), a leak rate of $\gamma = 0.7$, scalings for both bias and input equal to 0.1, and spectral radius $\rho = 0.99$. In terms of R^2 metric, the network had a test performance of $(0.99881673, 0.99900379)$ for each individual well bottom-hole pressure.

Figure 29 depicts an experiment where MOR of different state sizes was tested in terms of R^2 over the test data. One can infer that, after a given number of states (150), the performance remains consistently close to the original network in terms of R^2, despite having only 10% of the original number of states.

POD reduction that resulted in 92 states also showcased good performance compared to the original network of $1{,}400$ neurons. However, with only POD, the computational problem of computing $\mathbf{T}^T\mathbf{f}$ remains. What is left is to perform DEIM on the case where the reduced network has 92 states (representing an energy cutoff of 1%). Figure 30 depicts a simulation for the ESN, POD-ESN, and POD-DEIM ESN for the test data of the two-wells and one riser platform. Even though there was a reduction from $1{,}400$ to only 92 states, the behavior of the ESN and the POD-ESN managed to be close in terms of dynamics. The application of DEIM reduced the computation nodes from $1{,}400$ to $1{,}073$; however, some overshooting emerged, which was not present in the ESN and POD-ESN. Concerning the simulation run in Figure 30, the R^2 for the normalized bottom-hole pressure of each well was: $(0.9988, 0.9990)$ for the ESN, $(0.9979, 0.9981)$ for the POD-ESN, and $(0.9873, 0.9671)$ for the DEIM-POD-ESN. There is little drop in response quality from reducing the number of states from $1{,}400$ to 92 through POD, but performing interpolation from a standard POD to a POD-DEIM framework seems to affect the response more significantly. The small drop in response quality concerning the POD-ESN is expected, as the POD was performed requesting a 1% energy cutoff. In other words, the reduced-order model is 99% close to the original ESN regarding dynamic information.

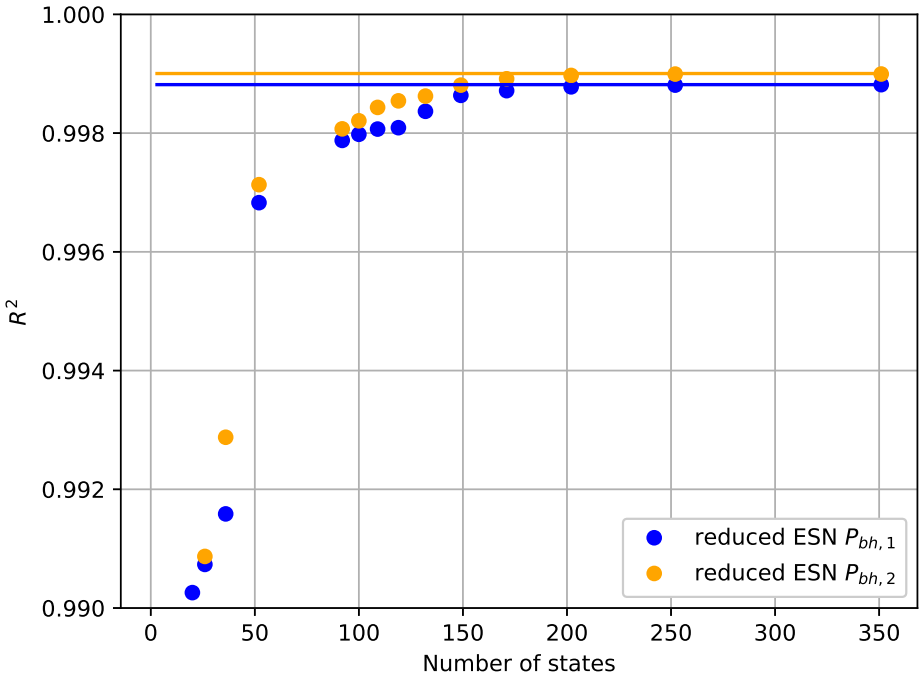


Figure 29 – POD-ESN for a system identification task. The full ESN network has $1{,}400$ neurons and was trained to model the platform with two wells and one riser. The x axis is the number of states of the reduced network, whereas the y axis is the R^2metric on the test set for each output variable (bottom-hole pressures). The bottom-hole pressure of the first well is represented in blue, while the orange color denotes the bottom-hole pressure of the second well. The R^2 of the original network corresponds to the horizontal lines of the respective colors for comparison.

4.5 DISCUSSION

POD-reduced ESN achieved a response close to the original ESN for the NARMA and the two-well one-riser case study, while it incurred a minor performance loss in the MC experiments. However, DEIM did not reach the same performance as POD in those experiments. These findings indicate that DEIM incurs more dynamic-information loss than POD, as the latter retains the number of activation functions in the reduced model even though it reduces the number of states. Thus, the capacity of a reservoir to represent a nonlinear system accurately is shown to be more influenced by the combination of the nonlinear functions in a high-dimensional space than by maintaining a high-dimensionality of the reservoir states themselves. In the context of MOR, this function combination is given by lifting the reduced states back to the original space just before applying the $\tanh$ nonlinearity.

The application of POD leads to some reduction in the memory required for storing and using the POD-reduced ESN. First, the state-to-output linear combination matrix $\mathbf{W}_\mathbf{r}^\mathbf{o}\mathbf{T}$ maps the reduced space directly to the output, invariably reducing its size. Also, the computation of the activation functions becomes slightly less expensive memory-wise because the resulting

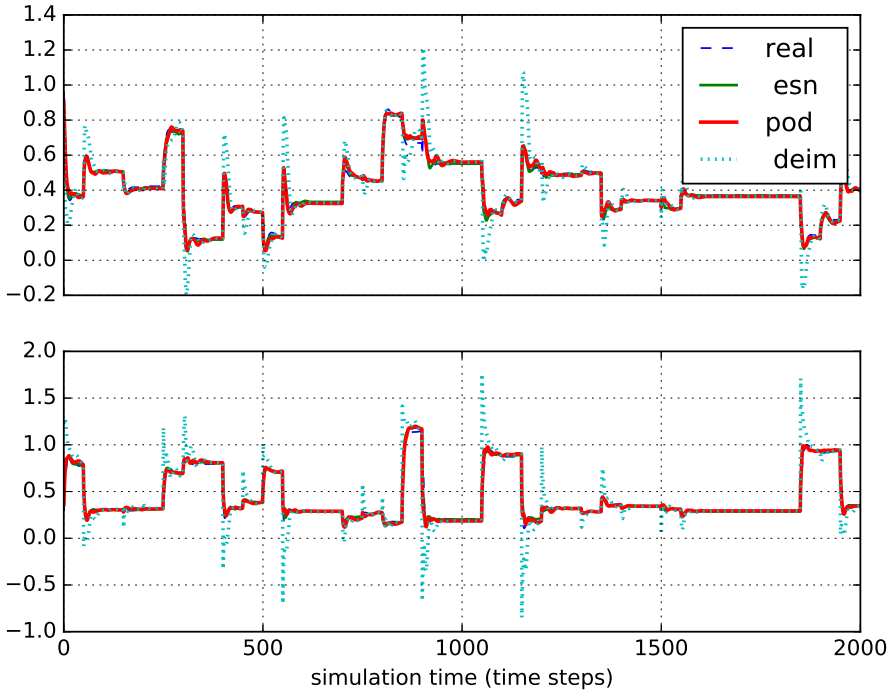


Figure 30 – Single simulation run involving a POD with 92 states (0.01 energy cutoff) and a DEIM interpolation with $m = 1{,}073$, put side by side with the original data for the bottom hole pressure p_{bh} of both wells (normalized), and the original ESN.

matrix $\mathbf{W}_{\mathbf{r}}^{\mathbf{r}}\mathbf{T}$, which is a product computed offline, has fewer elements. Of course, the resulting matrix is still large compared to an ESN with the same size as the reduction, rendering the same-size ESN less complex than the POD-reduced one.

Even though POD computes the same number of activation functions as the original ESN, the computation time is significantly reduced, as shown in Table 14. This table shows the mean time it took to execute a step in the full ESN against the time it took to perform a POD-ESN computation step for the NARMA experiment. For instance, when applying POD-ESN to reduce from 1400 states to 66 states, the results show an 80% decrease in mean execution time (from 0.767 ms to 0.147 ms) while still maintaining excellent performance, as this setup is near the horizontal line in Figure 28. All experiments were performed under similar conditions and with the same computer.

As shown in Table 14, even though there is no computation reduction in the nonlinear nodes, the computational time for a POD-ESN to compute a time step is reduced, even if by a small margin. This computation-speed gain happens precisely because the reduced-order ESN has fewer states, despite the nonlinear node computation remaining unchanged.

As previously discussed, the poor performance of DEIM in the memory capacity experiments corroborates the loss of stability incurred in the DEIM-reduced ESNs. Besides, even when the DEIM-reduced ESN dynamic system remained stable, as in the two-well experiment illustrated in Figure 30, the system experienced high overshoots translating into modeling

Table 14 – Mean execution time for the NARMA experiment composed of $5{,}000$ time steps.

	Mean Execution Time (ms)	St. Dev. (ms)
ESN (size=1400)	0.767	0.537
POD (size=3)	0.072	0.0498
POD (size=6)	0.078	0.0251
POD (size=7)	0.141	0.391
POD (size=8)	0.131	0.340
POD (size=9)	0.160	0.543
POD (size=10)	0.140	0.467
POD (size=11)	0.233	0.955
POD (size=13)	0.105	0.122
POD (size=17)	0.106	0.0738
POD (size=30)	0.135	0.0856
POD (size=66)	0.147	0.254
POD (size=73)	0.144	0.138
POD (size=82)	0.141	0.140
POD (size=92)	0.155	0.109
POD (size=106)	0.151	0.0744
POD (size=123)	0.149	0.0907
POD (size=149)	0.183	0.407
POD (size=186)	0.201	0.145
POD (size=248)	0.230	0.134
POD (size=375)	0.408	0.199

errors. The independent work (WANG, H.; LONG; LIU, X.-X., 2022) that also implements POD/DEIM on ESN, which appeared in the literature during the writing of this research, proposes a method to deal with the stability issue. However, the method is restricted to the particular class of ESNs with dynamic equations without the bias term. That method relies on expanding the nonlinear dynamics reduced by the DEIM so that the Jacobian contribution of the terms affected by $(\mathbf{P}^T\mathbf{U})$ becomes null concerning $\mathbf{u} = \mathbf{0}$. In this context, generalized methods (which account for the bias term as well) to guarantee stability retention of an ESN interpolated by DEIM are an interesting topic for future works.

4.6 SUMMARY

In this investigation, the POD achieved exceptional results in reducing the number of states of an ESN and maintaining performance. The reduced ESN performed nearly as well as the original ESN, despite the drastic reduction of states in a typical system identification task. This work also showcased how the nature of the excitation signal changes the singular value profile of the SVD, concluding that lower-frequency input signals can result in more efficient reductions. Ideally, the excitation signal should be as slow as necessary to identify a system.

However, despite performing MC tests considering signals that carry information from all frequencies, the POD-reduced network performed better than an ESN of the same size

trained on the data. Arguably, the superior performance of the POD-reduced ESN may be attributed to its ability to emulate the behavior of the larger original ESN. Additionally, the increased complexity of the reduced network, compared to an ESN of the same size, could contribute to its enhanced performance.

These findings imply that applying POD to reduce the number of states (reservoir size) of an ESN obtains a smaller model that behaves almost equivalently to the original one. However, some adaptation to the DEIM method may be necessary before it can be applied to increase model efficiency further. Also, reducing the reservoir size using POD has the advantage of interpretability since the states are sorted and pruned according to the energy contribution metric. Finally, applying POD to an ESN can show which linear combination of states contributes more significantly to the ESN dynamic behavior.

For possible future work, a suggestion is to test the developed POD-ESN model in predictive control applications, comparing the performance of the reduced-order model to its full-order counterpart. Further, there are applications in reservoir computing, such as time series prediction problems, which could benefit from a reservoir reduction using the POD-ESN. Another direction for future research is the study of ways to adapt DEIM to perform model reductions more consistently. In comparison to (JORDANOU, Jean Panaioti et al., 2023), a brief analysis on DEIM stabilization was added to this chapter, however further exploration of the DEIM stabilization properties in ESNs is a nice subject for future works on the area.

5 STABILITY ANALYSIS OF FEEDBACK CONTROL WITH ESN

This chapter concerns the theoretical development of a stability analysis method for two types of feedback control using the ESN model: a linear feedback gain and an NMPC. The methodology presented focuses on obtaining local stability conditions, as well as an estimate of the attraction region.

5.1 OVERVIEW

While it is possible to obtain a stability proof for linear MPCs (CISNEROS; WERNER, 2018), especially for the unconstrained case where the controller can be converted/reduced to a PID (CAMACHO; BORDONS, 1999), there is no systematic proof of stability for NMPC with ESN as the prediction model in LMI form considering the current knowledge. Hence, the motivation for developing an LMI-based method is to obtain such stability.

The strategy developed for this chapter is based on absolute stability, exploiting the similar behavior of the $\tanh(\cdot)$ function in an ESN and the saturation function, where certain sector conditions hold (CASTELAN, E. B.; TARBOURIECH, Sophie; QUEINNEC, 2008). Considering this similarity, one can obtain a sufficient condition for open-loop stability in ESNs. The work of (CISNEROS; WERNER, 2018) showcases a method to prove stability for constrained MPCs, where the solution of a quadratic programming problem reduces to a nonlinear operator with an output that follows a given sector condition based on the K.K.T conditions.

The core idea of the developed stability method is to combine both formulations in a similar method as (CASTELAN, E.; MORENO; PIERI, 2006) and (CASTELAN, E. B.; TARBOURIECH, Sophie; QUEINNEC, 2008), where absolute stability is proven for a system with both saturation and an unrelated nonlinearity (which in the case of this chapter, is the nonlinear operator function resulting from the NLP problem). The approach at hand is to calculate absolute stability conditions for an ESN controlled by an NMPC, which is a conservative approach; hence, it will provide a sufficient condition. However, one can systematically obtain stability regions and explicit equations from the resulting LMIs, thus obtaining a method to systematize the stability proof of such a nonlinear system and controller. Also, the first step to obtain the NMPC proof, is to demonstrate the absolute stability analysis of an ESN under feedback gain as an initial proof of concept for the stability analysis method.

5.1.1 Related Works

Discussion related to the stability properties of ESN is widely present in the literature, which is formulated as a discrete-time dynamic system. For instance, the so-called Echo State Property (JAEGER, 2001) is intimately related to the stability of a reservoir. Hence, any investigation into the ESP is directly related to the investigation of ESN stability as a dynamic

system. The initial strategy of enforcing the reservoir matrix to have eigenvalues inside the unit circle is a necessary condition (YILDIZ; JAEGER; KIEBEL, 2012). However, this condition would hold as necessary and sufficient if the activation function in the ESN was linear. The work (YILDIZ; JAEGER; KIEBEL, 2012) provides different conditions while realizing that the ones provided in (JAEGER, 2001) were insufficient to provide ESP. By declaring that the Lyapunov stability of the linear portion of the ESN is a sufficient condition for ESP, (YILDIZ; JAEGER; KIEBEL, 2012) makes a direct link between ESN stability and ESP. More recent work discusses the ESN stability itself. (BIANCHI; LIVI; ALIPPI, 2018) establishes criteria to evaluate the stability of an ESN, involving the use of tools such as recurrence analysis and the so-called Recurrence Quantification Analysis (RQA) metrics.

As for the stability in MPC, this discussion is very well established in terms of linear unconstrained MPC (CAMACHO; BORDONS, 1999). For the more general case, many proposals exist for different NMPC. One such example (SIMON; LOFBERG, 2016) expresses the stability problem of an MPC as a Mixed-Integer Linear Programming problem, where the problem is directly derived from the Lyapunov function, considering stability concepts and a stability condition for the system. However, such proof considers the system to be linear. Another work (ALAMIR, 2018) defines a method for ensuring stability in NMPC with a positive definite cost function. In MPC literature, it is normal to employ the so-called terminal constraints in the formulations. However, they tend to enhance the problem size considerably. The main idea of (ALAMIR, 2018) is to formulate a cost function with a stronger penalty further into the horizon, proving that the progressing penalty ensures stability. A more recent work (KOHLER; ZEILINGER; GRUNE, 2023) analyzes NMPC stability using Linear Programming (LP) problems through Lyapunov derived stability conditions.

In contrast to the previously mentioned works, the stability problem for this chapter is stated as an absolute stability problem, providing a stability checking method through LMIs for specific controllers considering the ESN as a model, such as a multivariable proportional gain controller and a NMPC.

5.2 ABSOLUTE STABILITY OF ECHO STATE NETWORKS

The ESN structure is very similar to a saturated linear system, as the whole right-hand side of the state equation is limited by $-1 < \tanh(\cdot) < 1$, and $\tanh(\cdot)$ is considered an approximation of the saturation function. Therefore, through algebraic manipulations, the stability of the ESN (and the PNMPC controller employing it as a model) can be verified through reformulation into an absolute stability problem. For the absolute stability problem, consider the following representation of the ESN:

$$\mathbf{x}[k+1] = (1-\gamma)\mathbf{x}[k] + \gamma(\mathbf{W_r^r}\mathbf{x}[k] + \mathbf{W_i^r}\mathbf{u}[k] + \phi(\mathbf{z}[k])) \tag{159a}$$

$$\mathbf{z}[k] = \mathbf{W_r^r}\mathbf{x}[k] + \mathbf{W_i^r}\mathbf{u}[k] \tag{159b}$$

$$\phi(\mathbf{z}[k]) = \tanh(\mathbf{z}[k]) - \mathbf{z}[k] \tag{159c}$$

or, in a more compact way to refer to this equation:

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \gamma\mathbf{W}_{\mathbf{i}}^{\mathbf{r}}\mathbf{u}[k] + \gamma\phi(\mathbf{z}[k]) \tag{160a}$$

$$\mathbf{z}[k] = \mathbf{W}_{\mathbf{r}}^{\mathbf{r}}\mathbf{x}[k] + \mathbf{W}_{\mathbf{i}}^{\mathbf{r}}\mathbf{u}[k] \tag{160b}$$

$$\phi(\mathbf{z}[k]) = \tanh(\mathbf{z}[k]) - \mathbf{z}[k] \tag{160c}$$

where $\mathbf{A} = (1-\gamma)\mathbf{I} + \gamma\mathbf{W}_{\mathbf{r}}^{\mathbf{r}}$. Do note that, although they hold the same notation, $\mathbf{z}[k]$ from this Chapter has no relation to the reduced state space, also named $\mathbf{z}$ from Chapter 4.

Based on the Lemma 1 in (TARBOURIECH, S.; PRIEUR; SILVA, 2006), for values of $\mathbf{z}$ and $\mathbf{w}$ inside the set:

$$S = \{(\mathbf{z},\mathbf{w}) \in \mathbb{R}^n \times \mathbb{R}^n : -\mathbf{1} \leq \mathbf{z} - \mathbf{w} \leq \mathbf{1}\}, \tag{161}$$

The nonlinearity $\phi(\mathbf{z})$ satisfies the following sector condition:

$$\phi(\mathbf{z})\mathbf{T}(\phi(\mathbf{z}) + \mathbf{w}) \leq 0 \tag{162}$$

for some diagonal positive definite matrix $\mathbf{T}$ with matching dimension.

In (TARBOURIECH, S.; PRIEUR; SILVA, 2006), the saturation function is considered. However, the hyperbolic tangent is not only an approximation of the saturation function but also a bounded and odd function. As with the case in (TARBOURIECH, S.; PRIEUR; SILVA, 2006), to prove that this sector also holds for the hyperbolic tangent function, one can prove that $1 - \mathbf{z} + \mathbf{w} \geq 0$ and $-1 - \mathbf{z} + \mathbf{w} \leq 0$, and both $\mathbf{z}$ and $\mathbf{w}$ belong to the set S.

Now, consider an element with index i of the sector condition:

$$(\tanh(\mathbf{z}_i) - \mathbf{z}_i)\mathbf{T}_{(i,i)}(\tanh(\mathbf{z}_i) - \mathbf{z}_i + \mathbf{w}_i) \leq 0 \tag{163}$$

which creates a transcendental product inequation where either:

$$\tanh(\mathbf{z}_i) \leq \mathbf{z}_i \tag{164}$$

$$\tanh(\mathbf{z}_i) \geq \mathbf{z}_i - \mathbf{w}_i \tag{165}$$

or

$$\tanh(\mathbf{z}_i) \geq \mathbf{z}_i \tag{166}$$

$$\tanh(\mathbf{z}_i) \leq \mathbf{z}_i - \mathbf{w}_i \tag{167}$$

because $\mathbf{T}_{(i,i)} > 0$. Solving the first inequalities reveals that:

$$0 \leq \mathbf{z}_i \tag{168}$$

$$\tanh(\mathbf{z}_i) \geq \mathbf{z}_i - \mathbf{w}_i \tag{169}$$

or

$$0 \geq \mathbf{z}_i \tag{170}$$

$$\tanh(\mathbf{z}_i) \leq \mathbf{z}_i - \mathbf{w}_i \tag{171}$$

Both lower inequalities are always true inside $-1 \leq \mathbf{z}_i - \mathbf{w}_i \leq 1$, as per the bounds for the $\tanh(\cdot)$ function.

Following the suggestion of (CASTELAN, E. B.; TARBOURIECH, Sophie; QUEINNEC, 2008), defining $\mathbf{w} = \mathbf{E_1 x} + \mathbf{z}$, where $\mathbf{E_1}$ is a matrix variable, makes the set S have the following form:

$$S = \{\mathbf{x} \in \mathbb{R}^n : -\mathbf{1} \leq \mathbf{E}_1\mathbf{x} \leq \mathbf{1}\} \tag{172}$$

and also, the next step is to redefine the sector condition (162) of the absolute stability problem as:

$$\phi(\mathbf{z})^T\mathbf{T}(\phi(\mathbf{z}) + \mathbf{E_1 x}[k] + \mathbf{W_r^r x}[k] + \mathbf{W_i^r u}[k]) \leq 0 \tag{173}$$

Or, in matrix form:

$$\begin{pmatrix}\mathbf{x}[k]\\ \phi[k]\\ \mathbf{u}[k]\end{pmatrix}^T \begin{pmatrix}0 & (\mathbf{E_1} + \mathbf{W_r^r})^T\mathbf{T} & 0\\ * & 2\mathbf{T} & \mathbf{T W_i^r}\\ * & * & 0\end{pmatrix} \begin{pmatrix}\mathbf{x}[k]\\ \phi[k]\\ \mathbf{u}[k]\end{pmatrix} \leq 0 \tag{174}$$

Utilizing this absolute stability proof strategy, it is then possible to prove the ESN stability when fed back by a linear control law $\mathbf{u} = \mathbf{Ky} = \mathbf{KW_r^o x}$. In this case, the sector condition (173) becomes:

$$\phi(\mathbf{z})^T\mathbf{T}(\phi(\mathbf{z}) + (\mathbf{E_1} + \mathbf{W_r^r} + \mathbf{W_i^r K W_r^o})\mathbf{x}[k]) \leq 0 \tag{175}$$

In parallel, considering the Lyapunov Function:

$$V(\mathbf{x}) = \mathbf{x}^T\mathbf{P}\mathbf{x} \tag{176}$$

The stability of a linear system treating ϕ as an exogenous input is:

$$V(\mathbf{x}[k+1]) - V(\mathbf{x}[k]) < 0 \tag{177a}$$

$$\begin{pmatrix}\mathbf{x}[k]\\ \phi[k]\end{pmatrix}^T \begin{pmatrix}\mathbf{A_f}^T\mathbf{P}\mathbf{A_f} & \gamma\mathbf{A_f}^T\mathbf{P}\\ * & \gamma^2\mathbf{P}\end{pmatrix} \begin{pmatrix}\mathbf{x}[k]\\ \phi[k]\end{pmatrix} - \begin{pmatrix}\mathbf{x}[k]\\ \phi[k]\end{pmatrix}^T \begin{pmatrix}\mathbf{P} & \mathbf{0}\\ * & \mathbf{0}\end{pmatrix} \begin{pmatrix}\mathbf{x}[k]\\ \phi[k]\end{pmatrix} < 0 \tag{177b}$$

which translates directly to the following LMI, given the ESN in absolute stability-proof form:

$$\begin{pmatrix}\mathbf{A_f}^T\mathbf{P}\mathbf{A_f} - \mathbf{P} & \gamma\mathbf{A_f}^T\mathbf{P}\\ * & \gamma^2\mathbf{P}\end{pmatrix} < 0 \tag{178}$$

where $\mathbf{A_f} = (\mathbf{A} + \gamma\mathbf{W_i^r K W_r^o})$, Joining the stability condition with the sector condition leads to the following matrix:

$$\begin{pmatrix}\mathbf{A_f}^T\mathbf{P}\mathbf{A_f} - \mathbf{P} & \gamma\mathbf{A_f}^T\mathbf{P} - (\mathbf{E_1} + \mathbf{W_r^r} + \mathbf{W_i^r K W_r^o})^T\mathbf{T}\\ * & \gamma^2\mathbf{P} - 2\mathbf{T}\end{pmatrix} < 0 \tag{179}$$

However, the present matrix can not yet be considered an LMI. By defining $\mathbf{S} = \mathbf{T}^{-1}$ and performing the following operation:

$$\begin{pmatrix}\mathbf{I} & 0\\ 0 & \mathbf{S}\end{pmatrix} \begin{pmatrix}\mathbf{A_f}^T\mathbf{P}\mathbf{A_f} - \mathbf{P} & \gamma\mathbf{A_f}^T\mathbf{P} - (\mathbf{E_1} + \mathbf{W_r^r} + \mathbf{W_i^r K W_r^o})^T\mathbf{T}\\ * & \gamma^2\mathbf{P} - 2\mathbf{T}\end{pmatrix} \begin{pmatrix}\mathbf{I} & 0\\ 0 & \mathbf{S}\end{pmatrix} < 0 \tag{180}$$

The inequality presents itself in the following form:

$$\begin{pmatrix} \mathbf{A_f}^T\mathbf{P}\mathbf{A_f} - \mathbf{P} & \gamma\mathbf{A_f}^T\mathbf{P}\mathbf{S} - (\mathbf{E_1} + \mathbf{W_r^r} + \mathbf{W_i^r}\mathbf{K}\mathbf{W_r^o})^T \\ * & \gamma^2\mathbf{S}\mathbf{P}\mathbf{S} - 2\mathbf{S} \end{pmatrix} < 0 \tag{181}$$

where the matrix variables are $\mathbf{P}$, $\mathbf{S}$, and $\mathbf{E_1}$, and the input gain matrix $\mathbf{K}$ is given.

To convert (181) into an LMI, it is necessary to perform the Schur's Complement (CASTELAN, E. B.; TARBOURIECH, Sophie; QUEINNEC, 2008), obtaining:

$$\begin{pmatrix} \mathbf{P}^{-1} & \mathbf{A_f} & \gamma\mathbf{S} \\ * & \mathbf{P} & (\mathbf{E_1} + \mathbf{W_r^r} + \mathbf{W_i^r}\mathbf{K}\mathbf{W_r^o})^T \\ * & * & 2\mathbf{S} \end{pmatrix} > 0 \tag{182}$$

To resolve the issue with the inversion of the Lyapunov matrix, one can consider $\mathbf{V} = \mathbf{P}^{-1}$, pre and post multiplying the second row and second column by it:

$$\begin{pmatrix} \mathbf{V} & \mathbf{A_f}\mathbf{V} & \gamma\mathbf{S} \\ * & \mathbf{V} & \mathbf{V}(\mathbf{E_1} + \mathbf{W_r^r} + \mathbf{W_i^r}\mathbf{K}\mathbf{W_r^o})^T \\ * & * & 2\mathbf{S} \end{pmatrix} > 0 \tag{183}$$

Instead of $\mathbf{E_1}$, declaring a new variable $\mathbf{Z} = \mathbf{E_1}\mathbf{W}$ results in:

$$\begin{pmatrix} \mathbf{V} & \mathbf{A_f}\mathbf{V} & \gamma\mathbf{S} \\ * & \mathbf{V} & \mathbf{Z}^T + \mathbf{V}(\mathbf{W_r^r} + \mathbf{W_i^r}\mathbf{K}\mathbf{W_r^o})^T \\ * & * & 2\mathbf{S} \end{pmatrix} > 0 \tag{184}$$

Finding a value for the positive definite $\mathbf{V}$, the diagonal matrix $\mathbf{S}$ and the real matrix $\mathbf{Z}$ is a sufficient condition for the ESN stability given the linear control law, alongside the ellipsoid condition, which will be further explored in Section 5.4.

Expressing the ESN in the form used in the absolute stability problem is convenient to demonstrate the stability of the ESN-NMPC, as shown in the next section.

5.3 CLOSED-LOOP STABILITY OF AN ESN-NMPC

This section describes the derivation of the absolute stability problem for an NMPC (Nonlinear MPC) using an ESN as the prediction model. The ESN model is used in its absolute stability form, which enables a QP representation of the NMPC. However, the problem must assume the dead zone non-linearity to be an exogenous input to the system. This analysis assumes that the plant is exactly equal to the model, namely, also an ESN.

The bulk of the stability proof developed here is inspired by the work of (CISNEROS; WERNER, 2018). The focus of (CISNEROS; WERNER, 2018) is to prove stability for MPCs that utilize quasi-LPV (Linear Parameter Varying) systems as the prediction model. This work employs the same MPC stabilization theory, but with the absolute stability representation of the ESN instead of an LPV model.

In terms of stability analysis, (CISNEROS; WERNER, 2018) exploits that a QP is a convex function and, as such, has only one minimum and abstracts the solution of the optimization problem as a nonlinear operator, with the linear term of the QP cost function

as the input, and the QP solution as the output. The linear term is chosen as input because it is the only term that depends (linearly) on the quasi-LPV system state. The fact that the output of the operator is a QP solution implies a sector condition based on the KKT necessary conditions that must always hold for the nonlinear operator output. As a sector condition is obtained and the QP is abstracted as a generic non-linearity, the MPC stability problem is turned into an absolute stability problem over the QP solution (TARBOURIECH, Sophie et al., 2011b).

The stability conditions for the NMPC are formulated by expressing the ESN in its absolute stability form. The result of this transformation is being able to represent the stability problem of the NMPC as a parameter-independent LMI, with the drawback of needing the linear portion of the ESN model to be open-loop stable.

Referring to the cost function of a NMPC problem, that has the following general form:

$$J(\mathbf{x}[k]) = \min_{\mathbf{U}[k]} \mathbf{X}^T[k]\hat{\mathbf{C}}^T\mathbf{Q}\hat{\mathbf{C}}\mathbf{X}^T[k] + \mathbf{U}[k]^T\mathbf{R}\mathbf{U}[k] \tag{185}$$

The result of the optimization problem, no matter the form of the constraints, must obey the geometric necessary optimality condition:

$$\nabla_{\mathbf{U}^*}\mathbf{J}(\mathbf{x}[k]) \cdot \mathbf{U}^* \leq 0 \tag{186}$$

By making the initial assumption (**naive approach**) that the absolute stability nonlinearity does not depend on the sector condition, the geometric optimality condition assumes the following quadratic form:

$$\tau(\boldsymbol{\psi}(\mathbf{c})^T\mathbf{H}\boldsymbol{\psi}(\mathbf{c}) + \mathbf{c}^T\boldsymbol{\psi}(\mathbf{c})) \leq 0 \tag{187}$$

The solution of the NLP is represented by the nonlinear operator $\boldsymbol{\psi}\mathbf{c}[k])$, with $\mathbf{c}[k]$ being the linear coefficient of geometric optimality condition. The definitions of $\mathbf{H}$ and $\mathbf{c}$ will be further elaborated upon later, as the ESN in absolute stability form will be expanded in the prediction and control horizons. The scalar $\tau > 0$ is a variable that serves the same purpose as $\mathbf{T}$ in the saturation sector condition.

In the context of closed-loop NMPC, $\phi(\mathbf{z}[k])$ is a free variable over the prediction horizon, which implies an expansion of the saturation sector condition over the horizon. Also, the reference signal $\mathbf{Y_{ref}}$ is considered 0, without loss of generality, as the unbiased ESN has an equilibrium point at $\mathbf{u} = 0$,$\mathbf{y} = 0$. Other possible configurations can be reduced to the one at hand through coordinate changes.

By defining the following variable:

$$\Phi(\mathbf{Z}[k]) = \begin{pmatrix} \phi(\mathbf{z}[k]) \\ \phi(\mathbf{z}[k+1]) \\ \vdots \\ \phi(\mathbf{z}[k+N_y-1]) \end{pmatrix} \tag{188}$$

and defining:

$$\mathbf{Z}[k] = \begin{pmatrix} \mathbf{z}[k] \\ \mathbf{z}[k+1] \\ \vdots \\ \mathbf{z}[k+N_y-1] \end{pmatrix} = \mathbf{\Lambda_C}\mathbf{x}[k] + \mathbf{G_C}\boldsymbol{\psi}[k] + \mathbf{G_D}\Phi \tag{189}$$

$$\mathbf{G_C} = \begin{pmatrix} \mathbf{W_i^r} & 0 & \dots \\ \gamma\mathbf{W_r^r}\mathbf{W_i^r} & \mathbf{W_i^r} & \dots \\ \vdots & \vdots & \dots \\ \mathbf{W_r^r}[(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^{N_y-2}\gamma\mathbf{W_i^r} & \mathbf{W_r^r}[(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^{N_y-3}\gamma\mathbf{W_i^r} & \dots \end{pmatrix} \tag{190}$$

$$\mathbf{G_D} = \begin{pmatrix} 0 & 0 & \dots \\ \gamma\mathbf{W_r^r} & 0 & \dots \\ \vdots & \vdots & \dots \\ \mathbf{W_r^r}[(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^{N_y-2}\gamma & \mathbf{W_r^r}[(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^{N_y-3}\gamma & \dots \end{pmatrix} \tag{191}$$

$$\mathbf{\Lambda_C} = \begin{pmatrix} \mathbf{W_r^r} \\ \mathbf{W_r^r}[(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}] \\ \vdots \\ \mathbf{W_r^r}[(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^{N_y-1} \end{pmatrix} \tag{192}$$

The null diagonal from $\mathbf{G_D}$ comes from the fact that no $\mathbf{z}$ depends on ϕ at its current time. Each row k of the matrices calculating $\mathbf{Z}[k]$ represents the contribution of each variable for $\mathbf{z}[k]$, where the same logic of calculating prediction in MPC is applied (CAMACHO; BORDONS, 1999).

The revised sector condition is the sum of each saturation sector condition over the prediction horizon, resulting in the expression:

$$\Phi(\mathbf{Z}[k])^T\mathbf{T}((I + \mathbf{G_D})\Phi(\mathbf{Z}[k]) + (\mathbf{E_1} + \mathbf{\Lambda_C})\mathbf{x}[k] + \mathbf{G_C}\boldsymbol{\psi}[k]) \leq 0 \tag{193}$$

where $\mathbf{E_1}$, in this case, has dimension $NN_y \times N$ to match $\mathbf{\Lambda_C}$, and $\mathbf{T} \in \mathbb{R}^{NN_y \times NN_y}$ since it composes all the saturation sectors.

Multiple saturation sets are considered to have the form (172), since it has the exact same form for every instant in the prediction horizon.

The quadratic form of the sector condition, this time considering the whole horizon, is:

$$\begin{pmatrix} \mathbf{x} \\ \Phi \\ \boldsymbol{\psi} \end{pmatrix}^T \begin{pmatrix} 0 & (\mathbf{E_1} + \mathbf{\Lambda_C})^T\mathbf{T} & 0 \\ * & \mathbf{T}\hat{\mathbf{G}}_\mathbf{D} + \hat{\mathbf{G}}_\mathbf{D}^T\mathbf{T} & \mathbf{T}\mathbf{G_C} \\ * & * & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \Phi \\ \boldsymbol{\psi} \end{pmatrix} \leq 0 \tag{194}$$

where $\hat{\mathbf{G}}_\mathbf{D} = \mathbf{I} + \mathbf{G_D}$.

The first step to defining matrix $\mathbf{H}$ and linear function $\mathbf{c}[k]$ is to state the absolute stability ESN prediction model over the horizons, given initial state $\mathbf{x}[k]$:

$$\mathbf{X}[k] = \Lambda\mathbf{x}[k] + \mathbf{G_1}\boldsymbol{\psi}(\mathbf{c}[k]) + \mathbf{G_2}\mathbf{\Phi}(\mathbf{z}[k]) \tag{195}$$

This means that the absolute stability problem considers $\mathbf{\Phi}$, the non-linearity computation along the prediction horizon, independent from the control action. The assumption is conservative because $\mathbf{z}$ clearly depends on $\mathbf{u}$. The absolute stability problem is already sufficient by definition since the nonlinearity is being abstracted as a set of nonlinearities inside a sector, but considering the nonlinearity as an independent variable all over the horizon makes the conservativeness of the absolute stability problem stronger. Matrices $\mathbf{\Lambda}$, $\mathbf{G_1}$, and $\mathbf{G_2}$ are defined as follows:

$$\mathbf{\Lambda} = \begin{pmatrix} [(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}] \\ [(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^2 \\ \vdots \\ [(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^{N_y} \end{pmatrix} \tag{196}$$

$$\mathbf{G_1} = \begin{pmatrix} \gamma\mathbf{W_i^r} & 0 & \dots \\ [(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]\gamma\mathbf{W_i^r} & \gamma\mathbf{W_i^r} & \dots \\ \vdots & \vdots & \dots \\ [(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^{N_y-1}\gamma\mathbf{W_i^r} & [(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^{N_y-2}\gamma\mathbf{W_i^r} & \dots \end{pmatrix} \tag{197}$$

$$\mathbf{G_2} = \begin{pmatrix} \gamma\mathbf{I} & 0 & \dots \\ [(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]\gamma & \gamma\mathbf{I} & \dots \\ \vdots & \vdots & \dots \\ [(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^{N_y-1}\gamma & [(1-\gamma)\mathbf{I} + \gamma\mathbf{W_r^r}]^{N_y-2}\gamma & \dots \end{pmatrix} \tag{198}$$

To finally obtain the sector condition associated with the NMPC, replace $\mathbf{X}[k]$ with its definition in the cost function (185), and derive the cost function with respect to the decision variable, according to the "naive approach" axiom (sector nonlinearity does not depend on decision variable). Then, the geometric optimality conditions present themselves in the quadratic form (187), with $\mathbf{H}$ and $\mathbf{c}[k]$ defined as:

$$\mathbf{H} = \mathbf{G_1}^T\widehat{\mathbf{C}}^T\mathbf{Q}\widehat{\mathbf{C}}\mathbf{G_1} + \mathbf{R} \tag{199}$$

$$\mathbf{c}[k] = \mathbf{C_z}\mathbf{x}[k] + \mathbf{D_z}\mathbf{\Phi}(\mathbf{Z}[k]) \tag{200}$$

$$\mathbf{C_z} = \mathbf{G_1}^T\widehat{\mathbf{C}}^T\mathbf{Q}\widehat{\mathbf{C}}\Lambda \tag{201}$$

$$\mathbf{D_z} = \mathbf{G_1}^T\widehat{\mathbf{C}}^T\mathbf{Q}\widehat{\mathbf{C}}\mathbf{G_2} \tag{202}$$

where $\widehat{\mathbf{C}} = \mathbf{I}_{N_y} \otimes \mathbf{C}$ and $\otimes$ is the Kronecker product. The matrix $\mathbf{H}$ is from the quadratic term of the sector condition, and $\mathbf{c}[k]$ is the linear term.

The representation of the sector condition in terms of $\mathbf{x}$ and $\boldsymbol{\phi}$ then becomes:

$$\begin{pmatrix} \mathbf{x}[k] \\ \mathbf{\Phi}(\mathbf{Z}[k]) \\ \boldsymbol{\psi}(\mathbf{c}) \end{pmatrix}^T \begin{pmatrix} 0 & 0 & \tau\mathbf{C_z}^T \\ * & 0 & \tau\mathbf{D_z}^T \\ * & * & 2\tau\mathbf{H} \end{pmatrix} \begin{pmatrix} \mathbf{x}[k] \\ \mathbf{\Phi}(\mathbf{Z}[k]) \\ \boldsymbol{\psi}(\mathbf{c}) \end{pmatrix} \leq 0 \tag{203}$$

which requires the center matrix of the condition to be negative semi-definite. Joining with the saturation-derived condition over the horizon, the full sector condition requirements of the system are that the following matrix is negative semi-definite:

$$\begin{pmatrix} 0 & (\mathbf{E_1} + \mathbf{\Lambda_C})^T\mathbf{T} & \tau\mathbf{C_z}^T \\ * & \mathbf{T}\hat{\mathbf{G}}_\mathbf{D} + \hat{\mathbf{G}}_\mathbf{D}^T\mathbf{T} & \tau\mathbf{D_z}^T + \mathbf{T}\mathbf{G_C} \\ * & * & 2\mathbf{H} \end{pmatrix} \leq 0 \tag{204}$$

The full system to calculate the absolute stability is represented as:

$$\mathbf{x}[k+1] = (1-\gamma)\mathbf{x}[k] + \gamma(\mathbf{W_r^r}\mathbf{x}[k] + \mathbf{W_i^r}\mathbf{K_c}\boldsymbol{\psi}(\mathbf{c}[k]) + \mathbf{K_y}\Phi(\mathbf{z}[k])) \tag{205a}$$

$$\mathbf{Z}[k] = \mathbf{\Lambda_C}\mathbf{x}[k] + \mathbf{G_C}\boldsymbol{\psi}(\mathbf{c}[k]) + \mathbf{G_D}\Phi \tag{205b}$$

$$\mathbf{c}[k] = \mathbf{C_z}\mathbf{x}[k] + \mathbf{D_z}\mathbf{\Phi}(\mathbf{Z}[k]) \tag{205c}$$

$$\mathbf{y}[k] = \mathbf{W_r^o}\mathbf{x}[k] \tag{205d}$$

Without loss of generality, assume that the bias $\mathbf{W_b^r}$ is zero for a simplified proof. The gain $\mathbf{K_c}$ is a block matrix that originates from $\boldsymbol{\psi}$ being the solution of the optimization problem. The actual controller only submits the first result to the plant. Therefore, $\mathbf{K_c} = (\mathbf{I},\mathbf{0},\mathbf{0},\cdots,\mathbf{0})$. The matrix $\mathbf{K_y}$ is analogous to $\mathbf{K_c}$, but over the prediction horizon. Otherwise, in the point of view of the computation of $\mathbf{x}$, all ϕ are considered decoupled.

To measure the stability of a linear state space system, consider the Lyapunov function in the form of:

$$V(\mathbf{x}[k]) = \mathbf{x}[k]^T\mathbf{P}\mathbf{x}[k] \tag{206}$$

where $\mathbf{P}$ is a symmetric, positive definite matrix. For a dynamic system to be stable, the Lyapunov function must be decreasing in terms of the system state progressing over time, therefore:

$$\mathbf{x}[k+1]^T\mathbf{P}\mathbf{x}[k+1] - \mathbf{x}[k]^T\mathbf{P}\mathbf{x}[k] < 0 \tag{207}$$

It is important to note that the state space system being analyzed represents the actual system, represented by an ESN, that receives the input calculated by the NMPC, which means that $\mathbf{P}$ has dimension $\mathbb{R}^{N\times N}$, with N being the number of ESN neurons.

Replacing $\mathbf{x}[k+1]$ with the absolute stability ESN, results in the following matrix, which must be negative definite:

$$\begin{pmatrix} \mathbf{A}^T\mathbf{P}\mathbf{A} - \mathbf{P} & \gamma\mathbf{A}^T\mathbf{P}\mathbf{K_y} & \gamma\mathbf{A}^T\mathbf{P}\mathbf{W_i^r}\mathbf{K_c} \\ * & \gamma^2{\mathbf{K_y}}^T\mathbf{P}\mathbf{K_y} & \gamma^2{\mathbf{K_y}}^T\mathbf{P}\mathbf{W_i^r}\mathbf{K_c} \\ * & * & \gamma^2{\mathbf{K_c}}^T{\mathbf{W_i^r}}^T\mathbf{P}\mathbf{W_i^r}\mathbf{K_c} \end{pmatrix} < 0 \tag{208}$$

Since both LMI (208) and LMI (204) must hold in an AND condition, the former subtracts the latter to form:

$$\begin{pmatrix} \mathbf{A}^T\mathbf{P}\mathbf{A} - \mathbf{P} & \gamma\mathbf{A}^T\mathbf{P}\mathbf{K_y} - (\mathbf{E_1} + \Lambda_C)^T\mathbf{T} & \gamma\mathbf{A}^T\mathbf{P}\mathbf{W_i^r}\mathbf{K_c} - \tau{\mathbf{C_z}}^T \\ * & \gamma^2{\mathbf{K_y}}^T\mathbf{P}\mathbf{K_y} - \mathbf{T}\hat{\mathbf{G}}_\mathbf{D} - \hat{\mathbf{G}}_\mathbf{D}^T\mathbf{T} & \gamma^2{\mathbf{K_y}}^T\mathbf{P}\mathbf{W_i^r}\mathbf{K_c} - \tau{\mathbf{D_z}}^T - \mathbf{T}\mathbf{G_C} \\ * & * & \gamma^2{\mathbf{K_c}}^T{\mathbf{W_i^r}}^T\mathbf{P}\mathbf{W_i^r}\mathbf{K_c} - 2\tau\mathbf{H} \end{pmatrix} < 0 \tag{209}$$

With the effect of the $\mathbf{S} = \mathbf{T}^{-1}$ operation performed for the linear feedback case, the matrix becomes:

$$\begin{pmatrix} \mathbf{A}^T\mathbf{P}\mathbf{A} - \mathbf{P} & \gamma\mathbf{A}^T\mathbf{P}\mathbf{K_y}\mathbf{S} - (\mathbf{E_1} + \Lambda_C)^T & \gamma\mathbf{A}^T\mathbf{P}\mathbf{W_i^r}\mathbf{K_c} - {\mathbf{C_z}}^T \\ * & \gamma^2\mathbf{S}{\mathbf{K_y}}^T\mathbf{P}\mathbf{K_y}\mathbf{S} - \hat{\mathbf{G}}_\mathbf{D}\mathbf{S} - \mathbf{S}\hat{\mathbf{G}}_\mathbf{D}^T & \gamma^2\mathbf{S}{\mathbf{K_y}}^T\mathbf{P}\mathbf{W_i^r}\mathbf{K_c} - \tau\mathbf{S}{\mathbf{D_z}}^T - \mathbf{G_C} \\ * & * & \gamma^2{\mathbf{K_c}}^T{\mathbf{W_i^r}}^T\mathbf{P}\mathbf{W_i^r}\mathbf{K_c} - 2\tau\mathbf{H} \end{pmatrix} < 0 \tag{210}$$

Performing the Schur's Complement, in the same vein of the $\mathbf{u} = \mathbf{Kx}$ case, results in:

$$\begin{pmatrix} \mathbf{P}^{-1} & \mathbf{A} & \gamma\mathbf{K_y S} & \gamma\mathbf{W_i^r K_c} \\ * & \mathbf{P} & (\mathbf{E_1} + \mathbf{\Lambda_C})^T & \tau\mathbf{C_z}^T \\ * & * & \hat{\mathbf{G}}_\mathbf{D}\mathbf{S} + \mathbf{S}\hat{\mathbf{G}}_\mathbf{D}^T & \mathbf{S}\tau\mathbf{D_z}^T + \mathbf{G_C} \\ * & * & * & 2\tau\mathbf{H} \end{pmatrix} > 0 \tag{211}$$

Then, by defining $\mathbf{V} = \mathbf{P}^{-1}$, and pre multiplying the second block row and post multiplying the second block column of the matrix:

$$\begin{pmatrix} \mathbf{V} & \mathbf{AV} & \gamma\mathbf{K_y S} & \gamma\mathbf{W_i^r K_c} \\ * & \mathbf{V} & \mathbf{V}(\mathbf{E_1} + \mathbf{\Lambda_C})^T & \tau\mathbf{V C_z}^T \\ * & * & \hat{\mathbf{G}}_\mathbf{D}\mathbf{S} + \mathbf{S}\hat{\mathbf{G}}_\mathbf{D}^T & \tau\mathbf{S D_z}^T + \mathbf{G_C} \\ * & * & * & 2\tau\mathbf{H} \end{pmatrix} > 0 \tag{212}$$

The next step is to define $\mathbf{Z_e} = \mathbf{E_1 V}$. In this case, $\mathbf{Z_e} \in \mathbb{R}^{N_y N \times N}$, therefore:

$$\begin{pmatrix} \mathbf{V} & \mathbf{AV} & \gamma\mathbf{K_y S} & \gamma\mathbf{W_i^r K_c} \\ * & \mathbf{V} & \mathbf{Z_e}^T + \mathbf{V}\mathbf{\Lambda}_\mathbf{C}^T & \tau\mathbf{V C_z}^T \\ * & * & \hat{\mathbf{G}}_\mathbf{D}\mathbf{S} + \mathbf{S}\hat{\mathbf{G}}_\mathbf{D}^T & \tau\mathbf{S D_z}^T + \mathbf{G_C} \\ * & * & * & 2\tau\mathbf{H} \end{pmatrix} > 0 \tag{213}$$

To get rid of τ multiplying both $\mathbf{S}$ and $\mathbf{V}$, pre and post multiply the fourth row and column by $\alpha = \tau^{-1}\mathbf{I}$, finally obtaining the LMI:

$$\begin{pmatrix} \mathbf{V} & \mathbf{AV} & \gamma\mathbf{K_y S} & \gamma\mathbf{W_i^r K_c}\alpha \\ * & \mathbf{V} & \mathbf{Z_e}^T + \mathbf{V}\mathbf{\Lambda}_\mathbf{C}^T & \mathbf{V C_z}^T \\ * & * & \hat{\mathbf{G}}_\mathbf{D}\mathbf{S} + \mathbf{S}\hat{\mathbf{G}}_\mathbf{D}^T & \mathbf{S D_z}^T + \alpha\mathbf{G_C} \\ * & * & * & 2\alpha\mathbf{H} \end{pmatrix} > 0 \tag{214}$$

Finding a positive definite $\mathbf{V}$, a positive diagonal $\mathbf{S}$, a positive α and a real $\mathbf{Z}$ that solves LMI (214) is a sufficient condition of the ESN-NMPC stability.

5.4 ELLIPSOID OF ATTRACTION

As with saturation systems (CASTELAN, E. B.; TARBOURIECH, Sophie; QUEINNEC, 2008), there is a method to maximize the estimate of the region of attraction of a given nonlinear system under saturation (in this case, $\tanh$) constraints.

Given that the ESN is subject to an absolute stability analysis of a system with behavior akin to a saturation nonlinearily, the ellipsoid defined by the Lyapunov matrix $\mathbf{P}$ serves as a region of attraction estimate:

$$\mathbf{x}^T\mathbf{P}\mathbf{x} \leq \gamma_2^{-1} \tag{215}$$

where γ_2 refers to the ellipsoid size, must be contained in the set defined by (172), which can also be represented by the following inequation, in the $\mathbf{u} = \mathbf{Kx}$ case:

$$|\mathbf{E_1 x}| \leq \mathbf{1} \tag{216}$$

Applying the squared norm on both sides implies the following inequality:

$$\mathbf{x}^T {\mathbf{E_1}}^T \mathbf{E_1} \mathbf{x} \leq N \tag{217}$$

where N, the number of neurons in the ESN, is the total number of times the nonlinearity appears in this formulation. Now, both sides are scalars. All the elements of the system have the same upper and lower bound in $(-1,1)$ and are repeated N times, which is this ellipsoid representation considered for the saturation set.

That (215) is contained in (217) means that every $\mathbf{x}$ that solves (215) must also solve (217). This means that, given $\mathbf{P}$ as a symmetric positive definite matrix variable, the following condition must hold $\forall \mathbf{x}$:

$$1 \geq \gamma_2 \mathbf{x}^T \mathbf{P} \mathbf{x} \geq N^{-1} \mathbf{x}^T {\mathbf{E_1}}^T \mathbf{E_1} \mathbf{x} \tag{218}$$

Which has as sufficient condition that the following matrix must be negative definite:

$$\frac{1}{N\gamma_2} {\mathbf{E_1}}^T \mathbf{E_1} - \mathbf{P} \leq 0 \tag{219}$$

Executing the Schur Complement on (219) results in:

$$\begin{pmatrix} \mathbf{P} & \mathbf{E_1} \\ {\mathbf{E_1}}^T & \gamma_2 N \mathbf{I} \end{pmatrix} \geq 0 \tag{220}$$

Since $\mathbf{Z_e} = \mathbf{E_1} \mathbf{V}$, pre and post multiplying the first row and column by $\mathbf{V}$ results in:

$$\begin{pmatrix} \mathbf{V} & \mathbf{Z_e} \\ {\mathbf{Z_e}}^T & \gamma_2 N \mathbf{I} \end{pmatrix} \geq 0 \tag{221}$$

Both LMI (214) and LMI (221) are then solved as a system of LMI to obtain a stability proof for the ESN-NMPC.

Do note that in the control gain case, $\mathbf{Z_e} \in \mathbb{R}^{N \times N}$, and in the NMPC case, $\mathbf{Z_e} \in \mathbb{R}^{N N_y \times N}$. Also, for the NMPC, the LMI is as follows:

$$\begin{pmatrix} \mathbf{V} & \mathbf{Z_e} \\ {\mathbf{Z_e}}^T & \gamma_2 N N_y \mathbf{I} \end{pmatrix} \geq 0 \tag{222}$$

Also, the region of attraction defined by the ellipsoid (215) is translated directly into the variables $\mathbf{P} = \mathbf{V}^{-1}$ and γ_2 obtained by the system of LMIs.

A possible objective function for absolute stability LMI/Semidefinite Programming Problems is as follows:

$$\min_{\mathbf{P},\mathbf{T},\gamma_2} \quad \beta_0 \gamma_2 + \beta_1 trace(\mathbf{P}) \tag{223}$$

where β_0,β_1 are parameters set by the user.

The interpretation for this objective function is that, by minimizing the trace of $\mathbf{P}$, the result maximizes the principal axes of the ellipsoid. However, since the problem is being defined in terms of $\mathbf{W}$,$\mathbf{S}$,$\mathbf{Z}$,γ_2, some variable changes must be made to avoid an inverted matrix in the

cost function. Instead of minimizing $\mathbf{V}^{-1}$, minimizing the trace of auxiliary positive definite variable $\mathbf{M_w}$ has the same effect (CASTELAN, E. B.; TARBOURIECH, Sophie; QUEINNEC, 2008), as long as the following LMI holds:

$$\begin{pmatrix} \mathbf{M_w} & \mathbf{I} \\ * & \mathbf{V} \end{pmatrix} > 0 \tag{224}$$

Thus, the full SDP optimization problem becomes:

$$\begin{aligned} \min_{\mathbf{V},\mathbf{S},\gamma_2,\mathbf{Z},\mathbf{M_w}} \quad & \beta_0\gamma + \beta_1 trace(\mathbf{M_w}) \\ \text{s.t.} \quad & (214), (221) \text{ and } (224) \\ & \mathbf{W},\mathbf{M_w} \geq 0 \\ & \mathbf{S} \geq 0 \text{ and diagonal} \end{aligned} \tag{225}$$

To recover the original variables, one just needs to perform:

$$\mathbf{P} = \mathbf{V}^{-1} \tag{226}$$

$$\mathbf{E_1} = \mathbf{Z}\mathbf{V}^{-1} \tag{227}$$

$$\mathbf{T} = \mathbf{S}^{-1} \tag{228}$$

5.5 NUMERICAL EXAMPLE: TANK SYSTEM

5.5.1 Implementation

The optimization problems were implemented using the PICOS Python library, with Mosek serving as the SDP problem solver.

5.5.2 Case Study Description

To exemplify the stability analysis method, this proof of concept experiment presents the following simple tank system:

$$\dot{h} = \frac{k_{in}u - a\sqrt{2gh}}{A} \tag{229}$$

The cylinder tank with area $A = 28 \times 10^{-4}$ has water being pumped from an external pump source with Voltage u (the control action) and gain $k_{in} = 3.14 \times 10^{-6}$. An orifice at the bottom of the tank with area $a = 0.071 \times 10^{-4}$ provides an outflow through gravity ($g = 9.81$). The only state variable for this system is the tank level h, and the input variable is u. For training, feature scaling was performed to scale the dataset from $(0.02,32)$ for h and $(0.1,5.0)$ for u to $(0,1)$ in the ESN point of view.

5.5.3 Identification and Linear Gain Stability Test

The ESN uses the following parameter configuration $N = 5$, $\gamma = 0.1$, $\rho = 0.99$, $f_i^r = f_b^r = 0.1$ to identify the ESN. Since the tank is a simple system, there is only a need

for 5 neurons for the ESN to copy the tank behavior with relative accuracy. To generate the dataset, the simulation applied an input signal of 50000 points, which is sufficiently high to obtain an accurate model. From the simulation result, the first 40000 data points trained the ESN, and the remaining 10000 were employed for testing, obtaining a testing error of 0.05, which is sufficient to consider the ESN as a valid model for the tank.

Since the unbiased ESN has an equilibrium of $\mathbf{x} = 0$ for a control action of the type $\mathbf{u} = \mathbf{K}\mathbf{x}$, the actual equilibrium point where stability is being tested is at $h = 0.02$. For this proof, the plant itself is irrelevant, as the stability analysis is performed only on the ESN model. The control gain considered was $\mathbf{K} = -0.01$

Solving Problem (225) reveals the following stability ellipsoid:

$$\mathbf{x}^T \gamma_2 \mathbf{P} \mathbf{x} \leq 1 \tag{230}$$

where

$$\gamma_2 = 3.51 \times 10^{-4} \tag{231}$$

$$\mathbf{P} = \mathbf{W}^{-1} = \begin{pmatrix} 5.87 & 2.22 & 6.41 & -1.73 & 1.09 \\ 2.22 & 5.42 & -4.36 & -1.83 & 0.55 \\ 6.41 & -4.36 & 22.35 & 1.130 & 1.70 \\ -1.73 & -1.83 & 1.13 & 1.82 & -0.0845 \\ 1.09 & 0.551 & 1.71 & -0.0845 & 0.731 \end{pmatrix} \times 10^{-5} \tag{232}$$

Since a positive definite matrix was found, this proves that the ESN is stable under any initial condition inside the given ellipsoid.

5.5.4 ESN-NMPC Stability Test

Now, the NMPC considers the following paramenters: $\mathbf{Q} = \mathbf{R} = 1$, $\mathbf{N_y} = 4$, and $\mathbf{N_u} = 2$.

Solving the LMI for the NMPC case results in:

$$\mathbf{x}^T \gamma_2 \mathbf{P} \mathbf{x} \leq 1 \tag{233}$$

where

$$\gamma_2 = 3.64 \tag{234}$$

$$\mathbf{P} = \mathbf{W}^{-1} = \begin{pmatrix} 0.656 & -0.268 & -0.277 & -1.24 & -0.0899 \\ -0.268 & 0.158 & 0.0998 & 0.527 & 0.0327 \\ -0.277 & 0.0999 & 0.340 & 0.642 & 0.0694 \\ -1.24 & 0.527 & 0.642 & 2.444 & 0.199 \\ -0.0899 & 0.0327 & 0.0694 & 0.199 & 0.0383 \end{pmatrix} \tag{235}$$

Since an NMPC controller is less robust than a linear gain controller, it is expected that the stability ellipsoid is smaller. Since a positive definite $\mathbf{P}$ was found, it serves as mathematical

proof that the system is stable. However, one thing to remember is that the version of the NMPC being analyzed is more conservative than the real NMPC, hence why this estimation serves only as a sufficient condition.

5.5.5 Summary

This chapter showed the development of stability proofs related to two different ESN controllers. A linear gain regulatory controller and an NMPC controller. Both were successful in proving stability conditions. In the case of the NMPC LMI, since the proof is more conservative than the controller itself, it failed to converge for control horizons over 2 and prediction horizons over 4. As long as the control horizon was 1, any prediction horizon value could be applied. This may be due to the fact that the independence assumption for the controller is too conservative. However, further investigation is needed, which is an interesting proposal for future work. The method to prove stability for the PNMPC is also more elaborate because of the free and forced response division, which also calls for future work in proving stability for the ESN-PNMPC.

6 CONCLUSION

This dissertation showcased different contributions to the employment of ESN as a model in the context of control engineering. The ESN-PNMPC was developed and further analyzed in different applications while also being compared with an NMPC for the ESP-lifted oil well. This comparison was relevant by showing how, while no constraints are being violated, both controllers have a close behavior, with the computation performed by the PNMPC being less complex and faster. Also, a method for recursively computing the PNMPC Jacobian was established, a convergence proof for the prediction filter was given, and the ESN-PNMPC performed well in the case studies when compared to different types of controllers. While the number of states in an ESN inherently dominates the number of inputs and outputs, the use of POD and DEIM to provide a reduced order approximation of the ESN managed to mitigate the problem by considerably reducing the number of model states while providing nearly-similar behavior. Such reduction has implications for any situation where the ESN is used as a model for optimization, optimal control, and MPC. The proposed absolute stability proof for the ESN with the linear feedback gain and the NMPC might be useful, enhancing ESN interpretability and providing a systematic way to ensure stability for such feedback systems numerically and linearly, without any symbolic manipulation or non-linear programming problem-solving. All in all, the developed methods can contribute to the ESN use as a model for control, with further research to better investigate the methods.

6.1 LIMITATIONS AND FUTURE WORK

The developed methods still have limitations that should be addressed in future work. While the PNMPC is already a well-established MPC method, the reduced order modeling might need further testing in the context of PNMPC. For instance, how reducing the order of the model through POD (and DEIM) affects the controller's performance. Also, how does the reduction affect the computational time, which might be useful when time is a limited resource (for instance, in robotics applications)? It was shown how successful the reduced-order modeling is in addressing the issue of the large number of states inherent in reservoir computing. As mentioned in Chapter 5, not only does the proof for the stability of PNMPC need further elaboration (preliminary formulations showed that other types of sector conditions might need to be stated), but improvements could also be pursued so that the SDP can converge using larger control horizons. If the problem is actually in the assumption, further investigation is needed to find less conservative assumptions that retain the NMPC stability proof as an LMI. However, assuming dependence between the deadzone nonlinearity over the whole control horizon, naturally violates the linearity of the problem. Finding such a way to relax the conservativeness of the stability condition might be a challenge.

REFERENCES

ALAMIR, Mazen. Stability proof for nonlinear MPC design using monotonically increasing weighting profiles without terminal constraints. **Automatica**, Elsevier BV, v. 87, p. 455–459, Jan. 2018.

AN, Jinwon; CHO, Sungzoon. **Variational Autoencoder based Anomaly Detection using Reconstruction Probability**. [S.l.], 2015.

ANDERSSON, Joel A E; GILLIS, Joris; HORN, Greg; RAWLINGS, James B; DIEHL, Moritz. CasADi – A software framework for nonlinear optimization and optimal control. **Mathematical Programming Computation**, v. 11, n. 1, p. 1–36, 2019.

ANTONELO, Eric A; 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, 2015.

ANTONELO, Eric A.; CAMPONOGARA, Eduardo; FOSS, Bjarne. Echo State Networks for Data-driven Downhole Pressure Estimation in Gas-lift Oil Wells. **Neural Networks**, v. 85, p. 106–117, 2017.

ARMENIO, L. B.; TERZI, E.; FARINA, M.; SCATTOLINI, R. Model Predictive Control Design for Dynamical Systems Learned by Echo State Networks. **IEEE Control Systems Letters**, v. 3, n. 4, p. 1044–1049, Oct. 2019. ISSN 2475-1456.

AYALA, Helon Vicente Hultmann; HABINEZA, Didace; RAKOTONDRABE, Micky; SANTOS COELHO, Leandro dos. Nonlinear black-box system identification through coevolutionary algorithms and radial basis function artificial neural networks. **Applied Soft Computing**, Elsevier, v. 87, p. 105990, 2020.

BIANCHI, Filippo Maria; LIVI, Lorenzo; ALIPPI, Cesare. Investigating Echo-State Networks Dynamics by Means of Recurrence Analysis. **IEEE Transactions on Neural Networks and Learning Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 29, n. 2, p. 427–439, Feb. 2018.

BINDER, B. J. T.; KUFOALOR, D. K. M.; PAVLOV, A.; JOHANSEN, T. A. Embedded Model Predictive Control for an Electric Submersible Pump on a Programmable Logic Controller. In: IEEE Conference on Control Applications (CCA). [S.l.]: IEEE, 2014. P. 579–585.

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.

BINO, Gianfranco; BASU, Shantanu; DEY, Ramit; AUDDY, Sayantan; MULLER, Lyle; VOROBYOV, Eduard I. Predicting Stellar Mass Accretion: An Optimized Echo State Network Approach in Time Series Modeling. **The Open Journal of Astrophysics**, The Open Journal, v. 6, Apr. 2023. ISSN 2565-6120.

BISHOP, Christopher M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. New York: Springer-Verlag Inc., 2006.

BRANDÃO, Adriano Silva Martins; LIMA, Daniel Martins;
COSTA FILHO, Marcus Vinicius Americano da; NORMEY-RICO, Julio Elias. A COMPARATIVE STUDY ON EMBEDDED MPC FOR INDUSTRIAL PROCESSES. In: ANAIS do XXII Congresso Brasileiro de Automática. [S.l.: s.n.], 2018.

CAMACHO, Eduardo; BORDONS, Carlos. **Model Predictive Control**. London, UK: Springer, 1999.

CASTELAN, E.B.; MORENO, U.F.; PIERI, E.R. de. Absolute stabilization of discrete-time systems with a sector bounded nonlinearity under control saturations. In: 2006 IEEE International Symposium on Circuits and Systems. [S.l.: s.n.], 2006. 4 pp.-.

CASTELAN, Eugênio B.; TARBOURIECH, Sophie; QUEINNEC, Isabelle. Control design for a class of nonlinear continuous-time systems. **Automatica**, Elsevier BV, v. 44, n. 8, p. 2034–2039, Aug. 2008.

CHATURANTABUT, Saifon; SORENSEN, Danny C. Nonlinear Model Reduction via Discrete Empirical Interpolation. **SIAM Journal on Scientific Computing**, Society for Industrial & Applied Mathematics (SIAM), v. 32, n. 5, p. 2737–2764, Jan. 2010.

CHEN, Chao; LIU, Hui. Dynamic ensemble wind speed prediction model based on hybrid deep reinforcement learning. **Advanced Engineering Informatics**, v. 48, p. 101290, 2021. ISSN 1474-0346.

CHEN, Chi-Tsong. **Linear System Theory and Design**. 3rd. New York, NY, USA: Oxford University Press, Inc., 1998. ISBN 0195117778.

CHEN, Hong; KREMLING, H; ALLGÖWER, Frank. Nonlinear Predictive Control of a Benchmark CSTR. **Proceedings of the 3rd European Control Conference, Rome-Italy.**, p. 3247–3252, Jan. 1995.

CHEN, Qiang; SHI, Linlin; NA, Jing; REN, Xuemei; NAN, Yurong. Adaptive echo state network control for a class of pure-feedback systems with input and output constraints. **Neurocomputing**, Elsevier, v. 275, p. 1370–1382, 2018.

CISNEROS, P. S. G.; WERNER, H. A Dissipativity Formulation for Stability Analysis of Nonlinear and Parameter Dependent MPC. In: 2018 Annual American Control Conference (ACC). [S.l.: s.n.], 2018. P. 3894–3899.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of Control, Signals, and Systems (MCSS)**, Springer London, v. 2, n. 4, p. 303–314, Dec. 1989.

DAHL, J.; VANDENBERGHE, L. **CVXOPT: A Python package for convex optimization**. [S.l.: s.n.], 2020. Available from: `http://abel.ee.ucla.edu/cvxopt/`.

DANESHFAR, Fatemeh; JAMSHIDI, Mohammad (Behdad). An octonion-based nonlinear echo state network for speech emotion recognition in Metaverse. **Neural Networks**, v. 163, p. 108–121, 2023. ISSN 0893-6080.

DOYA, K. Bifurcations in the learning of recurrent neural networks. In: PROC. of IEEE Int. Symp. on Circ. and Syst. [S.l.: s.n.], 1992. v. 6, p. 2777–2780.

EREN, Utku; PRACH, Anna; KOÇER, Başaran Bahadır; RAKOVIĆ, Saša V.; KAYACAN, Erdal; AÇIKMEŞE, Behçet. Model Predictive Control in Aerospace Systems: Current State and Opportunities. **Journal of Guidance, Control, and Dynamics**, v. 40, n. 7, p. 1541–1566, 2017. eprint: `https://doi.org/10.2514/1.G002507`.

FLOREZ, Horacio; GILDIN, Eduardo. Global/local model-order reduction in coupled flow and linear thermal-poroelasticity. **Computational Geosciences**, p. 1, May 2019.

GHASEMI, Mohammadreza; IBRAHIM, Ashraf; GILDIN, Eduardo. Reduced Order Modeling In Reservoir Simulation Using the Bilinear Approximation Techniques. **SPE Latin American and Caribbean Petroleum Engineering Conference Proceedings**, v. 2, May 2014.

GLOROT, Xavier; BORDES, Antoine; BENGIO, Yoshua. Deep Sparse Rectifier Neural Networks. In: PROCEEDINGS of the Fourteenth International Conference on Artificial Intelligence and Statistics. Fort Lauderdale, FL, USA: PMLR, Nov. 2011. v. 15. (Proceedings of Machine Learning Research), p. 315–323.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. [S.l.]: MIT Press, 2016. `http://www.deeplearningbook.org`.

HALUSZCZYNSKI, Alexander; AUMEIER, Jonas; HERTEUX, Joschka; RÄTH, Christoph. Reducing network size and improving prediction stability of reservoir computing. **Chaos: An Interdisciplinary Journal of Nonlinear Science**, AIP Publishing, v. 30, n. 6, p. 063136, June 2020.

HE, Zhengyu. Deep Learning in Image Classification: A Survey Report. In: 2020 2nd International Conference on Information Technology and Computer Application (ITCA). [S.l.: s.n.], 2020. P. 174–177.

HINAUT, Xavier; DOMINEY, Peter F. On-Line Processing of Grammatical Structure Using Reservoir Computing. In: INT. Conf. on Artificial Neural Networks. Lausanne, Switzerland: European Neural Networks Society, Sept. 2012. P. 596–603.

HOCHMAN, Amit; BOND, Bradley N.; WHITE, Jacob K. A stabilized discrete empirical interpolation method for model reduction of electrical, thermal, and microelectromechanical systems. In: 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC). [S.l.: s.n.], 2011. P. 540–545.

HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-term Memory. **Neural computation**, v. 9, p. 1735–80, Dec. 1997.

HOU, Zhong-Sheng; WANG, Zhuo. From model-based control to data-driven control: Survey, classification and perspective. **Information Sciences**, v. 235, p. 3–35, 2013.

JAEGER, Herbert. **Short term memory in echo state networks**. [S.l.], Mar. 2002.

JAEGER, Herbert. **The ''echo state'' approach to analysing and training recurrent neural networks – with an Erratum note**. [S.l.], 2001.

JAEGER, Herbert; HAAS, Harald. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. **Science**, v. 304, n. 5667, p. 78–80, Apr. 2004.

JAEGER, Herbert; LUKOSEVICIUS, 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.

JAHANSHAHI, Esmaeil; SKOGESTAD, Sigurd. Simplified dynamical models for control of severe slugging in multiphase risers. **IFAC Proceedings Volumes**, v. 44, n. 1, p. 1634–1639, 2011.

JAHANSHAHI, Esmaeil; SKOGESTAD, Sigurd; HANSEN, Henrik. Control structure design for stabilizing unstable gas-lift oil wells. **IFAC Proceedings Volumes**, v. 45, n. 15, p. 93–100, 2012.

JAHN, Frank; COOK, Mark; GRAHAM, Mark. **Hydrocarbon Exploration and Production**. 2nd ed. [S.l.]: Elsevier, 2008. (Developments in Petroleum Science 55).

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

JORDANOU, Jean P.; ANTONELO, Eric Aislan; CAMPONOGARA, Eduardo. Online Learning Control with Echo State Networks of an Oil Production Platform. **Eng. Appl. Artif. Intell.**, 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. ISSN 1474-0346.

JORDANOU, Jean Panaioti; AISLAN ANTONELO, Eric; CAMPONOGARA, Eduardo; GILDIN, Eduardo. Investigation of proper orthogonal decomposition for echo state networks. **Neurocomputing**, v. 548, 2023. ISSN 0925-2312.

JORDANOU, Jean Panaioti; ANTONELO, Eric Aislan; CAMPONOGARA, Eduardo. Echo State Networks for Practical Nonlinear Model Predictive Control of Unknown Dynamic

Systems. **IEEE Transactions on Neural Networks and Learning Systems**, v. 33, n. 6, p. 2615–2629, 2022.

KHALIL, Hassan K. **Nonlinear Systems**. [S.l.: s.n.], Mar. 2018. ISBN 0130673897.

KINGMA, Diederik P.; BA, Jimmy. **Adam: A Method for Stochastic Optimization**. [S.l.: s.n.], 2014. arXiv: 1412.6980 [cs.LG].

KOHLER, Johannes; ZEILINGER, Melanie N.; GRUNE, Lars. Stability and Performance Analysis of NMPC: Detectable Stage Costs and General Terminal Costs. **IEEE Transactions on Automatic Control**, Institute of Electrical and Electronics Engineers (IEEE), p. 1–16, 2023.

LIN, Xiaowei; YANG, Zehong; SONG, Yixu. Short-term Stock Price Prediction Based on Echo State Networks. **Expert Systems with Applications**, Pergamon Press, Inc., v. 36, n. 3, p. 7313–7317, 2009.

LIU, Wenjie; BAI, Yuting; JIN, Xuebo; WANG, Xiaoyi; SU, Tingli; KONG, Jianlei. Broad Echo State Network with Reservoir Pruning for Nonstationary Time Series Prediction. Ed. by Alexander Hošovský. **Computational Intelligence and Neuroscience**, Hindawi Limited, v. 2022, p. 1–15, Feb. 2022.

LIU, Yuexia; CHENG, Yunfei; WANG, Wu. A survey of the application of deep learning in computer vision. In: p. 68.

LØKSE, Sigurd; BIANCHI, Filippo Maria; JENSSEN, Robert. Training Echo State Networks with Regularization Through Dimensionality Reduction. **Cognitive Computation**, Springer Science and Business Media LLC, v. 9, n. 3, p. 364–378, Jan. 2017.

LUKOŠEVIČIUS, Mantas; MAROZAS, Vaidotas. Noninvasive fetal QRS detection using an echo state network and dynamic programming. **Physiological Measurement**, v. 35, n. 8, p. 1685–1697, July 2014.

MAASS, Wolfgang; MARKRAM, Henry. On the computational power of circuits of spiking neurons. **Journal of Computer and System Sciences**, v. 69, n. 4, p. 593–616, 2004. ISSN 0022-0000.

MALIK, Z. K.; HUSSAIN, A.; WU, Q. J. Multilayered Echo State Machine: A Novel Architecture and Algorithm. **IEEE Transactions on Cybernetics**, v. 47, n. 4, p. 946–959, Apr. 2017. ISSN 2168-2275.

MEHREZ, Mohamed W. Optimization based Solutions for Control and State Estimation in Dynamical Systems (Implementation to Mobile Robots) A Workshop. University Lecture. [S.l.], 2019.

MIJALKOVIC, Slobodan. Truly Nonlinear Model-Order Reduction Techniques. In: p. 1–5.

MOZER, Michael C. A Focused Backpropagation Algorithm for Temporal Pattern Recognition. **Complex Systems**, v. 3, n. 4, 1989.

NASCIMENTO, Tiago P; DÓREA, Carlos Eduardo Trabuco; GONÇALVES, Luiz Marcos G. Nonlinear model predictive control for trajectory tracking of nonholonomic mobile robots: A modified approach. **International Journal of Advanced Robotic Systems**, v. 15, n. 1, 2018. eprint: `https://doi.org/10.1177/1729881418760461`.

NELLES, Oliver. **Nonlinear System Identification: From Classical Approaches to Neural Networks, Fuzzy Models, and Gaussian Processes**. [S.l.]: Springer International Publishing, 2020. ISBN 9783030474393.

OTTER, Daniel W.; MEDINA, Julian R.; KALITA, Jugal K. **A Survey of the Usages of Deep Learning in Natural Language Processing**. [S.l.: s.n.], 2019. arXiv: `1807.10854 [cs.CL]`.

OUALI, Mohammed Assam; LADJAL, Mohamed. Nonlinear Dynamical Systems Modelling and Identification Using Type-2 Fuzzy Logic. Metaheuristic Algorithms Based Approach. In: IEEE. 2020 International Conference on Electrical Engineering (ICEE). [S.l.: s.n.], 2020. P. 1–6.

OZTURK, Mustafa C.; XU, Dongming; PRÍNCIPE, José C. Analysis and Design of Echo State Networks. **Neural Computation**, v. 19, n. 1, p. 111–138, 2007.

PAN, Y.; WANG, J. Model Predictive Control of Unknown Nonlinear Dynamical Systems Based on Recurrent Neural Networks. **IEEE Transactions on Industrial Electronics**, v. 59, n. 8, p. 3089–3101, 2012.

PASZKE, Adam et al. Automatic Differentiation in PyTorch. In: NIPS 2017 Workshop on Autodiff. Long Beach, California, USA: [s.n.], 2017.

PAVLOV, A.; KRISHNAMOORTHY, D.; FJALESTAD, K.; ASKE, E.; FREDRIKSEN, M. Modelling and model predictive control of oil wells with Electric Submersible Pumps. In: IEEE Conference on Control Applications (CCA). [S.l.: s.n.], 2014. P. 586–592.

PLUCÊNIO, A.; PAGANO, D.J.; BRUCIAPAGLIA, A.H.; NORMEY-RICO, J.E. A PRACTICAL APPROACH TO PREDICTIVE CONTROL FOR NONLINEAR PROCESSES. **IFAC Proceedings Volumes**, v. 40, n. 12, p. 210–215, 2007.

PLUCÊNIO, Agustinho. **Development of Non Linear Control Techniques for the Lifting of Multiphase Fluids**. 2013. PhD thesis – Federal University of Santa Catarina, Brazil.

RODAN, Ali; TINO, Peter. Minimum complexity echo state network. IEEE Trans Neural Netw. **IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council**, v. 22, p. 131–44, Nov. 2010.

SAKEMI, Yusuke; MORINO, Kai; LELEU, Timothée; AIHARA, Kazuyuki. Model-size reduction for reservoir computing by concatenating internal states through time. **Scientific Reports**, Springer Science and Business Media LLC, v. 10, n. 1, Dec. 2020.

SALEHINEJAD, Hojjat; SANKAR, Sharan; BARFETT, Joseph; COLAK, Errol; VALAEE, Shahrokh. **Recent Advances in Recurrent Neural Networks**. [S.l.: s.n.], 2017. arXiv: 1801.01078 [cs.NE].

SCHRAUWEN, Benjamin; VERSTRAETEN, David; VAN CAMPENHOUT, Jan. An overview of reservoir computing: theory, applications and implementations. In: PROCEEDINGS of the 15th European Symposium on Artificial Neural Networks. [S.l.: s.n.], 2007. P. 471–482.

SELGA, Rosa Castañé; LOHMANN, Boris; EID, Rudy. Stability Preservation in Projection-based Model Order Reduction of Large Scale Systems. **European Journal of Control**, v. 18, n. 2, p. 122–132, 2012. ISSN 0947-3580.

SILJAK, Dragoslav D. **Large-Scale Dynamic Systems: Stability and Structure (Dover Civil and Mechanical Engineering)**. [S.l.]: Dover Publications, Nov. 2007. ISBN 0486462854.

SIMON, Daniel; LOFBERG, Johan. Stability analysis of model predictive controllers using Mixed Integer Linear Programming. In: 2016 IEEE 55th Conference on Decision and Control (CDC). [S.l.]: IEEE, Dec. 2016.

SIVARAMAKRISHNAN, Janardhanan. **Model Order Reduction and Controller Design Techniques**. [S.l.: s.n.], Apr. 2013.

SUN, Xian-hang; XU, Ming-hai. Optimal control of water flooding reservoir using proper orthogonal decomposition. **Journal of Computational and Applied Mathematics**, v. 320, p. 120–137, 2017. ISSN 0377-0427.

TANAKA, Gouhei; YAMANE, Toshiyuki; HÉROUX, Jean Benoit; NAKANE, Ryosho; KANAZAWA, Naoki; TAKEDA, Seiji; NUMATA, Hidetoshi; NAKANO, Daiju; HIROSE, Akira. Recent advances in physical reservoir computing: A review. **Neural Networks**, v. 115, p. 100–123, 2019. ISSN 0893-6080.

TARBOURIECH, S.; PRIEUR, C.; SILVA, J.M.G. da. Stability analysis and stabilization of systems presenting nested saturations. **IEEE Transactions on Automatic Control**, v. 51, n. 8, p. 1364–1371, 2006.

TARBOURIECH, Sophie; GARCIA, Germain; SILVA, João; QUEINNEC, Isabelle. **Stability and Stabilization of Linear Systems with Saturating Actuators**. [S.l.: s.n.], Jan. 2011.

TARBOURIECH, Sophie; GARCIA, Germain; SILVA, João Manoel Gomes da; QUEINNEC, Isabelle. **Stability and Stabilization of Linear Systems with Saturating Actuators**. [S.l.]: Springer London, 2011.

TIBSHIRANI, Robert. Regression Shrinkage and Selection via the Lasso. **Journal of the Royal Statistical Society. Series B (Methodological)**, [Royal Statistical Society, Wiley], v. 58, n. 1, p. 267–288, 1996.

TORRES, José F.; HADJOUT, Dalil; SEBAA, Abderrazak; MARTÍNEZ-ÁLVAREZ, Francisco; TRONCOSO, Alicia. Deep Learning for Time Series Forecasting: A Survey. **Big Data**, Mary Ann Liebert Inc, v. 9, n. 1, p. 3–21, Feb. 2021.

VERSTRAETEN, Davd; DAMBRE, Joni; DUTOIT, Xavier; SCHRAUWEN, Benjamin. Memory versus non-linearity in reservoirs. In: INT. Joint Conference on Neural Networks. Barcelona, Spain: IEEE, 2010. P. 18–23.

VERSTRAETEN, David; SCHRAUWEN, Benjamin. On the Quantification of Dynamics in Reservoir Computing. In: ALIPPI, Cesare; POLYCARPOU, Marios; PANAYIOTOU, Christos; ELLINAS, Georgios (Eds.). **Artificial Neural Networks**. Berlin: Springer, 2009. P. 985–994.

WANG, Hai; LONG, Xingyi; LIU, Xue-Xin. fastESN: Fast Echo State Network. **IEEE Transactions on Neural Networks and Learning Systems**, 2022.

WANG, Yi; YU, Bo; WANG, Ye. Acceleration of Gas Reservoir Simulation Using Proper Orthogonal Decomposition. **Geofluids**, Hindawi Limited, v. 2018, p. 1–15, 2018.

WATANABE, Shuhei. **Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance**. [S.l.]: arXiv, 2023. Available from: `https://arxiv.org/abs/2304.11127`.

WATTER, Manuel; SPRINGENBERG, Jost Tobias; BOEDECKER, Joschka; RIEDMILLER, Martin A. Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images. **CoRR**, abs/1506.07365, 2015. arXiv: `1506.07365`.

WHITEAKER, Brian; GERSTOFT, Peter. Reducing echo state network size with controllability matrices. **Chaos: An Interdisciplinary Journal of Nonlinear Science**, AIP Publishing, v. 32, n. 7, p. 073116, July 2022.

XIANG, K.; LI, B. N.; ZHANG, L.; PANG, M.; WANG, M.; LI, X. Regularized Taylor Echo State Networks for Predictive Control of Partially Observed Systems. **IEEE Access**, v. 4, p. 3300–3309, 2016.

YANG, Cuili; WU, Zhanhong. Multi-objective sparse echo state network. **Neural Computing and Applications**, Springer Science and Business Media LLC, Sept. 2022.

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