Echo State Networks for Practical Nonlinear Model Predictive Control of Unknown Dynamic Systems

Jean Panaioti Jordanou, Eric Aislan Antonelo, Eduardo Camponogara

(Paper submitted for the Special Issue on New Frontiers in Extremely Efficient Reservoir Computing)

***Abstract*—Nonlinear Model Predictive Control (NMPC) of industrial processes is changeling in part because the model of the plant may not be completely known, but also for being computationally demanding. This work proposes an extremely efficient Reservoir Computing (RC)-based control framework that speeds up the NMPC of processes. In this framework, while an Echo State Network (ESN) serves as the dynamic RC-based system model of a process, the Practical Nonlinear Model Predictive Controller (PNMPC) simplifies NMPC by splitting the forced and the free responses of the trained ESN, yielding the so called ESN-PNMPC architecture. While the free response is generated by forward simulation of the ESN model, the forced response is obtained by a fast and recursive calculation of the input-output sensitivities from the ESN. The efficiency not only results from the fast training inherited by RC, but also from a computationally cheap control action given by the aforementioned novel recursive formulation and the computation in the reduced dimension space of input and output signals. The resulting architecture, equipped with a correction filter, is robust to unforeseen disturbances. The potential of the ESN-PNMPC is shown by application to the control of the four-tank system and an oil production platform, outperforming the predictive approach with an LSTM (Long-Short Term Memory) model, two standard linear control algorithms and approximate predictive control.**

***Index Terms*—Model Predictive Control, Reservoir Computing, Echo State Networks.**

I. Introduction

Effectively controlling complex industrial plants whose models are initially unknown constitutes a challenging task. Indeed, such cases are very common in the real world, demanding efficient data-driven schemes to control such plants. Usually, an approximate (black-box) model of an unknown plant is found by a *system identification* procedure [1].

Out of many possible control methods from the literature that can be applied to systems without known models [2], model predictive control (MPC) is one technique that has become standard for multivariate control in industry and academia [3]. Since its inception in the 1970s, MPC was successfully applied in the oil and gas [4], aerospace [5] and process industries, as well as in robotics [6]. MPC has the advantage of simplicity and intuitiveness over other approaches, as little knowledge about control theory is required to apply the method in an industrial level [3]. 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) [3]. 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.

Recurrent Neural Networks (RNN) are considered universal approximators of dynamical systems [7], 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 [8] and is computationally costly.

An alternative way to model dynamic systems is by Reservoir Computing (RC) [9], [10]. The basic assumption 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. 1). 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 Echo State Network (ESN) [11], [10]. This linear training also enables the model to be updated online with new data through algorithms such as Recursive Least Squares (RLS) [12].

Other works use Radial Basis Function (RBF) networks [13] or fuzzy logic [14] 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

The authors thank the financial support from CAPES under the PrInt CAPES-UFSC grant #698503P, FAPESC under Edital 09/2019, and a post-doctoral scholarship at DIN-UEM.

J. Jordanou, E. Antonelo, and E. Camponogara are with the Department of Automation and Systems Engineering, Federal University of Santa Catarina.

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 [15] 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 [16], 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 [16], the computational and memory demands are high when compared to our proposal based on PNMPC. PMNPC was shown to achieve good performance in several applications, such as in the control of oil and gas processes [17], 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, our work proposes an extremely efficient 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.

A. Contributions

The contributions of this new ESN-PNMPC method are:

1) a novel recursive formulation for fast calculation of sensitivities from the RC network, which yields a Jacobian matrix that directly relates output changes with control-input variation, unlike other works [16] that explicitly model the states of the RC network in the control algorithm, which is usually high-dimensional;
2) an extremely efficient control method based on RC nets, that results from the fast training inherited by RC combined with a computationally cheap control action, yielded by the aforementioned recursive formulation and computation in the reduced dimension space of input and output signals from the RC net;
3) a new methodology for tuning the parameters of a correction filter that achieves robustness to unpredictable disturbances;
4) a demonstration of the proposed ESN-PNMPC architecture in the control of two representative dynamic systems: the well-known four-tank system and an oil production platform consisting of two wells and one riser [12], in addition to comparisons with a PI controller, a linear MPC, and PNMPC with a LSTM model.

This paper extends considerably on our previously published conference work [4], specifically items 3) and 4) of the above contributions are completely new, including new extensive control simulations and analysis as well as experiments on the effect of the RC net hyperparameters on control performance.

II. Related Work

In [16], an ESN is trained to serve as the model in MPC, akin to our work. However, instead of separating the system into a nonlinear free response and a linear forced response obtained by calculating the sensitivity to the input, the whole ESN is approximated into a state space system for computation of the control action. The large number of states in the ESN results in the QP itself being approximated for that work, which leads to a more computational and memory demanding program compared to our work. Besides, their method does not allow to retain full precision of the state progression in the prediction horizon of MPC, as the state is also computed as a linear approximation. In contrast to [16] which uses a time-variant static gain over the error, our proposed ESN-PNMPC scheme filters the error, integrating it into a correction factor, which ensures zero steady-state prediction error [3], [15]. Besides, our work is applied to representative dynamic plants in industry and benchmark control systems, and not only to purely mathematical systems as in [16]. Another work [18] employs a Taylor linearized Echo State Network as predictive model, while using the trained ESN as an observer [19]. However, the controller itself works only for a predefined operating point of the linearized ESN, which limits its flexibility and applicability.

A more recent work [20] analyses the stability properties of an ESN-based NMPC. A quadratic cost function is considered, however their approach differs from ours in that the optimization problem is solved using the full nonlinear model, resulting in a more computationally expensive controller. Also, instead of integrating and filtering the prediction error, a nonlinear Luenberger observer term is applied. There are other works combining ESN and MPC, such as [21]. However, all of these methods differ from the current work in that they do not apply the Taylor expansion to compute sensitivity (forced response) neither do they design an error correction filter, both of which are going to be elaborated upon in the next section.

Other works employing ESNs for control of nonlinear plants using methods different from MPC include [22], [12]. For instance, [12] uses an architecture with two RC networks that learns online to simultaneously identify and control the dynamic system. As it does not use MPC, the control is only of regulatory nature, not allowing constraints to be added. The work [22] proposes a Backstepping controller that uses an ESN as the model and includes an error correction term. Despite showing promising results, such nonlinear control methods are highly dependent on model accuracy. In [23], a variation of ESNs for position-tracking is proposed that combines a Fuzzy inference scheme and a wavelet activation function for error correction in a Sliding Mode Control strategy.

With respect to NMPC with model linearization, [24] compares nonlinear to linear MPC, and introduces the concept

of Real Time Iteration (RTI), which consists in computing the NMPC solution with a single iteration of Sequential Quadratic Programming (SQP). Also, [24] argues how similar this strategy is to a linear MPC, if the QP sensitivities are obtained offline. The logic behind RTI is similar to PNMPC, however RTI reduces the nonlinear system into a linear state-space system, whereas PNMPC uses the output-to-input sensitivity to calculate the control action. Similar to the RTI framework in employing NMPC with successive linearizations, [25] adds a LSTM as the proxy model for the MPC of the cooling system of a large business center. The LSTM approach achieved a slightly lower test error for their problem, but it is harder to train than an ESN. Like other works based on linearization for NMPC, [25] reduces the nonlinear system into a Linear Time-Varying (LTV) state-space system, which is the main difference to our approach which relies on the separation of the free and forced response. According to the reported results, the control signals are updated every 20 minutes to counter the long computation time of NMPC linearization with the LSTM. In contrast, owing to the efficient computation of sensitivities, the proposed ESN-PNMPC achieves fast calculation of the control action in a dynamic system of comparable complexity presented in our work.

In [26], several algorithms are presented for suboptimal NMPC with linearization, including some cases with recurrent neural networks. The most similar one to PNMPC is MPC with nonlinear prediction and linearization along the Trajectory (MPC-NPLT). In fact, PNMPC can be viewed as a special case of MPC-NPLT with respect to model linearization, where the forced response is calculated assuming fixed the latest control signal over the prediction horizon. Besides, PNMPC is a practical realization in terms of procedures to efficiently compute the required sensitivity matrix, while MPC-NPLT gives only a skeleton which is not directly realizable. Further, PNMPC implements a filter that measures the real plant output for error correction, granting robustness to unforeseen disturbances by ensuring zero error from bias. This feature is crucial when using an inexact representation of the plant, such as an RNN, thus countering modeling errors, alongside errors from Taylor approximation.

Another example of NMPC algorithms utilizing Neural Networks (NN) linearized online as a model is the Approximate Predictive Control (APC) [27], that assumes a feedforward NN with external dynamics. The algorithm reduces the NN into a discrete transfer function, and obtains the free and forced response based on such linearization at each time step, as per [3], in contrast to our ESN trajectory linearization approach. Both methods calculate the dynamic sensitivity matrix [3] recursively. While the ESN is usually higher-dimensional than a feedforward NN with external dynamics, the latter is more difficult to train depending on the complexity of the considered dynamical system, when compared to ESNs [10], [11], [9].

III. Methods

A. Echo State Networks (ESN)

An ESN is a type of recurrent neural network with useful characteristics for system identification [28], 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.

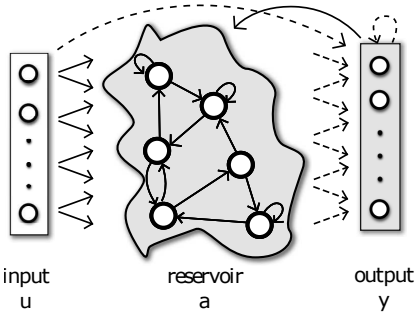


Fig. 1: Representation of an Echo State Network, one of the possible models in Reservoir Computing. Dashed connections (from Reservoir to Output Layer) are trainable, while solid connections are fixed and randomly initialized.

1) Model: Proposed in [10], [11], the ESN is governed by the following discrete-time dynamic equations:

$$\mathbf{a}[k+1] = (1-\gamma)\mathbf{a}[k] + \gamma f(\mathbf{W}_{\mathbf{r}}^{\mathbf{r}}\mathbf{a}[k] + \mathbf{W}_{\mathbf{i}}^{\mathbf{r}}\mathbf{u}[k] + \mathbf{W}_{\mathbf{b}}^{\mathbf{r}} + \mathbf{W}_{\mathbf{o}}^{\mathbf{r}}\mathbf{y}[k]) \quad (1)$$

$$\mathbf{y}[k+1] = \mathbf{W}_{\mathbf{r}}^{\mathbf{o}}\mathbf{a}[k+1], \quad (2)$$

where: the state of the reservoir neurons at time k is given by $\mathbf{a}[k]$; the current values of the input and output neurons are represented by $\mathbf{u}[k]$ and $\mathbf{y}[k]$, respectively; γ is called leak rate [28], which governs the percentage of the current state $\mathbf{a}[k]$ that is transferred into the next state $\mathbf{a}[k+1]$. The weights are represented in the notation $\mathbf{W}_{\mathbf{from}}^{\mathbf{to}}$, with "$\mathbf{b}$", "$\mathbf{o}$", "$\mathbf{r}$", and "$\mathbf{i}$" 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 [1]. Fig. 1 depicts a standard architecture of an echo state network.

The network has N neurons in the reservoir, which is the dimension of $\mathbf{a}[k]$ and is typically orders of magnitude higher than the number of network inputs. As long as regularization 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 (1). According to [29], the ESN with no output feedback connections (the output has no effect on the state), which is given by $\mathbf{W}_{\mathbf{o}}^{\mathbf{r}}$, 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) [11], 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 $\mathbf{W}_{\mathbf{r}}^{\mathbf{r}} < 1$. 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 [28]. However, reservoirs with $\rho > 1$ can still have the ESP since the effective spectral radius may still be lower than 1 [30], [31].
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 [32]. 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) Training: While in standard RNNs all weights are trained iteratively using gradient descent [33], 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, we 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 state matrix $\mathbf{A}$ that contains the resulting sequence of states. Then, we apply the Ridge Regression algorithm [34] by using $\mathbf{A}$ as the input data matrix and $\mathbf{D}$ as the output data matrix, or in this case a vector as we assumed single output. Ridge Regression results in solving the following linear system:

$$(\mathbf{A}^T\mathbf{A} - \lambda\mathbf{I})\mathbf{W_r^o} = \mathbf{A}^T\mathbf{D}, \tag{3}$$

where λ is the Tikhonov regularization parameter, which serves to penalize the weight magnitude, and avoid overfitting. There are also methods to apply least squares training in an online way [1], but these algorithms are not used in this work.

B. Practical Nonlinear Model Predictive Control (PNMPC)

Developed by [15], 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 an arbitrary differential function $h(u)$ with respect to an input u, we can better explain the intuition behind the PNMPC:

$$h(u) \approx h(\bar{u}) + \frac{\partial h(\bar{u})}{\partial u}\Delta u, \tag{4}$$

where $u = \bar{u} + \Delta u$, $\bar{u}$ is a fixed point, and Δu is a variation around that same point. The arbitrary nonlinear function $h(u)$ is split into: (i) a zeroth-order term $h(\bar{u})$ which is analogous to the PNMPC free response as it computes the current h 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 $h(\bar{u})$ will be simulated by running (1) 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. Assume a dynamic system in the form:

$$\mathbf{x}[k+i] = \mathbf{f}(\mathbf{x}[k+i-1], \mathbf{u}[k+i-1]) \tag{5}$$

$$\mathbf{y}[k+i] = \mathbf{g}(\mathbf{x}[k+i]) \tag{6}$$

$$\mathbf{u}[k+i-1] = \mathbf{u}[k-1] + \sum_{j=0}^{i} {}^{1}\Delta\mathbf{u}[k+j]. \tag{7}$$

In the context of this work, $\mathbf{f}$ models the state dynamics of an echo state network, as given by Eq. (1), and $\mathbf{g}$ is the output function defined by Eq. (2).

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

$$\widehat{\mathbf{Y}} = \mathbf{G}(\mathbf{x}[k], \mathbf{u}[k]) \cdot \mathbf{\Delta U} + \mathbf{F}(\mathbf{x}[k], \mathbf{u}[k]) + \mathcal{O}(\|\Delta\mathbf{U}\|^2), \tag{8}$$

where:

$$\mathbf{\Delta U} = \begin{pmatrix} \mathbf{\Delta u}[k] \\ \mathbf{\Delta u}[k+1] \\ \vdots \\ \mathbf{\Delta u}[k+N_u-1] \end{pmatrix}, \tag{9}$$

$$\mathbf{F} = \begin{pmatrix} \mathbf{g}(\mathbf{f}(\mathbf{x}[k], \mathbf{u}[k-1])) \\ \mathbf{g}(\mathbf{f}(\mathbf{x}[k+1], \mathbf{u}[k-1])) \\ \vdots \\ \mathbf{g}(\mathbf{f}(\mathbf{x}[k+N_y-1], \mathbf{u}[k-1])) \end{pmatrix}, \tag{10}$$

N_y is the prediction horizon, N_u is the control horizon, and the $\mathbf{G}$ matrix is the Jacobian of the state equations at the free response. The third term in (8) is the Taylor approximation error in Big-O notation, which showcases that as long as $\Delta\mathbf{U}$ is small, the approximation error will be small.

The vector $\mathbf{\Delta U}$ consists of the concatenation of each control increment applied to the calculation of the predictions

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

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:

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

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 [3].

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 [15] 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.

C. ESN-PNMPC

1) Introduction: The proposed ESN-PNMPC scheme consists of integrating an ESN as the state space model for PNMPC (see Fig. 2). 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.

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:

$$\frac{\partial \mathbf{y}[k+i]}{\partial \Delta\mathbf{u}[k+j]} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}[k+i]} \frac{\partial \mathbf{x}[k+i]}{\partial \Delta\mathbf{u}[k+j]} \tag{11}$$

$$\frac{\partial \mathbf{x}[k+i]}{\partial \Delta\mathbf{u}[k+j]} = \frac{\partial \mathbf{f}}{\partial \Delta\mathbf{u}[k+j]} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}[k+i-1]} \frac{\partial \mathbf{x}[k+i-1]}{\partial \Delta\mathbf{u}[k+j]}. \tag{12}$$

The implication in Equation (12) 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,

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

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. (7), 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. (11)-(12), the partial derivatives can recast in a recursive form:

$$\mathbf{G}_{ij} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}[k+i]} \frac{\partial \mathbf{x}[k+i]}{\partial \mathbf{\Delta u}[k+j]} \tag{13}$$

$$\frac{\partial \mathbf{x}[k+i]}{\partial \mathbf{\Delta u}[k+j]} = \begin{cases} \mathbf{J}(i \quad 1) + \mathbf{S}(i \quad 1) \frac{\partial \mathbf{x}[k+i-1]}{\partial \mathbf{\Delta u}[k+j]} & i > j \\ \mathbf{J}(i \quad 1) & i = j \\ 0 & i < j, \end{cases} \tag{14}$$

where $\mathbf{G}_{ij}$ represents the block element of $\mathbf{G}$ at row i and column j. The construction of $\mathbf{G}$ starts when $i = 1$, 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 $\mathbf{G}_{1,(j \leftarrow 1)}$ = 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 [16], [18], being

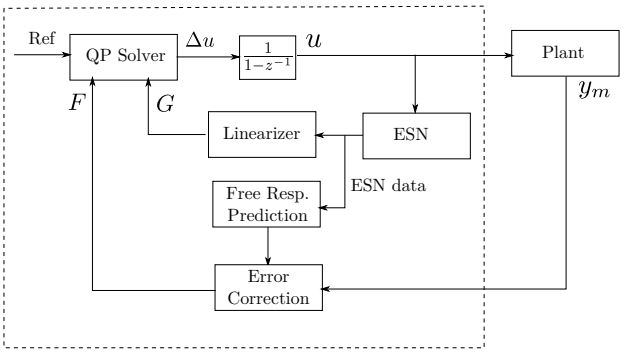


Fig. 2: Block diagram representation of the ESN-PNMPC. The ESN block represents an ESN trained to mimic the plant, which is used by the Linearizer block to compute $\mathbf{G}$ using the ESN Jacobian, and by the Free Response Prediction block. The Error Correction block provides an integrated filter that computes the correction factor, while the QP Solver block handles the resulting optimization problem.

given as follows:

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

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 [16], 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) Error Correction Filter: To treat disturbances and modeling errors, [15] 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 [3]. In practice, this is merely a different perspective to the problem, since the approach taken by [16] is equivalent to using a variable static gain as a filter.

Adding the filter, the free-response becomes:

$$\mathbf{F} = \begin{pmatrix} \mathbf{g}(\mathbf{f}(\mathbf{x}[k], \mathbf{u}[k \quad 1])) \\ \mathbf{g}(\mathbf{f}(\mathbf{x}[k+1], \mathbf{u}[k \quad 1])) \\ \vdots \\ \mathbf{g}(\mathbf{f}(\mathbf{x}[k+N_y \quad 1], \mathbf{u}[k \quad 1])) \end{pmatrix} + \mathbf{1}\boldsymbol{\eta}[k] \tag{15}$$

$$\Delta\boldsymbol{\eta}[k] = (1 \quad \omega)(\widehat{\mathbf{y}}[kjk \quad 1] \quad \mathbf{y_m}[k]) + \omega\Delta\boldsymbol{\eta}[k \quad 1] \tag{16}$$

$$\boldsymbol{\eta}[k] = \boldsymbol{\eta}[k \quad 1] + K\Delta\boldsymbol{\eta}[k] \tag{17}$$

$$\widehat{\mathbf{y}}[kjk \quad 1] = \mathbf{g}(\mathbf{f}(\mathbf{x}[k \quad 1], \mathbf{u}[k \quad 1])) + \boldsymbol{\eta}[k \quad 1], \tag{18}$$

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 [17] proposes a method on how to tune the filter parameters. To simplify this formulation, we assume that the system being controlled has only one output, however we can either use the same filter for different outputs, or apply different filters with the same formulation. We first define the a priori error $\epsilon[k]$ as:

$$\epsilon[k] = y_m[k] - y_{esn}[k|k-1], \tag{19}$$

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:

$$e[k] = y_m[k] - \widehat{y}[k|k-1]. \tag{20}$$

We desire to find a transfer function between the a priori error and the a posteriori error, which dictates the dynamics for the correction. To this end, we first expand Eq. (20), and substitute Eq. (19) inside, obtaining:

$$e[k] = \epsilon[k] - \eta[k-1]. \tag{21}$$

In terms of the a posteriori error, by putting Eq. (20) in (16) the equations that define the correction factor become:

$$\Delta\eta[k] = \omega\Delta\eta[k-1] + (1-\omega)e[k] \tag{22}$$

$$\eta[k] = \eta[k-1] + K\Delta\eta[k]. \tag{23}$$

Applying the z transform into Eqs. (21)-(23), we obtain:

$$e(z) = \epsilon(z) - z^{\ 1}\eta(z) \tag{24}$$

$$\eta(z) = \frac{K}{1 - z^{\ 1}}\Delta\eta(z) \tag{25}$$

$$\Delta\eta(z) = \frac{1-\omega}{1 - \omega z^{\ 1}}e(z); \tag{26}$$

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

$$\frac{e(z)}{\epsilon(z)} = \frac{z^2 - (\omega+1)z + \omega}{z^2 - (\omega + 1 - K(1-\omega))z + \omega}. \tag{27}$$

It is noticed that, since we have 1 as a zero in this transfer function, the steady state error is guaranteed to be 0 for a constant a prior error [17]. Our next step is to tune K and ω. We set the denominator of the transfer function to be equal to a given characteristic polynomial with two equal valued roots:

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

where $a = [0, 1)$ is the root of the polynomial, and the desired location of the poles for the transfer function. Defining the charateristic polynomial and forcing the error dynamics to have two real, stable poles, we now calculate the gain K and the filter frequency ω as follows, deriving from equation (27):

$$\omega = a^2 \tag{28}$$

$$K = \frac{a^2 - 2a + 1}{1 - a^2}. \tag{29}$$

These equations define K and ω based only on the pair of desired poles for the error correction dynamics. To turn off the filter dynamics, we may set $\omega = a = 0$, and $K = 1$.

By tuning K and ω so that the pole pair $\{a, a\}$ is stable, we can reject any constant-valued disturbance, as the dynamics have a zero $z_0 = 1$. This, of course, includes the Taylor series approximation error $\mathcal{O}(\|\Delta\mathbf{U}\|^2)$, which is seen by the filter as another source of disturbance.

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

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

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

where $\mathbf{Y}_{ref}$ 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 [3], the cost function is formulated as follows:

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

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:

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

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:

$$\mathbf{\Delta U}_{\min} \quad \mathbf{I\Delta U} \quad \mathbf{\Delta U}_{\max},$$

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

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

$$\begin{aligned} \min_{\mathbf{\Delta U}} \; & J(\mathbf{\Delta U}) = \mathbf{\Delta U}^T \mathbf{H} \mathbf{\Delta U} + \mathbf{c}^T \mathbf{\Delta U} \\ & \text{s.t. } \mathbf{I\Delta U} \leq \mathbf{\Delta U}_{\max} \\ & -\mathbf{I\Delta U} \leq -\mathbf{\Delta U}_{\min} \\ & \mathbf{T\Delta U} \leq \mathbf{1u}_{\max} - \mathbf{1u}[k-1] \\ & -\mathbf{T\Delta U} \leq -\mathbf{1u}_{\min} + \mathbf{1u}[k-1], \end{aligned} \tag{30}$$

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.

Finding a stability attraction region for such controller is not a trivial task, and constitutes another work on its own. For instance, [35] formulates a stability analysis method for MPC with a quasi-LPV model for a dynamical system, which might be applied for PNMPC. However, for the unconstrained and local case (when the operation is close to an equilibrium), closed-loop stability can be analyzed using stability analysis methods for linear MPC [3], e.g., reducing the linear MPC into a PID controller.

IV. Applications

A. Introduction

The ESN for the PNMPC was implemented from scratch using only Python and NumPy. Also, the Oger [36] Python toolbox was used for grid search, regularization, and parameter analysis (Section IV-C2). The plant models[3] were implemented in the JModelica framework, which compiles the Modelica language using Python. In order to wash out the initial transient of the ESN, it is advisable to first run the ESN-PNMPC alongside the plant for a few time steps in open loop (200 time steps in the case of the applications below).

B. Error Metrics

1) Model Error: The model error (ESN prediction) can be measured on the test data (with respect to the desired output $\mathbf{d}[k]$), or during control (w.r.t. the measured plant response $\mathbf{y_m}[k]$ for an input $\mathbf{u}[k]$), with or without the correction filter:

Relative Prior Error is the relative error of the ESN without the correction filter (given by $\mathbf{y}[k] = \mathbf{y_{esn}}[k]$):

$$e_{pre,\%}[k] = 100 \times \left| \frac{\mathbf{y_m}[k] - \mathbf{y_{esn}}[k]}{\mathbf{y_m}[k]} \right|. \tag{31}$$

Relative Posterior Error is the relative error of the ESN with the correction filter (given by $\hat{\mathbf{y}}[k|k-1]$):

$$e_{post,\%}[k] = 100 \times \left| \frac{\mathbf{y_m}[k] - \hat{\mathbf{y}}[k|k-1]}{\mathbf{y_m}[k]} \right|. \tag{32}$$

Integral of Absolute Error (IAE) for the ESN prediction on test data:

$$IAE = \sum_{k=1}^{N_{test}} |\mathbf{d}[k] - \mathbf{y_{esn}}[k]|. \tag{33}$$

Root Mean Squared Error (RMSE) for the ESN prediction on test data:

$$RMSE = \sqrt{\frac{1}{N_{test}} \sum_{k=1}^{N_{test}} (\mathbf{d}[k] - \mathbf{y_{esn}}[k])^2}. \tag{34}$$

While the first two metrics above are computed per timestep k during ESN-PNMPC control execution, the latter two summarize the error on a given test set of size N_{test}.

2) Tracking Error: The tracking error is measured with respect to the reference $\mathbf{y_{ref}}$ and computed by using the **IAE** during control execution (where N is the simulation time):

$$IAE = \sum_{k=1}^{N} |\mathbf{y_m}[k] - \mathbf{y_{ref}}[k]|. \tag{35}$$

[3]Plant models are described with more details in the supplementary material (attached to the current paper submission).

C. Four-tank System

The four-tank system [37] 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. A detailed description is given in Appendix A of the supplementary material (attached to the current paper submission), showing the coupling between pumps and the tanks. In this work, the objective of the ESN-PNMPC is to control the level of tanks 1 and 2 (h_1 and h_2), while rejecting a disturbance flow $\omega_{dist,3}$ that is input into tank 3, by manipulating both pump voltages u_1 and u_2.

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

$$J = \sum_{j=1}^{N_y}(q_1 e_1^2[k+j|k] + q_2 e_2^2[k+j|k]) + \sum_{j=1}^{N_u}{}^{1}(r_1 \Delta u_1^2[k+i|k] + r_2 \Delta u_2^2[k+i|k]), \quad (36)$$

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:

$$e_1[k+j|k] = h_{1,ref}[k+j|k] - h_1[k+j|k] \quad (37)$$

$$e_2[k+j|k] = h_{2,ref}[k+j|k] - h_2[k+j|k], \quad (38)$$

with $h_{i,ref}$ being the reference for h_i, $i \in \{1,2\}$. The cost function in (36) is the widely used quadratic penalization of the set point error and the control increment [3], and, if the system is linear in the input, constitutes a QP.

We also impose the following constraints in the system:

$$0V \leq u_1, u_2 \leq 5V \quad (39)$$

$$|\Delta u_1|, |\Delta u_2| \leq \Delta u_{\max} \quad (40)$$

$$h_{i,\min} \leq h_i \leq h_{i,\max},\ i \in \{3,4\}, \quad (41)$$

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, we consider two experimental cases:

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 (Section IV-C2). The second case is used in a disturbance rejection and tracking test (Section IV-C4), using the ESN with the configuration that yields the lowest control error obtained in the first case.

2) Identification and Hyperparameter Analysis: The identification of the model is based on the training of the readout output layer of the ESN according to (3), which finds the weights $\mathbf{W_r^o}$ connecting the reservoir layer to the output layer by solving the linear system in (3). $\mathbf{A}$ 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, (1) 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? We will 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) [1] 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 (using the non-minimum phase model in [37], which is also described in the supplementary material). 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) [34], while the latter $10,000$ timesteps were used to measure the ESN's test prediction performance using the IAE metric defined in (33). Control performance is computed in terms of IAE defined in (35) 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. 3.

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. 4a, while the mean IAE for the control task, while applying the ESN-PNMPC on the plant, is shown in Fig. 4b. For this control simulation, the PNMPC part was setup according to Section IV-C3. Here, the correction filter is not used, and thus, the control is in open-loop. This is because we aim to evaluate the capacity of the ESN in helping the control performance, without the *help* of the error

correction filter. We can note that the configuration of the spectral radius and leak rate that yields the minimum IAE for the ESN's prediction (Fig. 4a) 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. 4b, 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.

We have also performed the same type of grid search analysis for reservoirs containing 400 units. The IAE surface of the ESN's prediction is somewhat different from Fig. 4, being more flat, that is, the black area where the error is lowest is wider. In the following results section, we use 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.

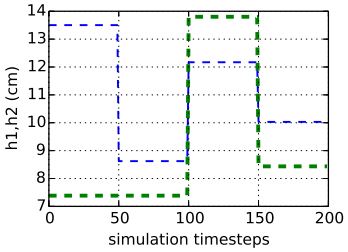
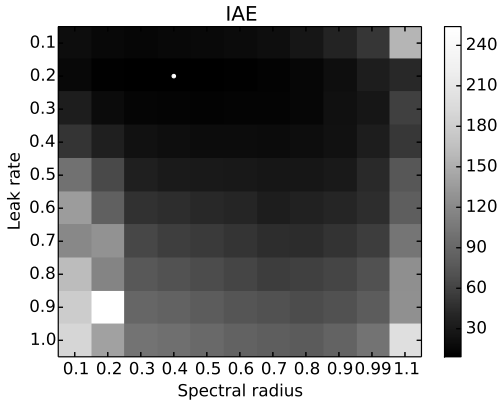


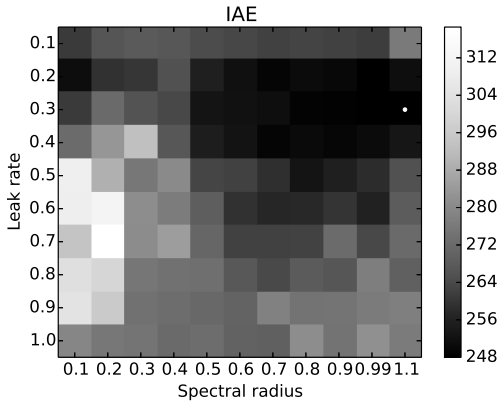
Fig. 3: Randomly generated reference signal $\mathbf{h_{ref}}$ for the level of tanks h_1 and h_2 to be used for control performance evaluation (IAE in (35)) with a duration of 200 timesteps.

Another experiment evaluates the test error (RMSE) as a function of the reservoir size and the size of the training set (Fig. 5). 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 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.

3) PNMPC Setup: After an open-loop test, we observed that the slowest step response of the system was of 35 time steps. We then decided to use a prediction horizon of 50 time steps (500 seconds), so that the steady state is captured in one



(a) IAE of ESN's prediction on test data



(b) IAE of MPC during control

Fig. 4: 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. (a) Test IAE between ESN's prediction and reference signal for 10,000 timesteps. (b) IAE between the plant's measured response during a MPC task and a randomly generated test reference signal for 200 timesteps from Fig 3.

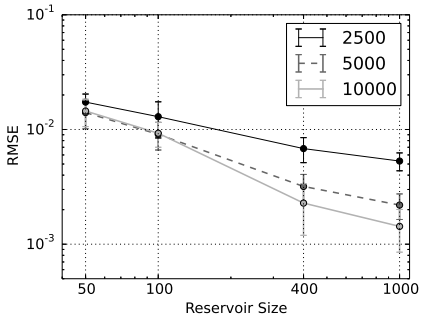


Fig. 5: Test RMSE as a function of reservoir size (50, 100, 400, 1000) and training set (2500, 5000 and 10,000 instances).

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

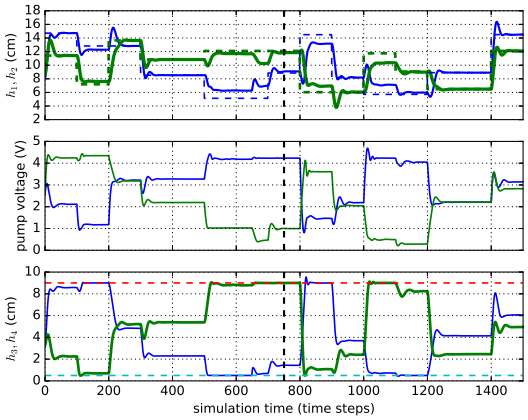
K and ω were tuned so that the pole of the a priori-posteriori error dynamics is $a = 0.5$ for each variable, which means that the a posteriori error shall converge to 0 at 70 seconds, once a bias appears in the a priori error. The identity matrix was used for both error weights $\mathbf{Q}$ and control variation weights $\mathbf{R}$, so the errors and the variations are equally penalized (the system is normalized in the PNMPC point of view).

4) Results for Tracking and Disturbance Rejection: 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 IV-C2. Note that the results on this section are made with completely new unseen data, as the reference signal for control is randomly generated once again.

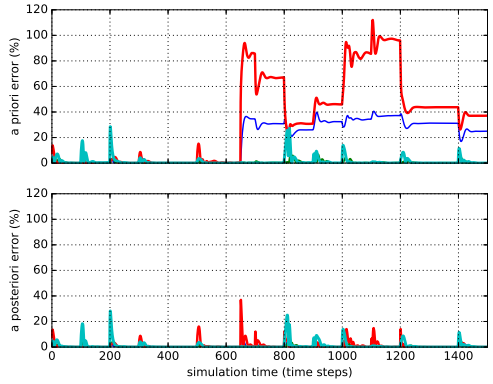
We used an APRBS signal with range $h_{i,ref} \in [5, 15]$, $i \in \{1, 2\}$ over $3,000$ time steps as a reference setpoint signal, and the step-type flow disturbance in Tank 3 ($\omega_{dist,3}$) was injected in the system halfway the simulation to showcase disturbance rejection and robustness to an 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. 6a 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, 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. 6a, we observe that 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. 6b, 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 is sacrificed, which is the reason why at some points the reference is not tracked, as shown in Fig. 6a.

5) Comparison with LSTM-PNMPC: We also compared ESN-PNMPC to a LSTM-PNMPC controller, where an LSTM [38] (Long Short-Term Memory) is used instead of an ESN. The LSTM network was trained for 40 epochs with the ADAM [39] optimization algorithm using exactly the same dataset as the ESN ($40,000$ samples, see Section IV-C2), 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* [40] is used to calculate 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. 7 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



(a) 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,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.



(b) 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 filter correction is applied.

Fig. 6: 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 $k = 750$. Subfigure 6a presents the full tracking, constrained variables and disturbance rejection results, and 6b presents the a priori and a posteriori error for all tank levels.

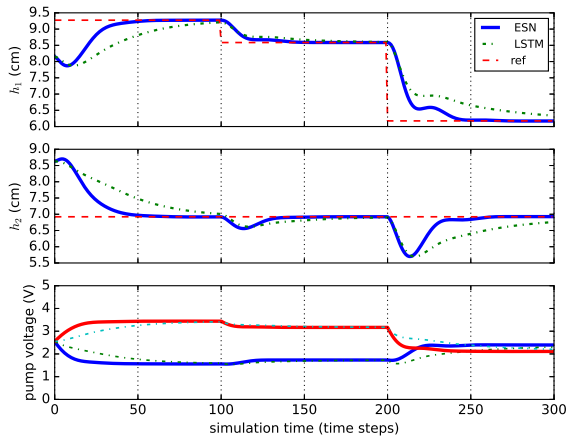


Fig. 7: Comparison between the ESN-PNMPC and LSTM-PNMPC for the control of the four-tank system, for 300 simulation time steps. The first and second plots correspond to the levels of tank 1 and 2, respectively, while the third plot shows the voltage for each pump.

TABLE I: 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 G calc. time (s)	**0.34**	174.8
best G calc. time (s)	**0.247**	170.9
worst 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

error for the controller run.

Table I summarizes the results, showing the IAE between 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. We can observe that 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 [40] 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. We note that 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 currently implemented LSTM. Using our available hardware (Intel Core i5, 3.10 GHz, 4 cores), 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 worst-case computation time of the controller is 0.671 s.

6) Comparison to Other Controllers: Here, we evaluate ESN-PNMPC in relation to a PI controller, a Linear MPC, specifically the Dynamic Matrix Control (DMC) strategy [3], and a linearizing MPC strategy called Approximate Predictive control (APC) [27] 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 [37]. Each discrete PI has the form:

$$\frac{U(z)}{E(z)} = K\frac{z - z_0}{z - 1}, \tag{42}$$

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 [37], which is a property of the system as a whole, and not of each individual transfer function.

Dynamic Matrix Control (DMC) [3]. 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 [3]. The optimization problem solved in the DMC iteration is exactly the same as in the PNMPC, with the same normalized variables.

APC consists of a Generalized Predictive Controller [3] where the discrete transfer function is obtained online by linearizing a neural network with external dynamics [1]. The coefficients of the transfer function are obtained directly through the output gradients with respect to the inputs. Details for the architecture and training of the feedforward neural network utilized for this comparison are available in Section VI-D of the supplementary material.

Fig. 8 shows that the DMC was slightly slower than the ESN-PNMPC, which was more agressive for larger changes in reference, such as at $k = 400$. Table II shows the IAE for the control task from Fig. 8, 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 operating point (thus having the matrix $\mathbf{G}$ fixed [3]), whereas ESN-PNMPC updates these matrices at each time step. An unconstrained DMC is equivalent to a PID controller [3], just as an unconstrained PNMPC can be

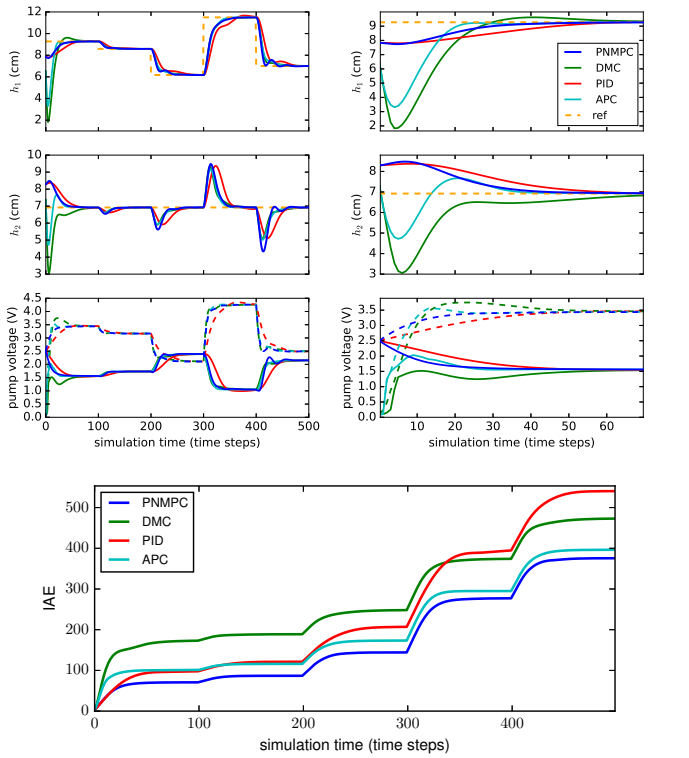


Fig. 8: Comparison between the ESN-PNMPC, a DMC controller, a PI controller and the APC for the four-tanks problem. The reference signal is given by a dashed orange line. Left: the control results for 500 time steps are shown, where the first two upper plots correspond to the levels of tank 1 and 2, respectively, while the third plot shows the corresponding pair of control actions for each controller (with u_2 in dashed line). Right: the initial 70 time steps from the left plot. Lower plot shows the IAE over simulation time for each method.

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) [41]), but this result from the four-tanks 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.

The ESN-PNMPC achieved better performance in terms of IAE than the APC, with the exact same MPC parameters. The APC controller reduces the nonlinear control problem into a pure GPC problem using a transfer function, while the PNMPC works with trajectory linearization instead [26]. The APC showed a worst case execution time of 0.41 seconds, and a mean execution time of 0.2 seconds, which is in the same order of magnitude as the ESN. This is explainable by the fact that the ESN model has more states, and therefore a more complex output-to-input gradient to calculate than the

TABLE II: Tracking error in relation to PI controller, Linear MPC (DMC) and APC (four-tanks problem).

	ESN-PNMPC	PI	DMC	APC
IAE	**375.56**	540.45	473	396.34
IAE (h_1)	**203.50**	321.04	290.76	251.91
IAE (h_2)	172.06	219.41	182.24	**144.43**

feedforward NN with external dynamics of APC. Also, in APC, the gradient is only calculated once per time step, to obtain the linearized transfer function, while the PNMPC must calculate the gradient over the whole prediction horizon, as it is based on trajectory linearization. Despite this difference in both approaches, the total control time for both controllers was in the same order of magnitude. Notice that the APC strategy itself is suitable to input-output dynamic models, but not to models such as the ESN. For APC, the transfer function coefficients are directly obtained by the neural network gradients. This is not the case of RNNs, which would require the chain rule for recursively computing the gradient. Finally, we expect that for more nonlinear, complex plants and cases of a restricted amount of training data, the ESN-PNMPC approach will increase its advantage in relation to APC, since the former can model a wide range of dynamic systems from data very well and with relative ease [9], [10], [11].

D. Oil Production Platform

The oil production platform application considered here consists of a composition model of two gas-lifted oil wells and one riser, connected by a manifold [12]. Previous applications of ESNs in oil and gas include [42], [12], [4]. Overall, the whole system has 120 algebraic variables, 10 state variables, 5 input variables, and exactly 5 boundary conditions[4].

1) Problem Formulation: The problem for this case study is akin to a control problem addressed in [12]. 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). To formulate this tracking problem in the context of predictive control, we use the following quadratic cost function:

$$J = \sum_{j=1}^{N_y} (q_1 e_1^2[k+j|k] + q_2 e_2^2[k+j|k]) + \sum_{j=1}^{N_u} {}^{1} (r_1 \Delta u_{ch1,}^2 + r_2 \Delta u_{ch,2}^2), \quad (43)$$

where:

$$e_1[k+j|k] = \widetilde{P}_{bh,1}[k+j|k] - P_{bh,1}[k+j|k], \quad (44)$$

$$e_2[k+j|k] = \widetilde{P}_{bh,2}[k+j|k] - P_{bh,2}[k+j|k], \quad (45)$$

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

[4]The system model is described in more details in Appendix B of the supplementary material (attached to the current paper submission).

In this specific case, we do not deal with output constraints. However, constraints related to valve limitation and rate limiting are added to the formulation:

$$0.01 \leq u_{ch,1}, u_{ch,2} \leq 1 \tag{46}$$

$$-\Delta u_{\max} \leq \Delta u_{ch,1}, \Delta u_{ch,2} \leq \Delta u_{\max}. \tag{47}$$

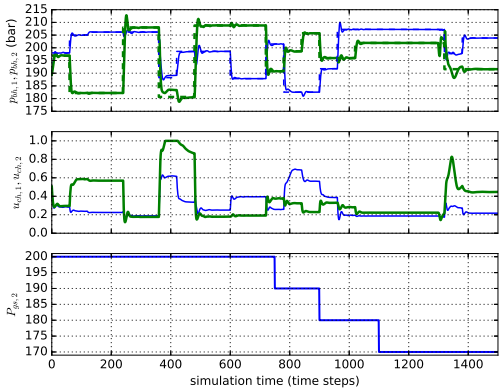
By default, just as in the four-tank experiment, $\Delta u_{\max} = 0.2$. The ESN and the PNMPC consider a sampling of 1 min for the platform, which is suitable for such applications.

2) Identification of the Platform: The datasets are generated by exciting the system with the APRBS [1] for both inputs $u_{ch,1}, u_{ch,2}$. An example is shown in section C of the Appendix. 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, presented in Fig. 9.

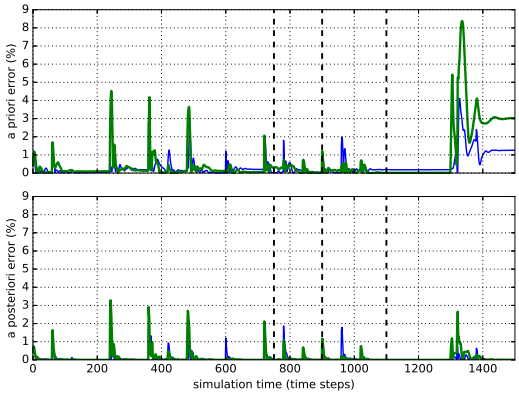
3) PNMPC Setup: After training the ESN model as described in the previous section, we tuned the PNMPC for this application. To decide the prediction horizon, we executed an open-loop test which showed that the system reaches steady-state after 30 time steps (minutes). For each prediction window to include a portion of steady-state time, we set the prediction horizon to be 40 time steps. The control horizon has length 5, chosen as a sixth of the prediction horizon [3]. We have also tuned the filter parameters K and ω so that the pole $a = 0.3$ is induced 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.

4) Tracking and Disturbance Rejection: 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 [43]) 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. 9, 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. We can infer that the model changes more drastically as the pressure keeps dropping, and that this parametric change affects our a priori prediction less than in the four-tank case. Even then,



(a) 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.



(b) 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 filter correction is applied. The vertical dashed lines mark each moment when $P_{gs,2}$ changes value.

Fig. 9: 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. Subfigure 9a shows the full tracking and disturbance rejection results, while subfigure 9b presents the a priori and a posteriori error for all well pressures.

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. 9, 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.

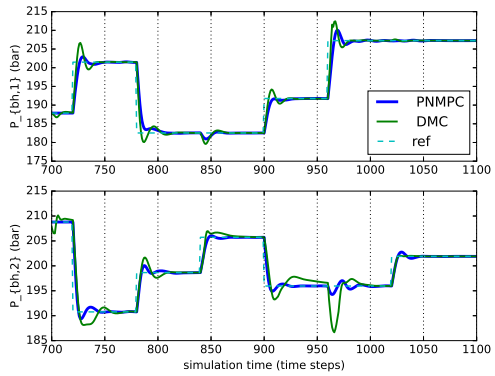


Fig. 10: Comparison between the ESN-PNMPC, and a DMC-type MPC for the oil production platform during pressure disturbances applied to the gas-lift supply, between time steps 700 and 1100 of Fig. 9a. The reference signal is given by the dashed cyan line for the bottom-hole pressures $P_{bh,1}$ and $P_{bh,2}$ 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.

We have also designed a DMC for the control of the two wells in the platform, using 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. 10 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 agressive, therefore more prone 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 clearly seen after $k = 950$ in Fig. 10. 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 [3] uses only the current modeling error to correct the free response.

V. Conclusion

This work has proposed an extremely efficient Reservoir Computing-based control scheme for control of unknown nonlinear dynamic systems, namely ESN-PNMPC, which consists of: an ESN which provides a fast, efficient system identification of dynamic systems; and the Practical NMPC which advocates the full nonlinear precision for the free response and linear approximation of the forced response. Our proposal relies on a new recursive formulation for the computation of derivatives of the output signals with respect to the inputs, granting an efficient calculation of the forced response and ultimately a computationally cheap control action when compared to the original PNMPC [15].

We have considerably extended our previous work [4], for instance: deriving a methodology to tune the filter parameters; analysing the ESN hyperparameters effect on control performance; showing the application of the proposed ESN-PNMPC in two different scenarios, namely the four-tank system and a two-well one-riser system; and comparing it to other control approaches, such as: LSTM with PNMPC, PID, DMC (Linear MPC) and Approximate Predictive Control (APC).

Given the good results achieved so far, as future work, we hope to employ ESN-PNMPC for controlling real world plants, dealing with real-time computation issues and produce an implementation into a micro-controller. A formal stability analysis for ESN-PNMPC is also currently being developed by the authors.

References

[1] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*, 1st ed. Berlin: Springer, 2001.
[2] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.
[3] E. Camacho and C. Bordons, *Model Predictive Control*. Springer, 1999.
[4] J. P. Jordanou, E. Camponogara, E. A. Antonelo, and M. A. S. Aguiar, "Nonlinear model predictive control of an oil well with echo state networks," *IFAC-PapersOnLine*, vol. 51, no. 8, pp. 13–18, 2018.
[5] U. Eren, A. Prach, B. B. Koçer, S. V. Raković, E. Kayacan, and B. Açıkmeşe, "Model predictive control in aerospace systems: Current state and opportunities," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 7, pp. 1541–1566, 2017.
[6] T. P. Nascimento, C. E. T. Dórea, and L. M. G. Gonçalves, "Nonlinear model predictive control for trajectory tracking of nonholonomic mobile robots: A modified approach," *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, 2018.
[7] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," 2017.
[8] K. Doya, "Bifurcations in the learning of recurrent neural networks," in *Proc. of IEEE Int. Symp. on Circ. and Syst.*, vol. 6, 1992, pp. 2777–2780.
[9] B. Schrauwen, D. Verstraeten, and J. Van Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the 15th European Symposium on Artificial Neural Networks*, 2007, pp. 471–482.
[10] H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication," *Science*, vol. 304, no. 5667, pp. 78–80, Apr 2004.
[11] H. Jaeger, "The ''echo state'' approach to analysing and training recurrent neural networks – with an erratum note," Fraunhofer Institute for Autonomous Intelligent Systems, Tech. Rep. GMD 148, 2001.
[12] J. P. Jordanou, E. A. Antonelo, and E. Camponogara, "Online learning control with echo state networks of an oil production platform," *Eng. Appl. Artif. Intell.*, vol. 85, pp. 214–228, 2019.
[13] H. V. H. Ayala, D. Habineza, M. Rakotondrabe, and L. dos Santos Coelho, "Nonlinear black-box system identification through coevolutionary algorithms and radial basis function artificial neural networks," *Applied Soft Computing*, vol. 87, p. 105990, 2020.
[14] M. A. Ouali and M. Ladjal, "Nonlinear dynamical systems modelling and identification using type-2 fuzzy logic. metaheuristic algorithms based approach." in *2020 International Conference on Electrical Engineering (ICEE)*. IEEE, 2020, pp. 1–6.

[15] A. Plucênio, D. Pagano, A. Bruciapaglia, and J. Normey-Rico, "A practical approach to predictive control for nonlinear processes," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 210–215, 2007.
[16] Y. Pan and J. Wang, "Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3089–3101, 2012.
[17] A. Plucênio, "Development of non linear control techniques for the lifting of multiphase fluids," Ph.D. dissertation, Federal University of Santa Catarina, Brazil, 2013.
[18] K. Xiang, B. N. Li, L. Zhang, M. Pang, M. Wang, and X. Li, "Regularized Taylor echo state networks for predictive control of partially observed systems," *IEEE Access*, vol. 4, pp. 3300–3309, 2016.
[19] C.-T. Chen, *Linear System Theory and Design*, 3rd ed. New York, NY, USA: Oxford University Press, Inc., 1998.
[20] L. B. Armenio, E. Terzi, M. Farina, and R. Scattolini, "Model predictive control design for dynamical systems learned by echo state networks," *IEEE Control Systems Letters*, vol. 3, no. 4, pp. 1044–1049, Oct 2019.
[21] L. B. Armenio, E. Terzi, M. Farina, and R. Scattolini, "Echo state networks: analysis, training and predictive control," *CoRR*, vol. abs/1902.01618, 2019.
[22] Q. Chen, H. Shi, and M. Sun, "Echo state network-based backstepping adaptive iterative learning control for strict-feedback systems: An error-tracking approach," *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3009–3022, 2020.
[23] S. I. Han and J. M. Lee, "Precise positioning of nonsmooth dynamic systems using fuzzy wavelet echo state networks and dynamic surface sliding mode control," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 11, pp. 5124–5136, 2013.
[24] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, pp. 1–19, 09 2016.
[25] E. Terzi, T. Bonetti, D. Saccani, M. Farina, L. Fagiano, and R. Scattolini, "Learning-based predictive control of the cooling system of a large business centre," *Control Engineering Practice*, vol. 97, p. 104348, 2020.
[26] M. Ławryńczuk, *Computationally Efficient Model Predictive Control Algorithms*. Springer International Publishing, 2014.
[27] J. Witt and H. Werner, *Approximate Model Predictive Control for Nonlinear Multivariable Systems*, 08 2010.
[28] H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," *Neural Networks*, vol. 20, no. 3, pp. 335–352, 2007.
[29] H. Jaeger, "Short term memory in echo state networks," German National Research Center for Information Technology, Tech. Rep. GMD Report 152, March 2002.
[30] M. C. Ozturk, D. Xu, and J. C. Príncipe, "Analysis and design of echo state networks," *Neural Computation*, vol. 19, no. 1, pp. 111–138, 2007.
[31] D. Verstraeten and B. Schrauwen, "On the quantification of dynamics in reservoir computing," in *Artificial Neural Networks*, C. Alippi, M. Polycarpou, C. Panayiotou, and G. Ellinas, Eds., 2009, pp. 985–994.
[32] D. Verstraeten, J. Dambre, X. Dutoit, and B. Schrauwen, "Memory versus non-linearity in reservoirs," in *Int. Joint Conference on Neural Networks*. Barcelona, Spain: IEEE, 2010, pp. 18–23.
[33] M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," *Complex Systems*, vol. 3, no. 4, 1989.
[34] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. New York: Springer-Verlag Inc., 2006.
[35] P. S. G. Cisneros and H. Werner, "A dissipativity formulation for stability analysis of nonlinear and parameter dependent MPC," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 3894–3899.
[36] D. Verstraeten, B. Schrauwen, S. Dieleman, P. Brakel, P. Buteneers, and D. Pecevski, "Oger: modular learning architectures for large-scale sequential processing," *Journal of Machine Learning Research*, vol. 13, pp. 2995–2998, 2012.
[37] K. Johansson, "The quadruple-tank process: A multivariable laboratory process with an adjustable zero," *IEEE Transactions on Control Systems Technology*, vol. 8, pp. 456–465, 2000.
[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–80, 12 1997.
[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.
[40] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS 2017 Workshop on Autodiff*, 2017.
[41] H. Chen, H. Kremling, and F. Allgöwer, "Nonlinear predictive control of a benchmark cstr," *Proceedings of the 3rd European Control Conference, Rome-Italy.*, pp. 3247–3252, 01 1995.
[42] E. A. Antonelo, E. Camponogara, and B. Foss, "Echo state networks for data-driven downhole pressure estimation in gas-lift oil wells," *Neural Networks*, vol. 85, pp. 106–117, 2017.
[43] E. Jahanshahi, S. Skogestad, and H. Hansen, "Control structure design for stabilizing unstable gas-lift oil wells," *IFAC Proceedings Volumes*, vol. 45, no. 15, pp. 93–100, 2012.
[44] A. S. M. Brandão, D. M. Lima, M. V. A. da Costa Filho, and J. E. Normey-Rico, "A comparative study on embedded MPC for industrial processes," in *Anais do XXII Congresso Brasileiro de Automática*, 2018.
[45] E. Jahanshahi and S. Skogestad, "Simplified dynamical models for control of severe slugging in multiphase risers," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1634–1639, 2011.

Jean P. Jordanou received the M.Sc. degree in automation and systems engineering from the Federal University of Santa Catarina, Brazil, in 2019. He joined as a Ph.D. student at the same institution rightly afterwards. His research interests include data-driven algorithms for optimization and control, reservoir computing, model order reduction and model predictive control.

Eric Aislan Antonelo received the Ph.D. and M.Sc. degrees in Computer Engineering respectively from Ghent University, Belgium, in 2011 and Halmstad University, Sweden, in 2006. He is currently a faculty member of the Department of Automation and Systems Engineering at the Federal University of Santa Catarina, Brazil. His research is mainly focused on reservoir computing and machine learning for industrial applications (modeling, detection, and control tasks), as well as imitation learning approaches for autonomous vehicles.

Eduardo Camponogara received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, USA, in 2000. After serving as a postdoctoral fellow at the Institute for Complex Engineered Systems, USA, he joined the faculty of the Department of Automation and Systems Engineering at the Federal University of Santa Catarina, Brazil, in 2002. His research interests include systems optimization, distributed optimization algorithms, and data-driven algorithms for optimization and control

VI. Supplementary Material for the Paper "Echo State Networks for Practical Nonlinear Model Predictive Control of Unknown Dynamic Systems"

A. Four-tank System

The four-tank system [37], widely used as a benchmark of multivariate and nonlinear control systems with coupled variables, is depicted in Figure 11. The system consists of two pumps $j \in \{1, 2\}$, two directional valves, where γ_j is how much of the pump flow enters tank j and not the other tank ($j + 1$ or $j + 2$) and four tanks $i \in \{1, 2, 3, 4\}$. 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.

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

$$\dot{h}_1 = \frac{\gamma_1 k_1 u_1 + \omega_3 - \omega_1}{A_1} \tag{48}$$

$$\dot{h}_2 = \frac{\gamma_2 k_2 u_2 + \omega_4 - \omega_2}{A_2} \tag{49}$$

$$\dot{h}_3 = \frac{(1 - \gamma_2) k_2 u_2 - \omega_3 + \omega_{dist,3}}{A_3} \tag{50}$$

$$\dot{h}_4 = \frac{(1 - \gamma_1) k_1 u_1 - \omega_4}{A_4}, \tag{51}$$

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:

$$\omega_i = a_i \sqrt{2 g h_i}, \tag{52}$$

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 [37], 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.

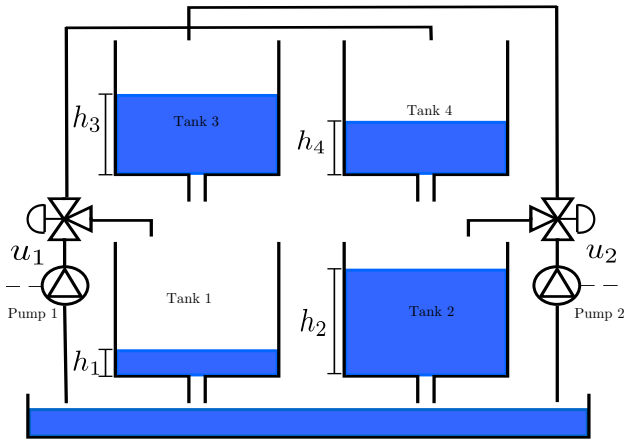


Fig. 11: Representation of a four-tank system. Adapted from [44]. Description is given in the text.

B. Oil Production Platform

The considered platform application consists of two wells and one riser, being the same one that was used in our previous work [12]. Another work that uses ESNs in oil and gas applications is [42], which presents a soft sensor for remote estimation in offshore production platforms.

We employ a compositional model of two gas-lifted oil wells and one riser, connected by a manifold. Figure 12 depicts the platform, whose components have the following properties:

Wells: Each well is modeled as the reduced order model in [43]. 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 $\dot{m}_{L,t}$; each state is calculated by the following mass balance equations:

$$\dot{m}_{G,a} = \omega_{G,in} - \omega_{G,inj} \tag{53}$$

$$\dot{m}_{G,t} = \omega_{G,inj} + \omega_{G,res} - \omega_{G,out} \tag{54}$$

$$\dot{m}_{L,t} = \omega_{L,res} - \omega_{L,out}. \tag{55}$$

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

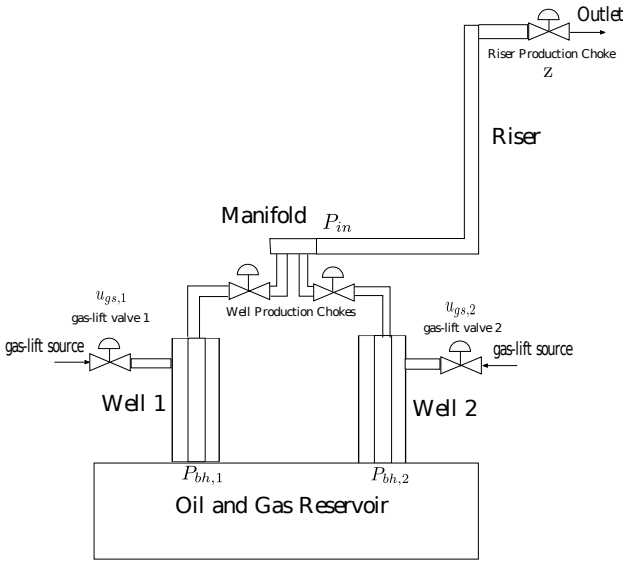


Fig. 12: Representation of an oil platform containing two wells and one riser. From [12].

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 [43] for more details on the well model.

Riser: The riser is modeled as a reduced order model developed in [45]. 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 12. 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 [12]. 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 12 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 [12] are used, where the boundary conditions are $P_{gs,1} = P_{gs,2} = 200$ bar and the output pressure is $P_o = 50$ bar.

C. Datasets for ESN Training

In this section, we present the plot of the datasets employed in training of both the four-tank and the oil platform system.

Figure 13 shows the data utilized to train the ESN for the four-tank control problem. The first plot shows the levels h_1 and h_2, which were directly employed as control variables, and the second plot shows the levels h_3 and h_4, which were raised because of the MPC problem constraints. The third plot shows the APRBS excitation signal employed in the system for the manipulated variables.

Figure 14 depicts the data utilized for the platform problem, where the first plot shows the controlled variables: each well bottom-hole pressure. The bottom-most plot shows the excitation signal utilized for each well choke valve.

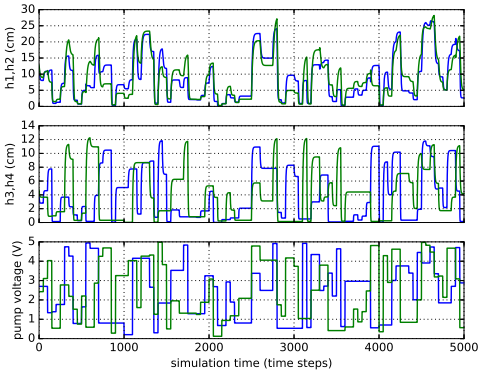


Fig. 13: Plot of the first $5,000$ training points for the four-tank control problem.

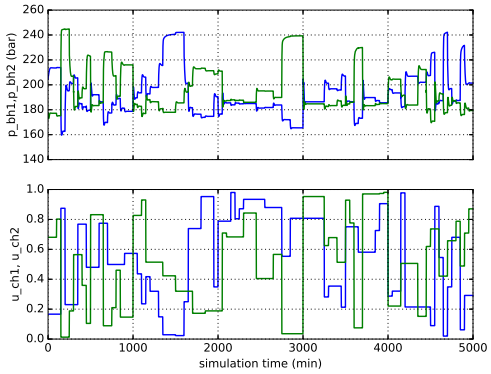


Fig. 14: Plot of the first $5,000$ training points for the platform control problem.

D. Feedforward NN for Approximate Predictive Control

This section describes the NN model embedded in the Approximate Predictive Control (APC). For the four-tanks system, a feedforward NN with 2nd order external dynamics is employed, as described below:

$$\mathbf{y}[k+1] = \mathbf{f}(\mathbf{u}[k], \mathbf{y}[k], \mathbf{y}[k-1]). \tag{56}$$

The network has a total of 10 inputs and 4 outputs, forming a dynamic system with 8 states (i.e., the four tank levels displaced in time twice).

The network has one hidden layer with 40 ReLu neurons, trained with the exact same dataset that was employed for the ESN ($40,000$ time steps for training and validation, $10,000$ time steps for testing). The network was implemented in Tensorflow/Keras, and trained with ADAM with a minibatch size of 32, with a regularization parameter of $\lambda = 10^{-8}$ for weight decay. The stop criterion was 100 epochs after the validation loss reaches a minimum, where the model with lowest validation error is saved. We experimented with different number of layers and neurons per layer, but a hidden layer of 40 neurons had the best control performance, while being faster in terms of computation time for being the smallest NN among the tested ones. The final trained model had a test error magnitude of around 10^{-3} for each output, which is similar

to the error by the ESN. Notice that the network for APC is harder to train than an ESN.