Nonlinear Model Predictive Control of Electrical Submersible Pumps based on Echo State Networks

Jean P. Jordanou[a], Iver Osnes[b], Sondre B. Hernes[b], Eduardo Camponogara[a], Eric Aislan Antonelo[a], Lars Imsland[b]

[a]*Department of Automation and Systems Engineering, Federal University of Santa Catarina, 88040-900 Florianópolis, Brazil, e-mail: eric.antonelo@ufsc.br, eduardo.camponogara@ufsc.br*
[b]*Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway, e-mail: lars.imsland@ntnu.no*

Abstract

Employed for artificial lifting in oil well production, Electrical Submersible Pumps (ESP) can be operated with Model Predictive Control (MPC) to drive an optimal production, while ensuring a safe operation and respecting system constraints. Due to the nonlinear dynamics of ESPs, Echo State Networks (ESNs), a recurrent neural network with fast training, are employed for efficient system identification of unknown dynamic systems. Besides the synthesis of highly accurate prediction models, this work contributes by designing two Nonlinear MPC (NMPC) strategies for the control of an ESP-lifted oil well: a standard Single-Shooting NMPC that embeds the ESN model completely, and the Practical Nonlinear Model Predictive Controller (PNMPC) that approximates the NMPC through fast trajectory-linearization of the ESN model. Another contribution is the implementation of an error correction filter to reject disturbances and counter modeling errors in both NMPC strategies. Finally, in computational experiments, both ESN-based NMPC strategies performed well in controlling simulated ESP-lifted oil wells when the model of the plant is unknown. However, PNMPC was more efficient and induced a similar performance to standard NMPC.

Keywords: Echo State Networks, Nonlinear Model Predictive Control, Control of Unknown Dynamic Systems, Oil Production Wells, Electrical Submersible Pumps

---

1. Introduction

An oil well consists of tubing that enables the extraction of hydrocarbons from a reservoir, lifting the fluids up to a platform for processing and storage (Jahn et al., 2008). When an oil well is drilled in a young reservoir, the internal pressure is sufficiently high to sustain the natural flow of fluids. As the oilfield matures, however, the pressure provided by the reservoir gradually diminishes, thereby raising the need to implement artificial lifting techniques.

Artificial lifting is widely used to boost oil production in wells (Jahn et al., 2008). Two notable forms of artificial lifting are gas-lift, and pumping systems such as the Electrical Submersible Pump (ESP), which are implemented according to the specific characteristics of the oilfield and well (Takacs, 2017). The ESP consists of a multistage centrifugal pump embedded directly inside the well (Pavlov et al., 2014), which contributes to oil lifting by introducing a positive pressure differential upwards the vertical tubing of the well (Takacs, 2017). When compared to other lifting methods, ESPs have the advantage of being efficient for high production rates, working well in highly deviated wells. ESPs have a low maintenance cost assuming no user fault, and small space requirements (Takacs, 2017).

However, ESPs are expensive to implement and demand a robust electric power supply with high voltage when operating at high velocity. Among other problems, gas suction can compromise the ESP efficiency, and abrasive materials such as sand can easily damage the pump. ESPs are more likely to be taken out of operation by catastrophic faults, instead of fatigue over time; however, factors such as temperature, flow rate, vibration, and power consumption have an influence on the pump longevity (Binder et al., 2014). These issues, alongside the aforementioned problems involving energy savings, pump efficiency drop, and damage control, justify the deployment of an advanced control technique into an ESP well. This automatic control will ensure an enhanced safe production while imposing constraints that improve the ESP's useful life (Binder et al., 2014; Pavlov et al., 2014). This is very relevant when the number of ESP faults attributed to human error was reduced from 80% to 23% after the operational constraints for the ESP wells were implemented in industry (Centrilift, 2008).

Among the advanced control techniques, Model Predictive Control (MPC) stands out as the ideal choice for operating an ESP-lifted oil well. MPC consists generally of a prediction model of the plant, and an optimization problem describing the system operation which is solved to compute the control actions (Camacho and Bordons, 1999). All the operational requirements for an ESP-lifted oil well can be implemented in the form of an optimization problem with constraints, such as in Pavlov et al. (2014), where a Nonlinear Model Predictive Control (NMPC) is implemented using a simplified, physics-based ESP model as the predictive model. While an NMPC is efficient in controlling a plant, the controller needs to solve an NLP (Nonlinear Programming) problem per time step, which tends to be costly. An alternative solution to the application of a full NMPC is the so-called Practical NMPC (Plucênio, 2013), denoted PNMPC, which uses a Taylor approximation of the nonlinear prediction model, making the computation of each control action less expensive.

Oil production plants involve complex phenomena that are hard to capture in a physical model, such as uncertainties associated with fluid production and other structural uncertainties. Data-driven modeling is the process of obtaining a system model through collected data, also called black-box system identification (Nelles, 2001). One of the main methods used in that context comes from models based on neural networks (Salehinejad et al., 2017), which includes feed-forward neural networks with output feedback, Radial Basis Function networks (Ayala et al., 2020), neuro-fuzzy models (Ouali and Ladjal, 2020) and Recurrent Neural Networks (RNN) (such as Long-Short Term Memory (LSTM) networks) (Salehinejad et al., 2017).

Our work focuses on recurrent neural networks, considered as universal approximators to dynamic systems. They consist of an internal dynamics type model (Nelles, 2001) that corresponds to a network with internal states (or memory). Training in conventional RNNs involves calculating gradients of the loss function with respect to the model parameters taking into account the evolving internal dynamics. This is usually done through Backpropagation Through Time (BPTT) (Mozer, 1995), which has numerical issues with vanishing or exploding RNN gradients as deep networks do. Popular methods such as the LSTM mitigate that problem by introducing specialized gates into the network structure (Salehinejad et al., 2017). However, such approaches increase the structural complexity of each network neuron, while still being trained through iterative nonlinear optimization methods. Instead, our work employs Echo State Networks (ESN) (Jaeger et al., 2007) to model dynamic processes, which is a type of RNN that provides an alternative and more efficient solution to conventional RNNs.

Categorized under the Reservoir Computing paradigm (Schrauwen et al., 2007), ESNs (Jaeger and Haas, 2004) are composed of two main layers: a recurrent non-adaptive layer which represents a pool of rich nonlinear dynamics, called "dynamic reservoir."; and a readout adaptive output layer which is a linear instantaneous combination of the dynamics from the fixed reservoir. System identification is achieved by training solely the memoryless output layer. As a result, training can be achieved with algorithms such as Ridge Regression (Jaeger, 2001), or Recursive Least Squares (RLS) (Jordanou et al., 2018). Assuming that the dynamic reservoir is sufficiently rich, a large number of different nonlinear systems can be identified, qsuch as learning complex goal-directed robot behaviors (Antonelo and Schrauwen, 2015), grammatical structure processing (Hinaut and Dominey, 2012), short-term stock prediction (Lin et al., 2009), and noninvasive fetal detection (Lukoševičius and Marozas, 2014). Also, in Chen and Liu (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.

In this work, ESNs are used with NMPC strategies to achieve control of unknown nonlinear plants, more specifically, of ESP-lifted oil wells. We investigate and compare full NMPC to PNMPC in successfully tracking the bottom-hole pressure of such ESP-lifted oil wells in different experiments.

1.1. Related Works

Conceived by Pavlov et al. (2014), the ESP-lifted oil well model adopted in this work was designed to serve as a simple simulator for testing MPC controllers before deployment to real ESP plants. Such model provides a simplified model concerning the pump energy transfer and pressure increase when compared to works such as Yang et al. (2021b) and Yang et al. (2021a). These latter works describe the internal properties of the ESP in more detail, considering a three-stage ESP and contributing with a model that describes the internal pressure pulsation.

Pavlov et al. (2014) also proposed a linear MPC to control such plants, using the model as a simulation test, and then employing the controller in a real testing facility. The implementation of one such MPC is reported by Binder et al. (2014), where an MPC was implemented in a Programmable Logic Controller (PLC), tested using the simulation model of Pavlov et al. (2014), and then in the testing facility. Binder et al. (2015) used the very same model as a Moving Horizon Estimator, which is a type of observer that minimizes the quadratic error along a horizon of past measurements, serving as a counterpart to observers in MPC. de A. Delou et al. (2019) states that the linear approximation of the model varies significantly with changes in the choke opening of the ESP-lifted well, and therefore develop an adaptive linear MPC, which applies a DMC controller that has a dynamic matrix employing a linear combination of two different step responses, with a weighting parameter as a function of the choke opening. Except for de A. Delou et al. (2019), all of the cited works implement a linear MPC to operate the same model of the ESP-lifted well. While linear MPC generally performs well, having low computation time as the controller only needs to solve one QP per time step, the linear model tends to lose precision in other operating points if the plant is nonlinear, and the controller performance is compromised if system behavior varies significantly depending on the operating region. The solution proposed by de A. Delou et al. (2019) is a convex combination of the step response in two opposite model regions. While the method was shown to perform well, there is no guarantee that a convex combination of two different step responses is represents well the nonlinear model. In contrast, our ESN-based approach consists of the use of a full nonlinear trained model, trading computational time performance for model precision and improving control performance.

The literature on MPC combined with Neural Networks is vast. For instance, Ławryńczuk (2014) presents a wide array of NMPC algorithms assuming different types of neural networks as the prediction model, including but not limited to Recurrent Neural Networks. His algorithms can be categorized under two main approaches: those that solve the NMPC by using the exact prediction model; and strategies that solve a QP by providing linear approximations of the neural network model. One such controller employed in this work (Plucênio et al., 2007), the Practical Nonlinear Model Predictive Control (PNMPC), was developed with model linearization in mind, applying an error correction filter to help reject disturbances and correct fine modeling errors. Our work differs from Plucênio et al. (2007) in that they assume the analytical derivative to be unavailable. As we are applying an ESN, we can easily obtain the derivatives analytically, through a recursive algorithm that computes the Jacobian along the whole prediction horizon.

A recent example of NMPC with Echo State Network is found in Cao and Huang (2020), where a variation of ESN (Echo State Gaussian Process) is proposed as a black-box model for the NMPC of a Pneumatic Muscle Actuator. Works of PNMPC and linearized MPC associated with ESNs are also found in the literature, such as Pan and Wang (2012), which linearizes the ESN into a state-space formulation. Xiang et al. (2016) employed a Taylor-linearized ESN as a prediction model, while the full nonlinear ESN serves as an observer. This topology implies a loss of precision when the system is far from the initially designed operating point, and also the fact that they identify a nonlinear model for what essentially is a linear MPC. In another work, Terzi et al. (2020) identified several Recurrent Neural Network models for application in a linearized NMPC to control the cooling system of a large business center.

Gros et al. (2016) compare linear to nonlinear MPC, postulating the so-called Real-Time Iteration (RTI) concept. Because they use Sequential Quadratic Programming (SQP) for optimization, multiple QPs are solved by NMPC at each sample time, whereas linear MPC demands the solution of just one QP. The Real-Time Iteration consists in performing one iteration of SQP for NMPC, which is similar to what PNMPC does in our work. The difference being that PNMPC performs trajectory linearization over the free response, while RTI reduces the nonlinear system into a linear state-space system. The implication is that trajectory linearization has full precision on the free response in relation to the full nonlinear model. This same comparison with PNMPC also holds for Pan and Wang (2012), which approximates the nonlinear ESN as a state-space linear model at each iteration. Directly including the ESN states in an optimization problem has negative implications to the computational time for solving QPs, which can become large due to the number of ESN states.

Armenio et al. (2019) develop a stability condition for the ESN to be effective in MPC (Input-State Stability), applying an NMPC for neutralizing pH in a simulated reactor. The main difference between the NMPC of Armenio et al. (2019) and the NMPC in our work is that we employ an error correction filter, whereas Armenio et al. (2019) use a Luenberger full-order observer for state correction, which is more computationally demanding, since there are as many gains as there are ESN states.

There are other works that apply Neural Networks for control outside the scope of MPC, such as Jordanou et al. (2018), that deploys two ESN to perform plant control by identifying the inverse model. Another one is Chang et al. (2004), where a Feedforward Neural Network is designed as a model for adaptive control. The Neural Network is combined with an Extended Kalman Filter approach for parameter update.

1.2. Contributions

The contributions of this work are:

- A demonstration from a simulated case study that echo state networks can be trained to model the dynamics of ESP-lifted oil wells directly from input-output data, without prior knowledge on the system structure.

- The design of two NMPC strategies relying on the ESN modeling of the plant, the first being a standard Single-Shooting NMPC that embeds the full ESN model, and the second following the Practical NMPC approach which relies on a linear approximation of the ESN model and fast computation of sensitivities.

- The derivation of the error correction filter used in PNMPC in state space, and its inclusion in the Single-shooting NMPC formulation to correct modeling errors and reject disturbances.

1.3. Paper Organization

Section 2 presents the ESP-elevated oil well and formalizes the control problem. Section 3 explains the Echo State Network. Section 4 describes the NMPC strategies based on ESNs designed to solve the ESP control problem, namely the Single Shooting NMPC and PNMPC. Section 5 reports on the experiments and discusses the results from the system identification and NMPC of the well. Section 6 draws conclusions of this work.

2. Electrical Submersible Pump in Artificial Lifting of Oil Wells

2.1. Mathematical Model

The mathematical model of the ESP-elevated Oil Well (depicted in Figure 1) was developed by Pavlov et al. (2014), with some improvements by Binder et al. (2015). It employs the average flow of a mixture as state, instead of taking into account each phase flow. This makes the model simpler to compute and use in control applications.

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

$$\dot{p}_{bh} = \frac{V_1}{\beta_1}(q_r - q) \tag{1a}$$

$$\dot{p}_{wh} = \frac{V_2}{\beta_2}(q - q_c) \tag{1b}$$

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

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:

$$q_r = PI(p_r - p_{bh}) \tag{2a}$$
$$q_c = C_c z \sqrt{p_{wh} - p_m} \tag{2b}$$

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.

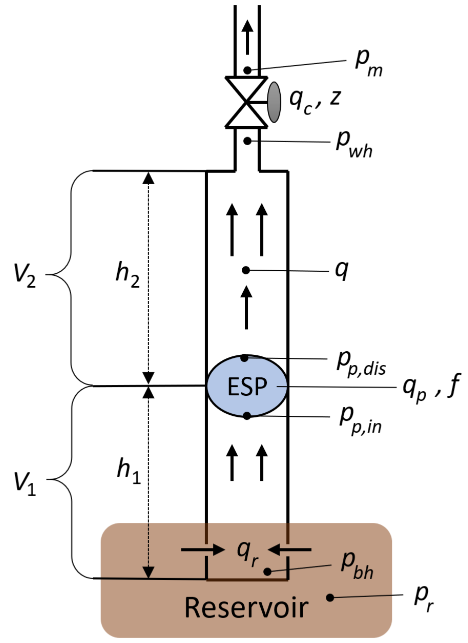

Figure 1: Schematic representation of an ESP lifted well, adapted from Binder et al. (2014).

The load loss due to friction is calculated as:

$$\Delta p_f = F_1 + F_2 \tag{3a}$$
$$F_i = 0.158 \frac{\rho L_i q^2}{D_i A_i^2} \left(\frac{\mu}{\rho D_i q} \right)^{\frac{1}{4}} \tag{3b}$$

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.

The ESP pressure increase is calculated as:

$$\Delta P_p = \rho g H \tag{4a}$$

$$H = C_H(\mu)\left(c_0 + c_1\left(\frac{q}{C_Q(\mu)}\frac{f_0}{f}\right) - c_2\left(\frac{q}{C_Q(\mu)}\frac{f_0}{f}\right)^2\left(\frac{f}{f_0}\right)^2\right) \tag{4b}$$

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 1 presents the model parameters of the ESP-lifted oil well.

Table 1: 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

2.2. Problem Statement

As one can see, the model described by the state equation system (1) is a simplified and lumped model for the ESP which is used as a reference in this work. In fact, models for offshore oil production are not very accurate (Jahn et al., 2008), since the composition of the fluid coming from the reservoir is actually not well known, which heavily affects the structure of the equations. However, models such as (1) can capture the fundamental behavior of this type of process.

The first problem is to identify a black-box model of dynamical interactions between relevant variables for the NMPC of the ESP-equipped oil well, which is represented in this work by the numerical simulation of (1). In this case, we want to identify the manipulated variables of the well as inputs to the model, to which end we gather data from the pump rotation frequency f and the valve choke opening z. As outputs, we utilize the system states: the bottom-hole pressure p_{bh}, the wellhead pressure p_{wh}, and the average flow rate of the fluid q, as they are quantities of interest in the model. There are many possible black-box models in the literature (Nelles, 2001), including Recurrent Neural Networks (Schmidhuber, 2015). We chose Echo State Networks as a black box model because of their ease to train and high capability of approximating nonlinear dynamic systems. ESNs are explained in the next section.

The next step is to employ the trained ESN as the model for an NMPC problem that minimizes a certain objective function while respecting operational constraints so that the ESP operational lifespan increases (Takacs, 2017). In this work, we consider the constraints that were applied by Pavlov et al. (2014). 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. As for the system outputs, the constraints consist of bounds on the bottom-hole pressure, wellhead 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 objective function in this work is the quadratic error of the tracking outputs when compared to their reference signals, over a given prediction horizon N_y, and the variation on control input signals over a control horizon N_u. When expressed mathematically, these constraints, together with the objective function, lead to the following optimization problem to be solved at discrete time k:

$$\min_{\mathbf{\Delta U}} \; J(\mathbf{y}[k], \mathbf{u}[k], \mathbf{\Delta U}) \tag{5a}$$

$$\text{s.t.} : \mathbf{1}_{N_u} \otimes (\mathbf{u}_{\min} - \mathbf{u}[k]) \leq (\mathbf{T}_{N_u} \otimes \mathbf{I}_m)\mathbf{\Delta U} \leq \mathbf{1}_{N_u} \otimes (\mathbf{u}_{\max} - \mathbf{u}[k]) \tag{5b}$$

$$\mathbf{1}_{N_y} \otimes \mathbf{y}_{\min} \leq \mathbf{Y} \leq \mathbf{1}_{N_y} \otimes \mathbf{y}_{\max} \tag{5c}$$

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 (5) are:

$$\mathbf{\Delta U} = \begin{pmatrix} \mathbf{\Delta u}[k] \\ \mathbf{\Delta u}[k+1] \\ \mathbf{\Delta u}[k+2] \\ \vdots \\ \mathbf{\Delta u}[k+N_u-1] \end{pmatrix}, \ \mathbf{Y} = \begin{pmatrix} \mathbf{y}[k+1] = \mathbf{g}(\mathbf{x}[k+1]) \\ \mathbf{y}[k+2] = \mathbf{g}(\mathbf{x}[k+2]) \\ \mathbf{y}[k+3] = \mathbf{g}(\mathbf{x}[k+3]) \\ \vdots \\ \mathbf{y}[k+N_y] = \mathbf{g}(\mathbf{x}[k+N_y]) \end{pmatrix}, \quad (6a)$$

$$\mathbf{u}[k+1] = \mathbf{u}[k] + \mathbf{\Delta u}[k], \ \mathbf{u} = \begin{pmatrix} z \\ f \end{pmatrix}, \ \mathbf{u}_{\min} = \begin{pmatrix} 0 \\ 35 \end{pmatrix}, \ \mathbf{u}_{\max} = \begin{pmatrix} 100 \\ 65 \end{pmatrix} \quad (6b)$$

$$\mathbf{y} = \begin{pmatrix} p_{bh} \\ p_{wh} \\ q \end{pmatrix}, \ \mathbf{y}_{\min} = \begin{pmatrix} 0 \\ 1 \\ 30 \end{pmatrix}, \ \mathbf{y}_{\max} = \begin{pmatrix} \infty \\ 60 \\ 50 \end{pmatrix} \quad (6c)$$

which lead to the objective function

$$J(\mathbf{y}[k], \mathbf{u}[k], \mathbf{\Delta U}) = (\mathbf{Y} - \mathbf{Y}_{ref})^T (\mathbf{I}_{N_y} \otimes \mathbf{Q})(\mathbf{Y} - \mathbf{Y}_{ref}) \\ + \mathbf{\Delta U}^T (\mathbf{I}_{N_u} \otimes \mathbf{R}) \mathbf{\Delta U} \quad (6d)$$

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 (1) 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. Such a case is assumed for this work. Strategies for solving the NMPC, such as the pure NLP-solving NMPC method and the PNMPC strategy, will be explained in Section 4.

3. Echo State Networks

According to Jaeger et al. (2007), the ESN is a recurrent neural network architecture that retains the same nonlinear system approximation capability of an RNN, while employing only a linear least squares algorithm as training. ESNs were conceived by Jaeger (2001) as a specialized form of the broader Reservoir Computing framework (Jaeger et al., 2007). Reservoir Computing (RC) is referred to as such because the network architecture consists of a dynamic, randomly initialized, untrainable layer, and a static layer that can be trained by least squares. The dynamic layer is referred to as a dynamic reservoir, as it must have a large

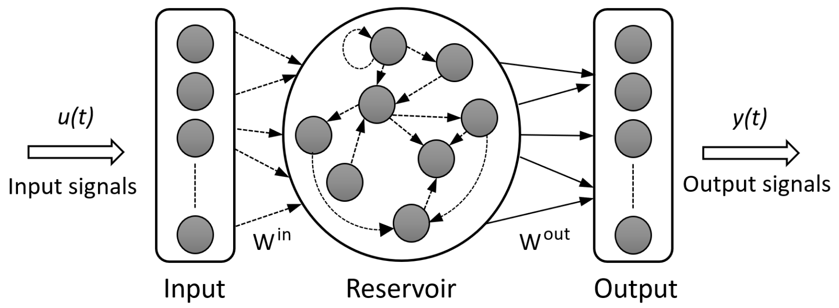

Figure 2: Representation of an Echo State Network. Solid connections (from Reservoir to Output Layer) are trainable, while dashed connections are fixed and randomly initialized.

pool of distinct dynamics, while the network output imitates a real system through a linear combination of these dynamics, represented by the static output.

The following equations describe the dynamics of the ESN:

$$\mathbf{x}[k+1] = (1-\gamma)\mathbf{x}[k] + \gamma f(\mathbf{W}^{\mathbf{r}}\mathbf{x}[k] + \mathbf{W}^{\mathbf{in}}\mathbf{u}[k] + \mathbf{b}) \tag{7a}$$

$$\mathbf{y}[k+1] = \mathbf{W}^{\mathbf{out}}\mathbf{x}[k+1] \tag{7b}$$

where: the reservoir state is given by $\mathbf{x}[k]$, the network input is $\mathbf{u}[k]$ and the output is $\mathbf{y}[k]$; $\gamma \in (0,1]$ is called leak rate (Jaeger et al., 2007), which is the percentage of the current state $\mathbf{x}[k]$ transferred into the next state $\mathbf{x}[k+1]$; the weights $\mathbf{W}^{\mathbf{r}}, \mathbf{W}^{\mathbf{in}}, \mathbf{b}$ and $\mathbf{W}^{\mathbf{out}}$ are the reservoir, input, bias and output weights, respectively; and $f = \tanh(\cdot)$ is the activation function for each reservoir neuron, also called a base function in system identification theory being widely used in the literature (Nelles, 2001). Figure 2 depicts the schematic of an echo state network.

The network has N neurons, which is the dimension of $\mathbf{x}[k]$ that must be several orders higher than the number of network inputs. As long as the training is regularized, N can be as large as needed, but at the expense of increased computation time of the model.

The procedure for training an ESN is (Jaeger et al., 2007):

1. Every weight of the network is initialized from a normal distribution $\mathcal{N}(0,1)$.
2. $\mathbf{W}^{\mathbf{r}}$ is scaled so that its spectral radius ρ (Eigenvalue with the largest module) is at a certain value which is able to create reservoirs with rich dynamical capabilities. Having $|\rho| < 1$ is a rule of thumb that works well in most cases, which also means that the linear expression inside the activation function induces a stable dynamic system (Jaeger et al., 2007).

3. $\mathbf{W^{in}}$ and $\mathbf{b}$ are multiplied by scaling factors f_i^r and f_b^r, respectively, to determine how the input will influence the network. The scaling parameters ρ, f_i^r, and f_b^r are crucial in the learning performance of the network, which have an impact on the nonlinear representation and memory capacity of the reservoir (Verstraeten et al., 2010).
4. After all the non-trainable dynamic weights are defined, what is left is to obtain $\mathbf{W^{out}}$ through Ridge Regression (Bishop, 2006). First, we run the dynamic part by entering the input data $\mathbf{u}[k]$ sequentially. Afterwards, we concatenate the set of state vectors reached by the ESN in a matrix:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}[0]^T \\ \mathbf{x}[1]^T \\ \vdots \\ \mathbf{x}[N_d] \end{pmatrix} \tag{8}$$

   Then, with the output data matrix $\mathbf{D}$ that concatenates each desired output the same way, we calculate the output weights as follows:

$$\mathbf{W^{out}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{D} \tag{9}$$

   where λ is the regularization parameter which helps avoid overfitting.

4. Nonlinear Model Predictive Control

This section elaborates further on solution strategies employed to solve problem (5), which is a Nonlinear Model Predictive Control problem (Camacho and Bordons, 1999). Model Predictive Control refers to the use of a predictive model (in our case, the ESN) to anticipate the plant response and calculate the control action according to some optimization problem, such as (5) (Camacho and Bordons, 1999).

There are several MPC strategies in the literature that are generally defined by their prediction model, the strategy for disturbance treatment, and the methods for solving the optimization problem at hand (Camacho and Bordons, 1999). The universe of linear MPCs is already very well defined and well solved (Camacho and Bordons, 1999), whereas nonlinear MPC tends to be more challenging, as one must solve a nonlinear optimization problem at each time step to compute the control action.

Figure 3 depicts the schematic of a generic MPC controller actuating on an ESP-lifted oil well. The optimizer block refers to the solution process of problem

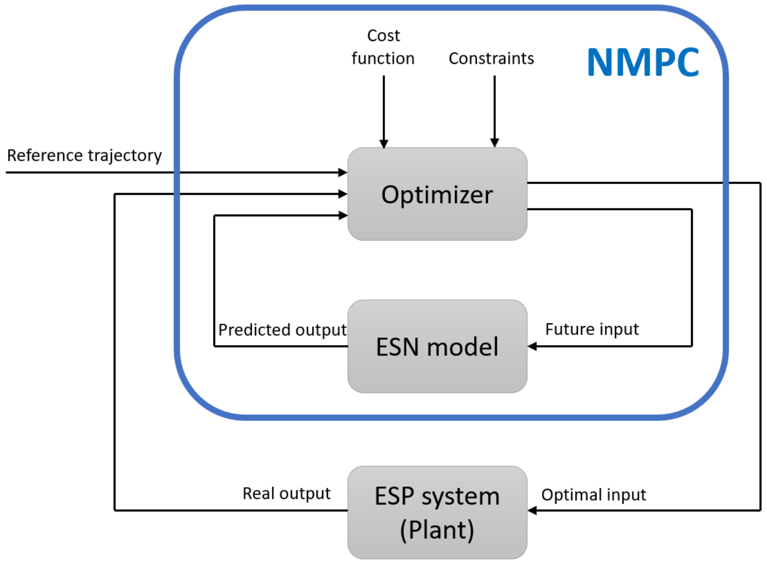

Figure 3: Representation of a generic NMPC controller using an ESN model to control an ESP-lifted oil well.

(5) that depends on two kinds of information: (i) the reference trajectory, which determines how the variables of interest in the system should behave; (ii) the predicted output over the horizon given a feasible input sequence, which lies within the set defined by the constraints. How exactly the model is designed and the optimizer implemented depends on the MPC strategy of choice.

In this work, we apply two different strategies for NMPC using an Echo State Network as the prediction model: (i) the pure solution of problem (5) following the Single Shooting (SS) strategy, and (ii) the Practical Nonlinear Model Predictive Control (PNMPC) strategy, as presented in (Jordanou et al., 2018).

4.1. Direct Single Shooting NMPC

The first strategy we explore consists of simply solving problem (5) directly. We assume the model to be the ESN itself and develop a formulation of the problem based on the concept of Single Shooting for a discrete-time dynamic system.

Single Shooting (SS) (Bock and Plitt, 1984) is a receding horizon strategy for solving optimal control problems such as (5). A shooting, in this context, is a simulation run given a certain input function (in continuous time) or, in the case of this work, a sequence in discrete time that yields a piecewise-constant control input. Because the SS method has only one shooting, all the outputs over the horizon are analytically bound to each other, leaving the ESN states as intermediate variables depending on a given control input sequence of $\mathbf{u}[k]$. In contrast, the multiple shooting method works with multiple system simulations, and each shooting has a respective initial condition defined by a decision variable in the optimization problem. In the context of ESN modeling, due to the large number of states, multiple shooting entails manipulating an excessively large number of decision variables and constraints in relation to the baseline problem. Also, as the ESN and the ESP-lifted well in this work are stable dynamical systems, SS is a valid choice to solve the optimization problem.

With single shooting in mind, and applying the ESN as a prediction model, the optimization problem (5) becomes:

$$\min_{\mathbf{U}} \sum_{i=1}^{N_y} \|\mathbf{W^{out}}\mathbf{x}[k+i] - \mathbf{y}_{ref}\|_{\mathbf{Q}} + \sum_{j=0}^{N_u-1} \|\mathbf{\Delta u}[k+j]\|_{\mathbf{R}} \tag{10a}$$

$$\text{s.t.}: \ \mathbf{1_{N_u}} \otimes (\mathbf{u}_{\min} - \mathbf{u}[k]) \leq (\mathbf{T_{N_u}} \otimes \mathbf{I_m})\mathbf{\Delta U} \leq \mathbf{1} \otimes (\mathbf{u}_{\max} - \mathbf{u}[k]) \tag{10b}$$

$$\mathbf{\Delta U}_{\min} \leq \mathbf{\Delta U} \leq \mathbf{\Delta U}_{\max} \tag{10c}$$

$$\mathbf{y}_{\min} \leq \mathbf{W^{out}}\mathbf{x}[k+i] \leq \mathbf{y}_{\max}, \ \forall i = \{1, 2, \ldots, N_y\} \tag{10d}$$

$$\mathbf{x}[k+i] = f^{(i)}(\mathbf{x}[k], \mathbf{U}), \ i = 1, \ldots, N_y \tag{10e}$$

in which the state prediction $\mathbf{x}[k+i]$ for time $(k+i)$ is calculated with the nonlinear operator $f^{(i)}$, defined by applying the ESN state-transition function recursively, as a function of the ESN state $\mathbf{x}[k]$ at the current time and the control sequence $\mathbf{U} = (\mathbf{u}[k+i] : i = 1, \ldots, N_u)$. More concretely,

$$\begin{aligned}
\mathbf{x}[k+1] &= f^{(1)}(\mathbf{x}[k], \mathbf{U}) \\
&= (1-\gamma)\mathbf{x}[k] + \gamma f(\mathbf{W^r}\mathbf{x}[k] + \mathbf{W^{in}}\mathbf{u}[k] + \mathbf{b}) && \text{(11a)} \\
\mathbf{x}[k+2] &= f^{(2)}(\mathbf{x}[k], \mathbf{U}) \\
&= (1-\gamma) f^{(1)}(\mathbf{x}[k], \mathbf{U}) \\
&\qquad + \gamma f(\mathbf{W^r} f^{(1)}(\mathbf{x}[k], \mathbf{U}) + \mathbf{W^{in}}\mathbf{u}[k+1] + \mathbf{b}) && \text{(11b)} \\
\vdots \ &= \quad \vdots \\
\mathbf{x}[k+N_y] &= f^{(N_y)}(\mathbf{x}[k], \mathbf{U}) \\
&= (1-\gamma) f^{(N_y-1)}(\mathbf{x}[k], \mathbf{U}) \\
&\qquad + \gamma f(\mathbf{W^r} f^{(N_y-1)}(\mathbf{x}[k], \mathbf{U}]) + \mathbf{W^{in}}\mathbf{u}[k+N_u] + \mathbf{b}) && \text{(11c)}
\end{aligned}$$

Notice that the ESN plant model is embedded in the nonlinear operators $f^{(i)}$, which renders the Single Shooting formulation (10) a Nonlinear Programming

(NLP) problem. Solving this problem to global optimality is not practical though, given the highly nonlinear and recursive structure of the ESN. Thus one contends with a local optimum that can be reached with standard NLP algorithms.

In the formulation (10), the predicted intermediate and final state of the plant are direct functions of the initial condition $\mathbf{x}[k]$, and the control sequence $\mathbf{U}$ over the control horizon. Unlike SS, Multiple Shooting (MS) does not rely on function composition and instead defines the intermediate state variables as decision variables. Single shooting was favored with respect to multiple shooting due to the high dimensionality of the resulting decision space.

There is a disadvantage of controlling a plant by solving the NMPC problem (10): the method is essentially open-loop, as information on the real plant is not used to compute the control action. Although this means that the control trajectory could be computed offline, the controller is not able to treat disturbances and modeling errors, which elicits the need of improving this strategy with regards to this aspect. To counter the open-loop limitation, a strategy based on filtering is presented in what follows.

4.2. Disturbance Treatment and Error Correction Filter

Several ways to deal with modeling errors and disturbance rejection are found in the MPC literature (Camacho and Bordons, 1999). In this work, the error between the output predicted by the ESN and the actual plant output is integrated and then filtered, with the result being added as a constant correction factor over the horizon, as done in (Plucênio et al., 2007; Jordanou et al., 2018).

Now, from the point of view of the optimization problem, the nonlinear model is represented as:

$$\mathbf{x}[k+i+1] = \mathbf{f}(\mathbf{x}[k+i], \mathbf{u}[k+i]) \tag{12a}$$

$$\mathbf{y}[k+i] = \mathbf{g}(\mathbf{x}[k+i]) + \boldsymbol{\delta}[k] \tag{12b}$$

with $\boldsymbol{\delta}[k]$ being the aforementioned correction factor.

This formulation assumes that the modeling error is constant along the whole horizon, which is a reasonable assumption for error correction that spares the MPC from accounting for future disturbances. The correction factor is always updated before the calculation of the control action, and obeys the following law:

$$\boldsymbol{\delta}[k] = \boldsymbol{\delta}[k-1] + K\Delta\boldsymbol{\delta}[k] \tag{13a}$$

$$\Delta\boldsymbol{\delta}[k] = (1-\omega)(\mathbf{y_m}[k] - \widehat{\mathbf{y}}[k|k-1]) + \omega\Delta\boldsymbol{\delta}[k-1] \tag{13b}$$

$$\widehat{\mathbf{y}}[k|\widehat{k}] = \mathbf{y_p}[k] + \boldsymbol{\delta}[\widehat{k}] \tag{13c}$$

where K is integrator gain, ω is the filter cutoff rate, $\mathbf{y_p}$ is the output calculated by the model (in this case, the ESN), $\mathbf{y_m}$ is the measured output, and $\widehat{k}$ is a time index that could be k or $k-1$. As seen from equation (13a), the correction factor works in a closed-loop fashion as it depends on values of $\boldsymbol{\delta}$ at previous time steps. After calculating $\boldsymbol{\delta}[k]$, the corrected predicted output $\widehat{\mathbf{y}}[k|k]$ is what essentially goes into the NLP that corresponds to MS formulation (10). Then the next model output prediction and output measurement are given, respectively $\mathbf{y_p}[k+1]$ and $\mathbf{y_m}[k+1]$, and the correction factor calculation procedure (13) is repeated after the control action is obtained.

It can be shown that the recursive calculation of $\boldsymbol{\delta}[k]$ leads to zero bias error. First, we define the *prior error* (before adding the correction factor into the prediction) and *posterior error* (after correction) of the MPC as:

$$\boldsymbol{\epsilon}[k] = \mathbf{y_m}[k] - \mathbf{y_p}[k] \tag{14a}$$
$$\mathbf{e}[k] = \mathbf{y_m}[k] - \widehat{\mathbf{y}}[k|k-1] \tag{14b}$$

From (28) and (13c) with $\widehat{k} = k-1$, the following equation relates $\boldsymbol{\epsilon}[k]$ to $\mathbf{e}[k]$:

$$\mathbf{e}[k] = \boldsymbol{\epsilon}[k] - \boldsymbol{\delta}[k-1] \tag{15}$$

Then, we express (13) in terms of $\mathbf{e}[k]$:

$$\boldsymbol{\delta}[k] = \boldsymbol{\delta}[k-1] + K\Delta\boldsymbol{\delta}[k] \tag{16a}$$
$$\Delta\boldsymbol{\delta}[k] = (1-\omega)\mathbf{e}[k] + \omega\Delta\boldsymbol{\delta}[k-1] \tag{16b}$$

Finally, we replace $\mathbf{e}[k]$ given by Eq. (15) in (16b) to obtain:

$$\begin{bmatrix} \boldsymbol{\delta}[k] \\ \Delta\boldsymbol{\delta}[k] \end{bmatrix} = \left[\begin{pmatrix} 1 & K \\ (\omega-1) & \omega \end{pmatrix} \otimes \mathbf{I} \right] \begin{bmatrix} \boldsymbol{\delta}[k-1] \\ \Delta\boldsymbol{\delta}[k-1] \end{bmatrix} + \left[\begin{pmatrix} 0 \\ 1-\omega \end{pmatrix} \otimes \mathbf{I} \right] \boldsymbol{\epsilon}[k] \tag{17}$$

which is the state space form of the filters, with $\boldsymbol{\epsilon}$ as the input and $\mathbf{I}$ as an identity matrix with matching dimension for $\boldsymbol{\delta}$. For univariate systems, the Kronecker product could be dropped. We now calculate the steady state for (17):

$$\begin{bmatrix} \boldsymbol{\delta} \\ \Delta\boldsymbol{\delta} \end{bmatrix} = \left[\begin{pmatrix} 0 & -K \\ 1-\omega & 1-\omega \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \omega-1 \end{pmatrix} \otimes \mathbf{I} \right] \boldsymbol{\epsilon} \tag{18}$$

which clearly results into:

$$\begin{bmatrix} \boldsymbol{\delta} \\ \Delta\boldsymbol{\delta} \end{bmatrix} = \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \mathbf{I} \right] \boldsymbol{\epsilon} \tag{19}$$

The posterior error is fully compensated at steady state only when the $\boldsymbol{\delta}$ equals the prior error, $\boldsymbol{\delta} = \boldsymbol{\epsilon}$, given that (19) guarantees zero constant posterior error $\mathbf{e} = \boldsymbol{\delta} - \boldsymbol{\epsilon}$ at steady state.

Therefore, the filter is capable of compensating any amount of constant prior error, as long as the system in (17) is stable, which can easily be attained by tuning ω and K accordingly.

4.3. Practical Nonlinear Model Predictive Control

While using the full nonlinear model yields reliable responses for the calculated optimal control action, NLPs tend to be computationally costly to solve. One possible solution to ease the complexity of the MPC problem, which might be convenient for real-time applications, is through Taylor's series linearization of the nonlinear model. The Practical Nonlinear Model Predictive Control (PNMPC) (Plucênio et al., 2007) is an MPC strategy developed with this simplification in mind. The filter introduced in the previous section is used here and the model is approximated into a linear counterpart so that, instead of an NLP, the problem to be solved at each time step is a Quadratic Programming (QP) problem, which is much less computationally expensive.

Consider the generic discrete-time nonlinear system, such as the ESN, placed into MPC terms:

$$\mathbf{x}[k+i] = \mathbf{f}(\mathbf{x}[k+i-1], \mathbf{u}[k+i-1]) \tag{20a}$$

$$\mathbf{y}[k+i] = \mathbf{g}(\mathbf{x}[k+i]) \tag{20b}$$

$$\mathbf{u}[k+i-1] = \mathbf{u}[k-1] + \sum_{j=0}^{i-1} \Delta\mathbf{u}[k+j] \tag{20c}$$

Through a Taylor's series approximation, the system can be expressed as:

$$\widehat{\mathbf{Y}} = \mathbf{G} \cdot \boldsymbol{\Delta}\mathbf{U} + \mathbf{F} \tag{21}$$

where $\mathbf{F}$ is the free response and $\mathbf{G}$ is the sensitivity matrix that models the forced response, which have the following forms:

$$\mathbf{F} = \begin{pmatrix} \mathbf{g}(\mathbf{f}(\mathbf{x}[k], \mathbf{u}[k-1])) \\ \mathbf{g}(\mathbf{f}(\mathbf{x}[k+1], \mathbf{u}[k-1])) \\ \vdots \\ \mathbf{g}(\mathbf{f}(\mathbf{x}[k+N_y-1], \mathbf{u}[k-1])) \end{pmatrix} + \mathbf{1} \otimes \boldsymbol{\delta}[k] \tag{22a}$$

$$\mathbf{G} = \begin{pmatrix} \frac{\partial \mathbf{y}[k+1]}{\partial \mathbf{u}[k]} & 0 & \cdots & 0 \\ \frac{\partial \mathbf{y}[k+2]}{\partial \mathbf{u}[k]} & \frac{\partial \mathbf{y}[k+2]}{\partial \mathbf{u}[k+1]} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}[k+N_y]}{\partial \mathbf{u}[k]} & \frac{\partial \mathbf{y}[k+N_y]}{\partial \mathbf{u}[k+1]} & \cdots & \frac{\partial \mathbf{y}[k+N_y]}{\partial \mathbf{u}[k+N_u-1]} \end{pmatrix} \tag{22b}$$

the free response $\mathbf{F}$ is corrected according to the correction factor $\boldsymbol{\delta}$, which is in turn calculated by filtering the prediction error.

Notice that the system is linearized only with respect to the input. Therefore, while $\mathbf{F}$ is computed nonlinearly by assuming that the control action $\mathbf{u}[k-1]$ remains constant, the forced response is computed by multiplying $\mathbf{G}$, a sensitivity matrix containing the system's Jacobians, with the control increment vector $\Delta\mathbf{U}$ along the horizon, which is what reduces the MPC problem to a QP.

The calculation of $\mathbf{F}$ is straightforward, as it merely requires direct function evaluation and the correction factor calculation. However, the calculation of $\mathbf{G}$ is another matter. Early works assumed unattainability of the Jacobians (Plucênio et al., 2007) and employed a finite-difference scheme, which could lead to high computational complexity for multivariate systems. In this work, where a known state equation model (the ESN) is available, the controller employs the following recursive strategy by using the chain rule:

$$\frac{\partial \mathbf{y}[k+i]}{\partial \Delta \mathbf{u}[k+j]} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}[k+i]} \frac{\partial \mathbf{x}[k+i]}{\partial \Delta \mathbf{u}[k+j]} \tag{23a}$$

$$\frac{\partial \mathbf{x}[k+i]}{\partial \Delta \mathbf{u}[k+j]} = \frac{\partial \mathbf{f}}{\partial \Delta \mathbf{u}[k+j]} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}[k+i-1]} \frac{\partial \mathbf{x}[k+i-1]}{\partial \Delta \mathbf{u}[k+j]} \tag{23b}$$

Then, the computation of each block-element in $\mathbf{G}$, with row i and column j, is as follows:

$$\mathbf{G}_{ij} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}[k+i]} \frac{\partial \mathbf{x}[k+i]}{\partial \boldsymbol{\Delta} \mathbf{u}[k+j]} \tag{24a}$$

$$\frac{\partial \mathbf{x}[k+i]}{\partial \boldsymbol{\Delta} \mathbf{u}[k+j]} = \begin{cases} \mathbf{B}(i-1) + \mathbf{A}(i-1) \frac{\partial \mathbf{x}[k+i-1]}{\partial \boldsymbol{\Delta} \mathbf{u}[k+j]} & i > j \\ \mathbf{B}(i-1) & i = j \\ 0 & i < j \end{cases} \tag{24b}$$

where $\mathbf{B}(i-1)$ is the Jacobian of $\mathbf{f}$ at state $\mathbf{x}[k+i-1]$ with respect to $\mathbf{u}$, and $\mathbf{A}(i-1)$ is the Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}[k+i-1]$.

The computation is done per row i, where row $i+1$ depends on values for row i, and $i=1$ only calculates $\mathbf{B}(0)$, which is the Jacobian $\partial_{\mathbf{u}[k-1]}\mathbf{f}$ about $\mathbf{x}[k]$, for the first element.

Given an ESN, the respective derivatives to calculate the Jacobian are:

$$\frac{\partial \mathrm{g}}{\partial \mathbf{x}} = \mathbf{W}^{\mathbf{out}} \tag{25a}$$

$$\mathbf{J}(i) = \frac{\partial \mathbf{f}}{\partial \mathbf{z}_i}\mathbf{W}^{\mathbf{in}} \tag{25b}$$

$$\mathbf{S}(i) = (1-\gamma)\mathbf{I} + \gamma\frac{\partial \mathbf{f}}{\partial \mathbf{z}_i}(\mathbf{W}^{\mathbf{r}}) \tag{25c}$$

$$\mathbf{z}_i = \mathbf{W}^{\mathbf{r}}\mathbf{x}[k+i] + \mathbf{W}^{\mathbf{in}}\mathbf{u}[k-1] + \mathbf{b} \tag{25d}$$

By applying (21) to the optimization problem (5), the resulting problem becomes the QP:

$$\min_{\mathbf{\Delta U}} \quad J(\mathbf{\Delta U}) = \mathbf{\Delta U}^T\mathbf{H}\mathbf{\Delta U} + \mathbf{c}^T\mathbf{\Delta U} \tag{26a}$$

$$\text{s.t.}: \quad \mathbf{I}\mathbf{\Delta U} \leq \mathbf{\Delta U}_{\max} \tag{26b}$$

$$-\mathbf{I}\mathbf{\Delta U} \leq -\mathbf{\Delta U}_{\min} \tag{26c}$$

$$\mathbf{T}\mathbf{\Delta U} \leq \mathbf{1}\mathbf{u}_{\max} - \mathbf{1}\mathbf{u}[k-1] \tag{26d}$$

$$-\mathbf{T}\mathbf{\Delta U} \leq -\mathbf{1}\mathbf{u}_{\min} + \mathbf{1}\mathbf{u}[k-1] \tag{26e}$$

$$\mathbf{G}\Delta\mathbf{U} \leq \mathbf{1} \otimes \mathbf{y}_{\max} - \mathbf{F} \tag{26f}$$

$$-\mathbf{G}\Delta\mathbf{U} \leq \mathbf{F} - \mathbf{1} \otimes \mathbf{y}_{\min} \tag{26g}$$

where:

$$\mathbf{H} = \mathbf{G}^T\mathbf{Q}\mathbf{G} + \mathbf{R} \tag{27a}$$

$$\mathbf{c} = 2\mathbf{G}^T\mathbf{Q}^T(\mathbf{Y}_{ref} - \mathbf{F}) \tag{27b}$$

In this formulation, a rate-limiting constraint is added so that the Taylor approximation prediction does not lose precision, as the linear approximation is almost an equivalence for small control increments.

With this, the PNMPC successfully reduces the NMPC problem to a mere QP, while being quite robust (Plucênio, 2013).

5. Experiments and Results

This section showcases the experiments regarding the control of an ESP-lifted oil well with the Single-Shooting NMPC and the ESN-based PNMPC presented above. A discussion on the results obtained from these experiments follows.

5.1. Implementation

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 and 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.

5.2. Metrics and Experimental Setup

The experiments for the control of the ESP-lifted oil well, with an ESN as the prediction model, are divided into two phases: (i) the identification of the ESP using the ESN as a model; and (ii) a comparative study of each method applied to the MPC problem of the ESP-lifted oil well.

For the identification part: first, a dataset is obtained from a single simulation run of the system, which is then divided into a training/cross-validation phase and a testing phase. The excitation signal must be able to capture the system behavior at the working operation zone (Nelles, 2001), being a critical part of the system identification procedure. Afterwards, the hyperparameters are set so that the ESN achieves sufficiently good performance while taking into account the computational cost. The key metric to evaluate the ESN performance at a given dataset is the Normalized Root Mean Square Error (NRMSE) defined as:

$$NRMSE = \sqrt{\frac{1}{N}\sum_{k=0}^{N}\left\|\frac{\mathbf{y}[k]-\mathbf{y}_{\text{ESN}}[k]}{\mathbf{y}_{\max}-\mathbf{y}_{\min}}\right\|^2} \tag{28}$$

where $\mathbf{y}[k]$ is the plant output and $\mathbf{y}_{\text{ESN}}[k]$ is the ESN predicted output at time k, $\mathbf{y}_{\min}$ and $\mathbf{y}_{\max}$ define the bounds on outputs, and N is the number of samples. As the ESN is normalized using the maximum and the minimum point in the dataset (and also compared against normalized data), the denominator for (28) can be dropped.

For the next step, the two MPC methods (Direct SS NMPC and PNMPC) are applied to the control of the ESP. As the two methods solve essentially the same optimization problem, they are tested under the same parameters for $\mathbf{Q}$ and $\mathbf{R}$, along the same horizon. Their performance is compared through the IAE (Integral Absolute Error) of each variable:

$$IAE = \sum_{k=0}^{N} |y[k] - y_{\text{ref}}| \tag{29}$$

where y is the variable of interest (an element of the output vector $\mathbf{y}$). The IAE serves as a metric on how well a controller performs within a given simulation. Being a sum of absolute errors, the IAE has a high magnitude and, as such, it is widely used for comparison between different controllers.

With respect to a single variable, the control variation serves as a measure of the controller conservativeness:

$$\Delta u_{\text{tot}} = \sum_{k=0}^{N} |\Delta u[k]| \tag{30}$$

The higher the total control variation, the higher the energy spent by the controller in changing the manipulated variable.

5.3. Identification of ESP-lifted Oil Wells

The first step toward model identification regards the design of the excitation signal which, for this work, consists of an Amplitude-modulated Pseudo-Random Binary Signal (APRBS) (Nelles, 2001), which is a stair signal with values drawn from the uniform distribution. The APRBS signal has a minimum period of 5 time steps ($5/12$ s) at the first half of the dataset, and a minimum period of 40 ($40/12$ s) steps for the rest of the simulation. The signal amplitude obeys the upper and lower bound for both frequency f and choke opening z. By following this variation in the control frequency, 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 4 showcases the APRBS excitation input signal used to generate the dataset. The signal is $12,000$ time steps ($1,000$ s) long, with values ranging from $(z_{\min}, f_{\min}) = (0.1, 35)$ to $(z_{\max}, f_{\max}) = (1.0, 65)$ with respect to the choke

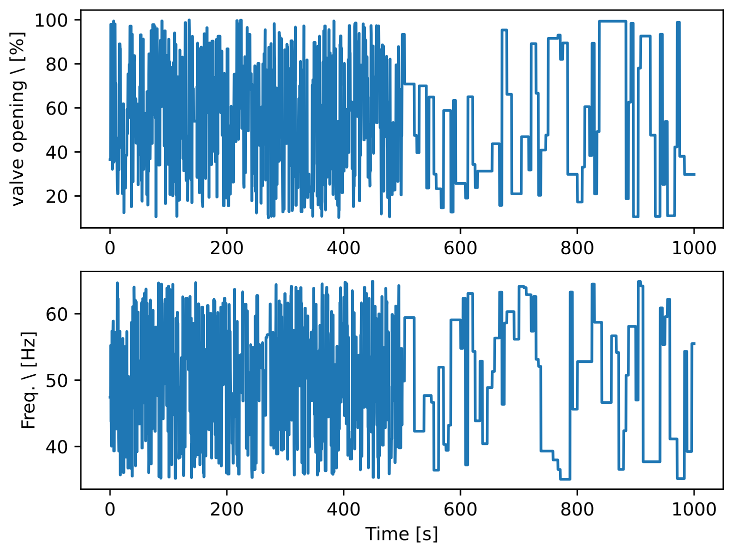

Figure 4: APRBS excitation signal utilized to generate the dataset by inputting it into the ESP-lifted well.

opening and pump frequency variables, respectively. The initial condition for the system was $p_{\text{bh}} = 70$ bar, $p_{\text{wh}} = 20$ bar, and $q = 36\ m^3/h$. Starting from these initial conditions, the output signal for p_{bh}, p_{wh} and q is obtained directly and deterministically by simulating the well model with the input sequence in Figure 4, which is not shown for simplicity reasons. Thus, the 2-dimensional excitation input signal along with the desired 3-dimensional output signal form the training dataset for the identification task. An APRBS signal of 300 time steps was used as an excitation signal for a validation set, distinct from the training dataset shown in Figure 4.

Table 2: 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

Table 2 showcases the results from an experiment that aimed to assess the impact of the number of neurons on ESN performance. The performance improvement achieved in ESNs with more than 300 neurons was not significant, while the computation time increased almost linearly, which led us to select 300 reservoir states for the echo state network.

Table 3: 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 3. 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, we trained a 128-unit Long Short-Term Memory (LSTM) and a 64 Gated Recurrent Unit (GRU) network, using the same training data as the ESN, to com-

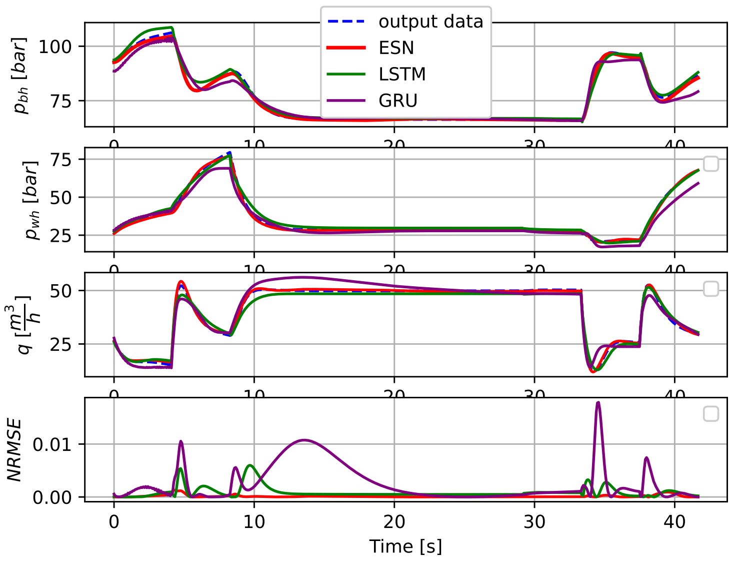

Figure 5: Prediction of the ESN, LSTM, and GRU on test data in comparison to the ESP plant (blue dashed line) given the test excitation signal. The first plot presents the bottom-hole pressure p_{bh}, the second plot, the wellhead pressure p_{wh}, and the third plot, the flow. The fourth plot is the NRMSE at the given time-step.

pare their performance against the ESN in identifying the ESP-lifted oil well. By means of a grid grid-search, the number of cells for the LSTM and GRU were chosen so that each network achieves satisfactory performance on identifying the ESP-lifted 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.

The test data consists of 1000 time steps. Figure 5 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 5. Also, Table 4 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 4: 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

5.4. ESN-based MPC for ESP-lifted Oil Wells

This section showcases control problems involving the ESP-lifted oil well. Each problem regards different settings for controlled variables, which must track a reference signal, and settings for manipulated variables. Both the Single-Shooting ESN-NMPC and the ESN-PNMPC solve each proposed problem, with precisely the same parameter values.

The problems are defined as follows:

1. With the choke opening fixed at $z = 1$ (fully open), the pump frequency f is manipulated to track the reference signals for the well bottom-hole pressure (p_{bh}), while adhering to the constraints imposed by the problem defined in (26). We also perform a disturbance rejection experiment in the reservoir pressure (a variation of $\Delta p_r = -10$ bar from the original parameter value) for this specific control problem, to demonstrate the performance of the error-correction filter.
2. Both the choke opening z and pump frequency f serve as manipulated variables. This time, 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, we set a constant reference signal for q at a sufficiently high level for the system in a way that, although unreachable, this reference causes the controller to maximize production within the same reference-tracking MPC setting defined in (26).

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.

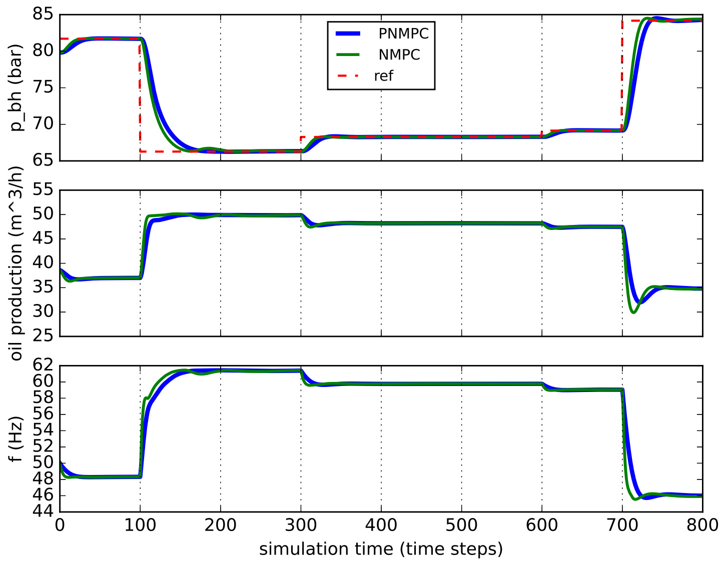

Figure 6: Results of the experiment for the bottom-hole pressure tracking induced by both the SS ESN-NMPC and the ESN-PNMPC. The first plot regards the bottom-hole pressure, the second plot shows the oil flow q, and the third plot presents the manipulated variable: the pump frequency f.

5.4.1. Bottom-hole pressure tracking with pump frequency manipulation

We now present the results from the experiments regarding the tracking of the bottom-hole pressure p_{bh} by manipulating only the pump frequency f. For this problem, the parameters set for each controller were:

$$\mathbf{Q} = 0.7 \tag{31a}$$

$$\mathbf{R} = 5 \tag{31b}$$

$$K = 1/3 \tag{31c}$$

$$\omega = 0.25 \tag{31d}$$

$$\Delta U_{\max} = -\Delta U_{\min} = 0.2 \tag{31e}$$

Figure 6 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 5: 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**

Table 5 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 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, we computed the relative error between the control action of each controller RE_{MPC} (the solution to their respective optimization problems):

$$RE_{MPC} = \sum_{k=1}^{N} \left| \frac{u_{PNMPC}[k] - u_{NMPC}[k]}{u_{NMPC}[k]} \right| = 0.06, \tag{32}$$

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 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.

We also performed a disturbance rejection experiment on this control problem, by injecting a disturbance Δp_r that decreases the reservoir pressure in 10

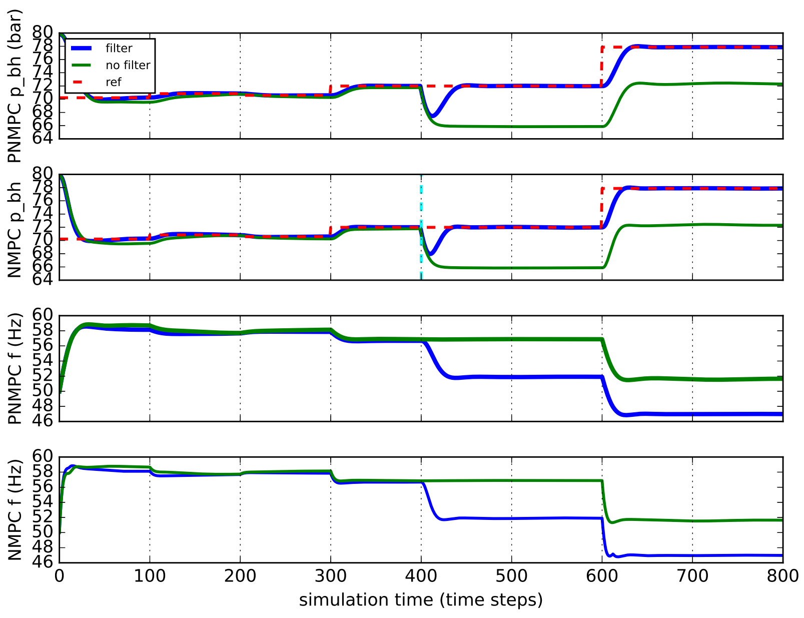

Figure 7: Results of the experiment for the bottom-hole pressure tracking with disturbance rejection induced by both the SS ESN-NMPC and ESN-PNMPC. The first and the second plots show the bottom-hole pressure tracking by PNMPC and NMPC, respectively. Both plots have a blue, thick line representing the 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.

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 7 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.

5.4.2. 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:

$$\mathbf{Q} = \begin{pmatrix} 0.7 & 0 \\ 0 & 0.1 \end{pmatrix} \tag{33a}$$

$$\mathbf{R} = \begin{pmatrix} 3 & 0 \\ 0 & 5 \end{pmatrix} \tag{33b}$$

$$K = 1/3 \tag{33c}$$

$$\omega = 0.25 \tag{33d}$$

$$\Delta U_{\max} = -\Delta U_{\min} = 0.2\mathbf{1} \tag{33e}$$

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 8 showcases the results of the experiments. We see that 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's control action varies more than NMPC's one. The NMPC seems to prioritize production more than the bottom-hole pressure, which explains the conservativeness of the generated control action. Table 6 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 6: Metrics for each controller in the problem of pressure tracking with target production.

	ESN-NMPC	ESN-PNMPC
IAE (p_{bh}) [bar]	1724.65	**870.78**
IAE (q) [m^3/h]	**7222.01**	8106.24
Δu_{tot} (z)	**0.37**	0.95
Δu_{tot} (f) [Hz]	**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 $RE_{MPC}(f) = 11.04$.

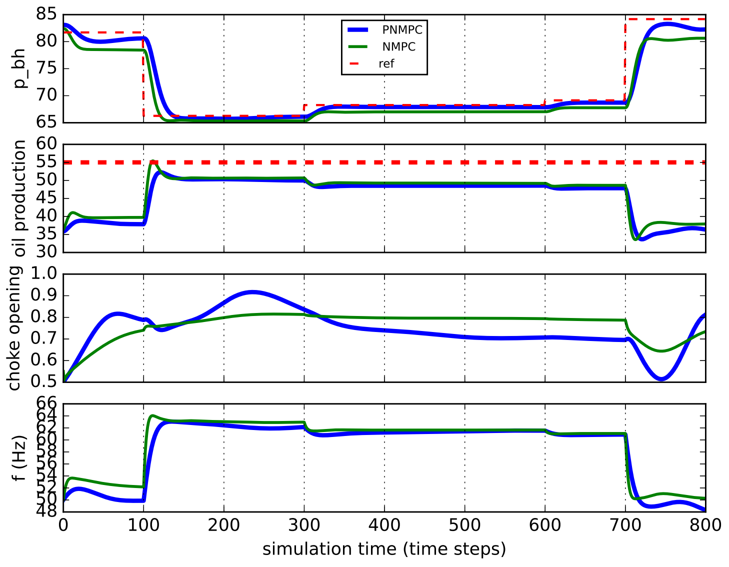

Figure 8: Results for the bottom-hole pressure tracking while maximizing oil production for both the ESN-NMPC and the ESN-PNMPC. The first plot shows the controlled bottom-hole pressure. The second plot shows the oil flow q with the target oil production given by the horizontal dashed red line. The bottom plots present the manipulated variables choke opening z, and pump frequency f.

There is a significant difference between each controller because we are purposefully driving the system 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.

6. Conclusion

This work has shown the successful application of ESN-based NMPC to the control of an ESP-lifted well characterized by a system of differential equations. The ESN was able to capture the patterns of the well with high accuracy and therefore was embedded in the proposed MPC strategies as the prediction model. In conjunction with an error correction filter, both Single-Shooting NMPC and PNMPC achieved promising results with the ESN for the application at hand. We conclude that the capacity of the ESN to learn to reproduce the nonlinear dynamics from unknown plants in a data-driven way is an ideal and pragmatic match to MPC strategies for control.

In tracking control experiments, PNMPC achieved a performance approaching that of NMPC, but at a significantly lower computational complexity, which is remarkable given that the latter relies on advanced software with automatic differentiation and an NLP solver, whereas the former uses a QP solver and a simple recursive algorithm to compute sensitivities. We recommend the ESN-PNMPC approach especially if control action calculation must be fast.

The ESN needs a large number of states to correctly represent a plant by training only the output weights. This makes the sensitivities (gradient) computation costly due to function composition. Normally, by defining the dynamic system state as decision variables and the system equations as constraints (Multiple-Shooting), one would mitigate the cost of computing gradients. However, multiple shooting would entail a large number of decision variables and constraints for ESN. This motivates future research on how to efficiently achieve practical Multiple-Shooting NMPC with ESNs. Future work also includes the application of proposed approaches into a real-world ESP-lifted well plant.

Acknowledgments

The authors acknowledge the support of CAPES-SIU (project #88881.153841/2017-01) and FAPESC under Edital 09/2019.

References

de A. Delou, P., de Azevedo, J.P.A., Krishnamoorthy, D., de Souza, M.B., Secchi, A.R., 2019. Model predictive control with adaptive strategy applied to an electric submersible pump in a subsea environment. IFAC-PapersOnLine 52, 784–789. doi:10.1016/j.ifacol.2019.06.157.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M., 2019. CasADi – A software framework for nonlinear optimization and optimal control. Mathematical Programming Computation 11, 1–36. doi:10.1007/s12532-018-0139-4.

Antonelo, E.A., Schrauwen, B., 2015. On learning navigation behaviors for small mobile robots with reservoir computing architectures. IEEE Transactions on Neural Networks and Learning Systems 26, 763–780. doi:10.1109/TNNLS.2014.2323247.

Armenio, L.B., Terzi, E., Farina, M., Scattolini, R., 2019. Model predictive control design for dynamical systems learned by echo state networks. IEEE Control Systems Letters 3, 1044–1049. doi:10.1109/LCSYS.2019.2920720.

Ayala, H.V.H., Habineza, D., Rakotondrabe, M., dos Santos Coelho, L., 2020. Nonlinear black-box system identification through coevolutionary algorithms and radial basis function artificial neural networks. Applied Soft Computing 87, 105990. doi:10.1016/j.asoc.2019.105990.

Binder, B.J.T., Kufoalor, D.K.M., Pavlov, A., Johansen, T.A., 2014. Embedded model predictive control for an electric submersible pump on a programmable logic controller, in: IEEE Conference on Control Applications (CCA), IEEE. pp. 579–585. doi:10.1109/CCA.2014.6981402.

Binder, B.J.T., Pavlov, A., Johansen, T.A., 2015. Estimation of flow rate and viscosity in a well with an electric submersible pump using moving horizon estimation. IFAC-PapersOnLine 48, 140–146. doi:10.1016/j.ifacol.2015.08.022.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc.

Bock, H.G., Plitt, K.J., 1984. A multiple shooting algorithm for direct solution of optimal control problems. IFAC Proceedings Volumes 17, 1603–1608. doi:10.1016/S1474-6670(17)61205-9.

Camacho, E., Bordons, C., 1999. Model Predictive Control. Springer.

Cao, Y., Huang, J., 2020. Neural-network-based nonlinear model predictive tracking control of a pneumatic muscle actuator-driven exoskeleton. IEEE/CAA

Journal of Automatica Sinica 7, 1478–1488. doi:10.1109/JAS.2020.1003351.

Centrilift, B.H. (Ed.), 2008. Centrilift Europe and Africa ESP Failures 1999-2008.

Chang, T.K., Yu, D.L., Yu, D.W., 2004. Neural network model adaptation and its application to process control. Advanced Engineering Informatics 18, 1–8. doi:10.1016/j.aei.2004.01.001.

Chen, C., Liu, H., 2021. Dynamic ensemble wind speed prediction model based on hybrid deep reinforcement learning. Advanced Engineering Informatics 48, 101290. doi:10.1016/j.aei.2021.101290.

Dahl, J., Vandenberghe, L., 2020. CVXOPT: a python package for convex optimization. URL: http://abel.ee.ucla.edu/cvxopt/.

Gros, S., Zanon, M., Quirynen, R., Bemporad, A., Diehl, M., 2016. From linear to nonlinear MPC: bridging the gap via the real-time iteration. International Journal of Control 93, 62–80. doi:10.1080/00207179.2016.1222553.

Hinaut, X., Dominey, P.F., 2012. On-line processing of grammatical structure using reservoir computing, in: Int. Conf. on Artificial Neural Networks, pp. 596–603. doi:10.1007/978-3-642-33269-2_75.

Jaeger, H., 2001. The ‘‘echo state’’ approach to analysing and training recurrent neural networks - with an Erratum note. Technical Report GMD 148. German National Research Center for Information Technology.

Jaeger, H., Haas, H., 2004. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. Science 304, 78–80. doi:10.1126/science.1091277.

Jaeger, H., Lukosevicius, M., Popovici, D., Siewert, U., 2007. Optimization and applications of echo state networks with leaky-integrator neurons. Neural Networks 20, 335–352. doi:10.1016/j.neunet.2007.04.016.

Jahn, F., Cook, M., Graham, M., 2008. Hydrocarbon Exploration and Production. Developments in Petroleum Science. 2nd edition ed., Elsevier.

Jordanou, J.P., Camponogara, E., Antonelo, E.A., de Aguiar, M.A.S., 2018. Nonlinear model predictive control of an oil well with echo state networks. IFAC-PapersOnLine 51, 13–18. doi:10.1016/j.ifacol.2018.06.348.

Ławryńczuk, M., 2014. Computationally Efficient Model Predictive Control Algorithms. Springer International Publishing.

Lin, X., Yang, Z., Song, Y., 2009. Short-term stock price prediction based on echo state networks. Expert Systems with Applications 36, 7313–7317. doi:10.1016/j.eswa.2008.09.049.

Lukoševičius, M., Marozas, V., 2014. Noninvasive fetal QRS detection using an echo state network and dynamic programming. Physiological Measurement 35, 1685–1697. doi:10.1088/0967-3334/35/7/1685.

Mehrez, M.W., 2019. Optimization based solutions for control and state estimation in dynamical systems (implementation to mobile robots) a workshop. doi:10.13140/RG.2.2.21613.23521.

Mozer, M.C., 1995. Backpropagation: Theory, architecture, and applications, L. Erlbaum Associates Inc., Hillsdale, NJ, USA. chapter A focused backpropagation algorithm for temporal pattern recognition, pp. 137–169.

Nelles, O., 2001. Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models. 1 ed., Springer-Verlag Berlin Heidelberg.

Ouali, M.A., Ladjal, M., 2020. Nonlinear dynamical systems modelling and identification using type-2 fuzzy logic. Metaheuristic algorithms based approach, in: International Conference on Electrical Engineering (ICEE), pp. 1–6. doi:10.1109/ICEE49691.2020.9249916.

Pan, Y., Wang, J., 2012. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. IEEE Transactions on Industrial Electronics 59, 3089–3101. doi:10.1109/TIE.2011.2169636.

Pavlov, A., Krishnamoorthy, D., Fjalestad, K., Aske, E., Fredriksen, M., 2014. Modelling and model predictive control of oil wells with electric submersible pumps, in: IEEE Conference on Control Applications (CCA), pp. 586–592. doi:10.1109/CCA.2014.6981403.

Plucênio, A., 2013. Development of Non Linear Control Techniques for the Lifting of Multiphase Fluids. Ph.D. thesis. Federal University of Santa Catarina, Brazil.

Plucênio, A., Pagano, D., Bruciapaglia, A., Normey-Rico, J., 2007. A practical approach to predictive control for nonlinear processes. IFAC Proceedings Volumes 40, 210–215. doi:10.3182/20070822-3-ZA-2920.00035.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., Valaee, S., 2017. Recent advances in recurrent neural networks. arXiv:1801.01078.

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural Networks 61, 85–117. doi:10.1016/j.neunet.2014.09.003.

Schrauwen, B., Verstraeten, D., Van Campenhout, J., 2007. An overview of reservoir computing: theory, applications and implementations, in: Proceedings of the 15th European Symposium on Artificial Neural Networks, pp. 471–482.

Takacs, G., 2017. Electrical Submersible Pumps Manual: Design, Operations, and Maintenance. San Diego: Elsevier Science.

Terzi, E., Bonetti, T., Saccani, D., Farina, M., Fagiano, L., Scattolini, R., 2020. Learning-based predictive control of the cooling system of a large business centre. Control Engineering Practice 97, 104348. doi:10.1016/j.conengprac.2020.104348.

Verstraeten, D., Dambre, J., Dutoit, X., Schrauwen, B., 2010. Memory versus non-linearity in reservoirs, in: Int. Joint Conference on Neural Networks, pp. 18–23. doi:10.1109/IJCNN.2010.5596492.

Xiang, K., Li, B.N., Zhang, L., Pang, M., Wang, M., Li, X., 2016. Regularized Taylor echo state networks for predictive control of partially observed systems. IEEE Access 4, 3300–3309. doi:10.1109/ACCESS.2016.2582478.

Yang, Y., Zhou, L., Han, Y., Hang, J., Lv, W., Shi, W., He, Z., Pan, B., 2021a. Pressure pulsation investigation in an electrical submersible pump based on morlet continuous wavelet transform. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science doi:10.1177/0954406221100077.

Yang, Y., Zhou, L., Shi, W., He, Z., Han, Y., Xiao, Y., 2021b. Interstage difference of pressure pulsation in a three-stage electrical submersible pump. Journal of Petroleum Science and Engineering 196, 107653. doi:10.1016/j.petrol.2020.107653.