Reinforcement Learning based MPC with Neural Dynamical Models

Saket Adhau, Sébastien Gros, and Sigurd Skogestad

Abstract— **This paper presents an end-to-end learning approach to developing a Nonlinear Model Predictive Control (NMPC) policy, which does not require an explicit first-principles model and assumes that the system dynamics are either unknown or partially known. The paper proposes the use of available measurements to identify a nominal Recurrent Neural Network (RNN) model to capture the nonlinear dynamics, which includes constraints on the state variables and input. To address the issue of suboptimal control policies resulting from simply fitting the model to the data, the paper uses Reinforcement learning (RL) to tune the NMPC scheme and generate an optimal policy for the real system. The approach's novelty lies in the use of RL to overcome the limitations of the nominal RNN model and generate a more accurate control policy. The paper discusses the implementation aspects of initial state estimation for RNN models and integration of neural models in MPC. The presented method is demonstrated on a classic benchmark control problem: cascaded two tank system (CTS).**

Index Terms— **Reinforcement Learning, Nonlinear Model Predictive Control, Recurrent Neural Networks**

I. Introduction

Model Predictive Control (MPC) is a widely used control strategy in various fields such as process control, robotics, and autonomous systems [1]. The success of MPC depends on the availability of a mathematical model that predicts the future behavior of the system and optimizes control actions accordingly [2]. However, uncertainties in model parameters, measurement errors, stochiasticity, unmodeled dynamics, and incomplete knowledge of the system can introduce errors, leading to suboptimal control performance. To address these challenges, the authors of [3] proposed combining data-driven and model-based control to enhance MPC performance. This approach enables the control of complex systems with nonlinear dynamics and uncertainties, without relying on an accurate mathematical model of the system.

Although the approach presented in [3] can address model uncertainties, it still assumes the availability of some form of mathematical model to define the state and action spaces. In real-world scenarios, a detailed mathematical model or complete knowledge of the system dynamics may not be available, and only input-output data may be accessible. In such cases, techniques such as system identification [4] or data-driven approaches [5] can be used to estimate the system's dynamics and construct a model that can be utilized for control purposes.

In this paper, we propose to construct a "domain-aware" neural network model for modeling the unknown system dynamics. The purpose of this model is to replace the classic dynamic model used in [3]. The neural network can handle complex nonlinear systems where defining a state space may be difficult. During the training phase, we include constraints on both the state variables and input of the model to ensures that it remains physically meaningful and consistent with the real system. By constraining the state variables, we can ensure that the model remains within a physically feasible region of operation, and by constraining the input, we ensure that the control inputs generated by the model are within the range of physically feasible inputs. This dynamical model will then be integrated into an RL-based MPC framework proposed in [3] to build an optimal MPC policy with focus on closed loop performance. Our aim is to improve the closed loop performance of the system by using this neural network model, which provides more flexibility and adaptability compared to a classic dynamic model.

The initial state estimation is addressed using a past window of both measurements and outputs. The measured outputs are sequentially fed to the model instead of the predicted ones. We present discussions on adaptation of the neural model in case of model mismatch, and also the integration of neural dynamical models in MPC pipeline. We analyze the performance of the proposed approach on a classical benchmark problem: Cascaded Two tank system (CTS). The proposed method is highly sample efficient with only one requirement: a sufficiently excited time-series data set of the system dynamics.

The paper is organized into several sections. Section I provides an introduction to the problem and motivation for the research. Section II reviews the relevant literature and presents the background information on the RL-based MPC approach. Section III describes the approach to system identification using neural networks. In Section IV, we present the implementation of our proposed approach and the results of our experiments. Finally, Section V concludes the paper.

II. Background

Obtaining an accurate mathematical model to represent the dynamics of a physical system can be a daunting task, particularly for complex systems with nonlinear dynamics

This work was supported by the Research Council of Norway, under the IKTPLUSS program (Project number 299585) and Safe Reinforcement Learning using MPC (SARLEM) (Project number 300172)

Saket Adhau is with the Department of Sustainable Energy Technology, SINTEF Industry, 7031 Trondheim, Norway (e-mail: saket.adhau@sintef.no)

Sigurd Skogestad is with the Department of Chemical Engineering, Norwegian University of Science and Technology, 7491, Trondheim, Norway. (e-mail: sigurd.skogestad@ntnu.no)

Sebastien Gros is with the Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway (e-mail: sebastien.gros@ntnu.no)

and uncertainties. A well-performing model should be able to capture the essential dynamics of the system and accurately predict its behavior in response to control inputs. Additionally, the model should be inexpensive to evaluate and differentiate, especially for real-time MPC applications. This necessitates that the model be of appropriate complexity, neither too simple to overlook important dynamics nor too complex to become computationally intractable.

While System identification is often considered an essential step in developing an MPC strategy, its necessity depends on the specific application and the characteristics of the system being controlled. Numerous techniques have been developed for system identification, ranging from classical methods to more advanced data-driven approaches [6].

Classical system identification methods include black-box modeling techniques such as ARX, ARMAX, and state-space modeling [7]. These methods do not require an explicit understanding of the underlying system dynamics to model the system, and can be effective for complex nonlinear systems. However, they still require some knowledge of the system and its behavior, specifically the inputs and outputs of the system and their relationship to each other. As a result, these methods may not be suitable for systems with limited or poor quality input-output data.

Neural dynamical models are a powerful tool for modeling nonlinear dynamics as they can capture complex and nonlinear relationships between input and output data without relying on a mathematical model [8]. Unlike traditional modeling techniques, these models can handle nonlinearities and interactions between system variables. RNNs are a type of neural network that can maintain information from previous inputs, making them well-suited for time series data and increasingly popular for tackling nonlinear problems [9], [10]. With their ability to capture the dynamic behavior of a system, neural dynamical models can be useful for system identification and control in a wide range of applications.

Recently, researchers [11] proposed an approach for modeling the dynamics of complex systems using RNNs, demonstrating the potential of this method in accurately identifying and controlling nonlinear systems. Another study [12] utilized constrained block-nonlinear neural networks to identify and model nonlinear systems. The authors demonstrated that this method can effectively capture the dynamics of a wide range of systems, and can be used to design control strategies that achieve improved performance compared to traditional approaches.

A. *Reinforcement Learning*

Consider a problem described by Markov Decision Process (MDP) with potentially stochastic state transition dynamics denoted by,

$$\mathbb{P}[\boldsymbol{s}_+|\boldsymbol{s},\boldsymbol{a}]\,, \tag{1}$$

where $\boldsymbol{s},\boldsymbol{a}$ is the current state-action pair and $\boldsymbol{s}_+$ is the subsequent state and $\mathbb{P}$ is the conditional probability. The state-action space $\boldsymbol{s}\in\mathcal{S},\boldsymbol{a}\in\mathcal{A}$ is assumed to be continuous and $\mathbb{P}$ is a probability density, but the theory proposed here is valid in general. The notation used for state transitions in (1) is standard in MDP literature, whereas the control literature commonly employs $\boldsymbol{s}_+ = f(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\zeta})$ where $\boldsymbol{\zeta}$ is a stochastic variable and f is a possibly nonlinear function.

Let us consider a stage cost $L(\boldsymbol{s}_k,\boldsymbol{a}_k)$ associated to the MDP can take the form of:

$$L(\boldsymbol{s}_k,\boldsymbol{a}_k) = \ell(\boldsymbol{s}_k,\boldsymbol{a}_k) + \mathcal{I}_\infty(\boldsymbol{h}(\boldsymbol{s}_k,\boldsymbol{a}_k))\,, \tag{2}$$

where $\ell(\boldsymbol{s}_k,\boldsymbol{a}_k)$ captures the cost given to different input-output pairs, and the constraints

$$\boldsymbol{h}(\boldsymbol{s}_k,\boldsymbol{a}_k)\leq 0\,, \tag{3}$$

capture undesirable states and inputs, and infinite values are given to L when (3) is violated. Additionally, we use the indicator function,

$$\mathcal{I}_\infty(\boldsymbol{x}) = \begin{cases}\infty & \text{if } \boldsymbol{x}_i>0 \text{ for some } i\\ 0 & \text{otherwise.}\end{cases} \tag{4}$$

With the addition of a discount factor $0<\gamma\leq 1$, given (1) and (2), any optimal discounted policy $\boldsymbol{\pi}_\star$ minimizes the expected total discounted cost,

$$J(\boldsymbol{\pi}) = \mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k L(\boldsymbol{s}_k,\boldsymbol{a}_k)\middle|\boldsymbol{a}_k=\boldsymbol{\pi}(\boldsymbol{s}_k)\right]\,, \tag{5}$$

where the expected value $\mathbb{E}[\cdot]$ is taken over the (possibly) stochastic state transition dynamics (1) in closed loop with policy $\boldsymbol{\pi}$.

The optimal action-value function $Q_\star$, value function $V_\star$, and optimal policy $\boldsymbol{\pi}_\star(\boldsymbol{s})$ associated to the MDP, are defined by the Bellman equations [13]:

$$Q_\star(\boldsymbol{s},\boldsymbol{a}) = L(\boldsymbol{s},\boldsymbol{a}) + \gamma\mathbb{E}\left[V_\star(\boldsymbol{s}_+)\,|\,\boldsymbol{s},\boldsymbol{a}\right]\,, \tag{6a}$$
$$V_\star(\boldsymbol{s}) = Q_\star(\boldsymbol{s},\pi_\star(\boldsymbol{s})) = \min_{a} Q_\star(\boldsymbol{s},\boldsymbol{a})\,. \tag{6b}$$

Throughout the paper we will assume that the associated stage cost L, and the discount factor γ yield a well posed problem, i.e. the value function defined by (6) are well posed, and finite over some non-empty sets.

We then consider a model of the real system having state transition dynamics

$$\mathbb{P}[\hat{\boldsymbol{s}}_+|\boldsymbol{s},\boldsymbol{a}] \tag{7}$$

which typically do not match (1) perfectly. We now consider a modified stage cost defined as:

$$\hat{L}(\boldsymbol{s},\boldsymbol{a}) = \begin{cases}Q_\star(\boldsymbol{s},\boldsymbol{a})-\gamma\mathcal{V}^+(\boldsymbol{s},\boldsymbol{a}) & \text{if } |\mathcal{V}^+(\boldsymbol{s},\boldsymbol{a})|<\infty\\ \infty & \text{otherwise}\end{cases} \tag{8}$$

where $\mathcal{V}^+(\boldsymbol{s},\boldsymbol{a}) = \mathbb{E}[V_\star(\hat{\boldsymbol{s}}_+)|\boldsymbol{s},\boldsymbol{a}]$ where the expectation is taken over the distribution (7). We now briefly reiterate the central theorem in [3], stating that under some condition, the policy $\boldsymbol{\pi}_\star$ that minimizes the stage cost L for the true dynamics (1) can also be generated using the model (7) combined with the stage cost $\hat{L}$. The approach here used is to bypass the difficult evaluation of (8), and replacing it by learning $\hat{L}$ directly from the data.

Let us consider the parametrization of value function $V_\star$ using the following ENMPC scheme parameterized using θ:

$$V_\theta(\boldsymbol{s}) = \min_{\boldsymbol{x},\boldsymbol{u}} \; \lambda_\theta(\boldsymbol{x}) + \gamma^N V_\theta^{\mathrm{f}}(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} \gamma^k \ell_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{9a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{s}, \tag{9b}$$

$$\boldsymbol{x}_{k+1} = \mathbf{f}_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{9c}$$

$$\boldsymbol{g}(\boldsymbol{u}_k) \leq 0, \tag{9d}$$

$$\boldsymbol{h}_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k) \leq 0, \quad \boldsymbol{h}_\theta^{\mathrm{f}}(\boldsymbol{x}_N) \leq 0, \tag{9e}$$

where the ENMPC scheme (9) holds a model parametrization f_θ, a constraint parametrization $\boldsymbol{h}_\theta$, a parametrization of the stage cost ℓ_θ and terminal cost V_θ^{f} with the storage function λ_θ. While the pure input constraints are fixed $\boldsymbol{g}(a) \leq 0$ are arguably fixed, the mixed constraints in the ENMPC scheme ought to be modified, in order to capture the domain where $\hat{L}(\boldsymbol{s}, \boldsymbol{a})$ is finite. Additionally, a relaxed version of L and of mixed constraints is considered to avoid infinite penalties in case of constraint violations.

Further, we define the action-value function $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ as

$$Q_\theta(\boldsymbol{s}, \boldsymbol{a}) = \min_{\boldsymbol{u},\boldsymbol{x}} \quad (9a)\,, \tag{10a}$$

$$\text{s.t.} \quad (9b) - (9e)\,, \tag{10b}$$

$$\boldsymbol{u}_0 = \boldsymbol{a}\,. \tag{10c}$$

The proposed parametrization trivially satisfies the fundamental equalities underlying the Bellman equations, i.e.,

$$\pi_\theta(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}} Q_\theta(\boldsymbol{s}, \boldsymbol{a}), \quad V_\theta(\boldsymbol{s}) = \min_{\boldsymbol{a}} Q_\theta(\boldsymbol{s}, \boldsymbol{a}). \tag{11}$$

B. *Q−Learning for ENMPC*

In the $Q-$learning algorithm, the action-value function is represented by $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$, where $\boldsymbol{\theta}$ is a vector of parameters. The classical approach to $Q-$learning involves updating these parameters based on temporal differences δ, which are computed from the difference between the predicted and actual rewards received from taking a particular action in a given state. This approach is coupled with instantaneous policy updates, where:

$$\delta_k = \ell(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma \min_{\boldsymbol{a}_{k+1}} Q_\theta\left(\boldsymbol{s}_{k+1}, \boldsymbol{a}_{k+1}\right) - Q_\theta\left(\boldsymbol{s}_k, \boldsymbol{a}_k\right), \tag{12a}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \delta_k \nabla_\theta Q_\theta\left(\boldsymbol{s}_k, \boldsymbol{a}_k\right), \tag{12b}$$

where the scalar $\alpha > 0$ is the learning rate. For further clarification such as the gradient calculation in the RL-based MPC scheme, readers are directed to the paper by [3].

C. *Need for a Model*

In the approach presented above, a model is needed to formulate the state and action spaces, which are essential for training the RL agent. While the agent can learn the optimal policy from data, it still requires a model to operate within a well-defined state and action space. Therefore, even though the RL-based MPC scheme can handle model mismatch or uncertainties, a model is still required to formulate the state and action spaces, as well as to evaluate the performance of the learned policy.

III. Recurrent Neural Networks

The use of RNNs for modeling dynamical systems in MPC settings has several advantages over classical system identification methods. RNNs excel at capturing complex and nonlinear relationships between input and output data, making them well-suited for time series data and nonlinear problems.

A. *Constrained system identification using RNNs*

Consider a discrete-time dynamical system, denoted by $\mathcal{S}$ which takes input values $u \in \mathbb{R}^{n_u}$ and produces output values $y \in \mathbb{R}^{n_y}$. At any given time step k, the system's output may depend on all previous input samples, i.e., it exhibits memory,

$$y_k = \mathcal{S}(u_k, u_{k-1}, \ldots, u_0). \tag{13}$$

Assuming that the system $\mathcal{S}$ can be adequately represented by a nonlinear dynamical model, denoted by $\mathcal{M}$:

$$\hat{y}_k = \mathcal{M}(u_k, u_{k-1}, \ldots, u_0; \theta) \tag{14}$$

where $\theta \in \mathbb{R}^{n_\theta}$ is a parameter vector to be determined.

We use bold symbols $\mathbf{u}$ and $\mathbf{y}$ to denote the sequence of N input and output samples in a dataset $\mathcal{D}$.

In this paper, we assume that the model has the following state-space representation:

$$\boldsymbol{x}_{k+1} = \mathcal{F}(\boldsymbol{x}_k, \boldsymbol{u}_k; \theta_{\mathcal{F}}) \tag{15a}$$

$$\hat{\boldsymbol{y}}_k = \mathcal{G}(\boldsymbol{x}_k, \boldsymbol{u}_k; \theta_{\mathcal{G}}), \tag{15b}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ is the unknown system state, $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is the control input at time k. In this paper, $\mathcal{F}$ and $\mathcal{G}$ are the state-update and output mappings respectively, both parameterized by $\theta = \theta_{\mathcal{F}}, \theta_{\mathcal{G}}$. In particular, $\mathcal{F}$ and $\mathcal{G}$ are neural networks, and the overall model (15) is a RNN.

The data is assumed to be available in the form of input-output tuples, denoted by

$$\mathcal{D} = \{(\mathbf{y}_k^i, \mathbf{u}_k^i), (\mathbf{y}_{k+1}^i, \mathbf{u}_{k+1}^i), \ldots, (\mathbf{y}_{k+N}^i, \mathbf{u}_{k+N}^i)\}, i \in \mathbb{N}_1^n \tag{16}$$

where n is the number of sample trajectories with $\mathbb{N}$ time steps. The primary aim is to learn a constrained neural equivalent of the unknown system dynamics, given input-output time-series dataset (16) obtained by observing the system.

B. *System identification loss*

The neural state space dynamics (15) are trained on the sampled input-output trajectories (16) using the loss function

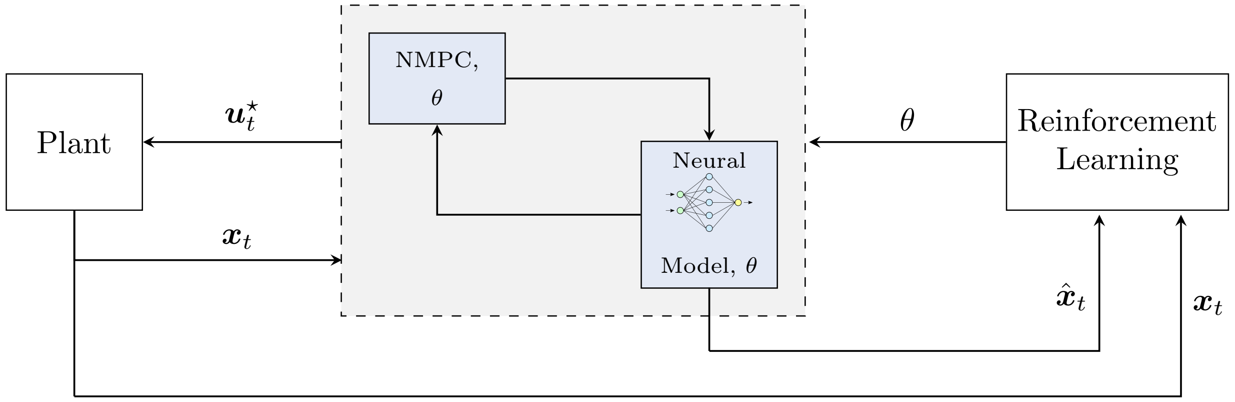


Fig. 1: Illustration depicting the proposed methodology that employs nonlinear neural dynamical models, constructed using measurement data, within a RL based MPC framework, to attain an optimal control policy.

described below:

$$\begin{aligned}\mathcal{L}_\phi(\mathbf{Y}^{\text{true}}, \mathbf{Y}, \bar{\mathbf{Y}}, \underline{\mathbf{Y}}|\theta) = \\ \frac{1}{nN}\sum_{i=1}^{n}\sum_{k=1}^{N}\Bigg(\Big|\Big|\mathbf{y}_k^{\text{true},i} - \mathbf{y}_k^i\Big|\Big|_2^2 + Q_y\Big|\Big|p(\mathbf{y}_k^i, \underline{\mathbf{y}}_{k+1}^i)\Big|\Big|_2^2 + \\ +Q_y\Big|\Big|p(\mathbf{y}_k^i, \bar{\mathbf{y}}_{k+1}^i)\Big|\Big|_2^2 + Q_u\Big|\Big|p(f_u(\mathbf{u}_k^i), \underline{f}_u)\Big|\Big|_2^2 + \\ Q_u\Big|\Big|p(f_u(\mathbf{u}_k^i), \bar{f}_u)\Big|\Big|_2^2\Bigg),\end{aligned} \tag{17}$$

where k represents the time step of the prediction horizon N, whereas i is the batch index of n sampled trajectories. The tracking loss is given by the first term, taken as two norm over the residual vector between true $\mathbf{Y}^{\text{true}} = \{\mathbf{Y}_1^{\text{true},i}, \dots, \mathbf{Y}_N^{\text{true},i}\}$ and predicted $\mathbf{Y} = \{\mathbf{Y}_1^i, \dots, \mathbf{Y}_N^i\}$ output trajectories over N step. The second term is optional and is used to promote smoothening of the trajectories of dynamic models. Furthermore, the third and fourth terms impose constraints on the output trajectories using penalty functions. We apply inequality constraints on predictions during the training phase itself, making the unconstrained optimization problem amenable to the gradient-based optimization methods used in deep learning. We use the following penalty functions for time-varying lower and upper bounds $\bar{\mathbf{y}}_k, \underline{\mathbf{y}}_k$:

$$\underline{p}(\mathbf{y}_k, \underline{\mathbf{y}}_k) = \max(0, -\mathbf{y}_k + \underline{\mathbf{y}}_k)\,, \tag{18a}$$
$$\bar{p}(\mathbf{y}_k, \bar{\mathbf{y}}_k) = \max(0, \mathbf{y}_k - \bar{\mathbf{y}}_k)\,, \tag{18b}$$

The penalty functions shown in (18) are easy to implement and can be modified according the specific requirements as shown in [14]. The iterative gradient-based optimization algorithms, like Adam [15], can be used to minimize the loss. The required derivatives computation is accomplished using standard reverse-mode Automatic Differentiation (AD) algorithms and software [16]. As standard practice is followed for nominal model training, we will not discuss it in more detail.

C. RNN initial state estimation

The estimation of initial state in an RNN model is very important, especially for our application of MPC [17]. The initial state can affect the state transition predictions and thus needs to be estimated accurately. Extending the ideas used in [11] we denote the predicted output $\hat{\boldsymbol{y}}_k$ which is a part of the state:

$$\boldsymbol{x}_{k+1} = \mathcal{F}(\boldsymbol{x}_k, \hat{\boldsymbol{y}}_k, \boldsymbol{u}_k; \theta_{\mathcal{F}}) \tag{19a}$$
$$\hat{\boldsymbol{y}}_{k+1} = \mathcal{G}(\boldsymbol{x}_{k+1}, \hat{\boldsymbol{y}}_k, \boldsymbol{u}_k; \theta_{\mathcal{G}}), \tag{19b}$$
$$\text{for } k = 0, 1, \dots, N_c - 1. \tag{19c}$$

While the predictions are performed using (19), the initial state is estimated using a window of the past data $\{\boldsymbol{y}_{k-1}, \dots, \boldsymbol{y}_{k-N_c}, \boldsymbol{u}_{k-1}, \dots, \boldsymbol{u}_{k-N_c}\}$, where the measured outputs $\boldsymbol{y}_{k-i}$ are sequentially fed to the model instead of the predicted ones for N_c steps, and $\boldsymbol{x}_{k-N_c}$ is set to zero. In particular, the state estimation is performed by opening the output prediction loop for the first N_c steps:

$$\boldsymbol{x}_{k+1} = \mathcal{F}(\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{u}_k; \theta_{\mathcal{F}}) \tag{20a}$$
$$\hat{\boldsymbol{y}}_{k+1} = \mathcal{G}(\boldsymbol{x}_{k+1}, \boldsymbol{y}_k, \boldsymbol{u}_k; \theta_{\mathcal{G}}), \tag{20b}$$

D. Using RNN models in MPC scheme

In MPC, the control law is typically obtained by solving an optimization problem that involves the system model. If a neural network is chosen as the system model, it is crucial for it to be differentiable to calculate gradients for the optimization problem. This is because most optimization algorithms typically used in MPC rely on the gradient information to find the optimal control inputs.

Fortunately, most neural network architectures used in practice, such as feedforward networks and recurrent networks, are differentiable and can be used in MPC. However, there are additional challenges when using neural network-based models in MPC. The choice of network architecture and the selection of appropriate regularization methods are some of the important considerations. Despite these challenges, neural network-based models can still be a powerful tool for improving the performance of MPC systems when carefully designed and implemented.

The detailed procedure for the proposed method is illustrated in Fig. 1. The method begins by constructing a neural model of the system using available measurement data. This model is then utilized in RL based MPC scheme to optimize the closed loop performance. By employing the RL-MPC scheme, we are able to modify the cost function as well as model (if required) to adapt and learn from the system's behavior in real-time, leading to improved control performance.

IV. Case Study

To demonstrate the effectiveness of neural network-based models in an RL-based MPC approach, we utilize the Cascaded Tanks System (CTS)described in [18].

The CTS is a control system for fluid level that involves two tanks with free outlets, supplied by a pump. The controlled pump transfers water from a bottom reservoir to the upper tank. The water from the upper tank flows into the lower tank via a small opening and subsequently into the reservoir via another small opening. The system input, denoted as u, is the water flow from the bottom reservoir to the upper tank. Meanwhile, the state variables h_1 and h_2 represent the water levels in the upper and lower tanks, respectively.

The identification of the CTS system poses a significant challenge due to several factors. The system's hard saturation nonlinearity, combined with its weakly nonlinear behavior during regular operation, makes the identification process complex. Moreover, the overflow from the upper to the lower tank introduces input-dependent process noise, which further complicates the problem.

DataSet: The experimental dataset was obtained from the collection [1] of public benchmarks widely used in system identification. These datasets are commonly used as a benchmark for testing the accuracy and robustness of system identification methods. The training and test datasets contain 1024 points each, collected at a constant sampling time $\Delta t = 5\,\text{s}$.

Metrics: We use Root Mean Square Error (RMSE) to evaluate the performance index as it is suggested in the description of the benchmark problem [18]:

Neural Network: The proposed method utilizes the PyTorch Deep Learning (DL) framework [16] for training neural network models with the adaptation of RNN based system identification module from [11]. The network comprises of one hidden layer with 128 neurons, utilizing Leaky ReLU as the activation function. Gradient-based optimization is carried out using the Adam optimizer [15], with the learning rate parameter set to 10^{-3}. The batch size is set to 64, and $n = 10000$ iterations are carried out to ensure convergence to a cost function plateau. The training process takes approximately 400 seconds.

The neural networks' weight parameters are initialized with random Gaussian variables with zero mean and a standard deviation of 10^{-4}, while the bias terms are initialized to zero, These values are observed to be effective in aiding the convergence of the model during the training process.

RL and MPC: In this work, for formulating the MPC problem, we use CasADi [19] with IPOPT as the Nonlinear Programming (NLP) solver [20]. For learning the optimal policy, we use $Q-$learning with a learning rate of $\alpha = 10^{-3}$. The discount factor, was set to $\gamma = 0.99$.

To enable the use of arbitrary neural network models trained in PyTorch with CasADi, we use the implementation [2] from [21]. This implementation allows seamless integration of neural network models trained in PyTorch into the CasADi framework, enabling the use of deep learning techniques in control problems. All the simulations have been performed on a Macbook Pro with Intel Core i7 running at $2.6\,\text{GHz}$ and $16\,\text{GB}$ of memory.

A. Result Analysis:

Fig. 2a displays the time trajectories of both the true and model output. Due to visualization constraints, only a subset of the test dataset is presented. Notably, the fitted model accurately characterizes the system dynamics with a high degree of precision. The neural loop training achieved an RMSE of 0.2912 indicating a satisfactory level of accuracy in the model's predictions. Although our proposed approach has resulted in better RMSE results compared to the state-of-the-art black-box nonlinear identification methods used for this benchmark [22]–[24], it is important to note that the main goal of this approach is not to obtain a highly accurate model of the system. Rather, the primary objective is to develop a model that can effectively function within the RL-MPC framework. The RL-MPC approach can then fine-tune the control policy as required.

Fig. 2b depicts the simulated closed loop trajectories of RL based MPC for the CTS benchmark problem. Our intention is to showcase that the proposed approach is capable of achieving the desired results. It is important to note that these results are presented solely for the purpose of demonstrating the capabilities of the approach and are not compared to any other technique.

V. Conclusion

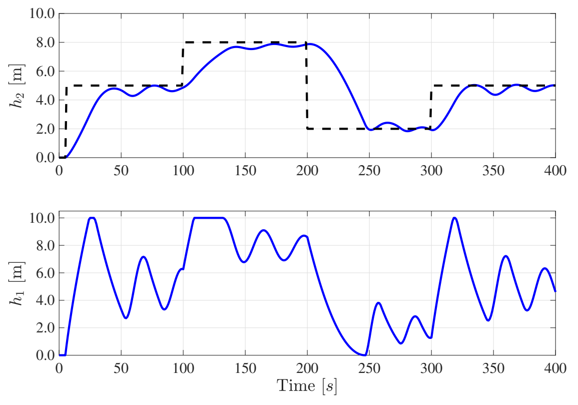
In many real-world scenarios, obtaining a completely accurate model of a system can be challenging, and often measurement data is only available. In this paper, we propose a method that utilizes this measurement data to build a nonlinear neural dynamical model of the system.

The developed neural model may not be completely accurate, but we show that it is still possible to achieve optimal control policies using an RL-based MPC framework. RL-MPC uses a combination of reinforcement learning and model predictive control to learn and improve the overall closed loop performance even when the model is incomplete or inaccurate.

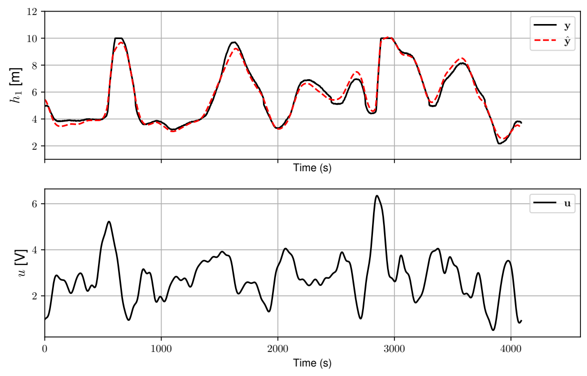
This approach has significant potential for practical applications in various fields where accurate models are difficult to

[1]http://www.nonlinearbenchmark.org

[2]https://github.com/TUM-AAS/ml-casadi



(a)



(b)

Fig. 2: (a) CTS Benchmark: Open-loop trajectories of the trained neural model (red) and real system (black). (b) CTS Benchmark: Simulated closed-loop control trajectories demonstrating RL based MPC with neural dynamical model. The upper pane depicts the reference (dashed black) alongside the controlled state h_2, while the lower panel shows the measured state h_1.

obtain or not available. The use of neural networks and RL-based MPC can offer a reliable and adaptive control strategy that can learn and improve based on the system's feedback.

REFERENCES

[1] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, pp. 733–764, 2003.

[2] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[3] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, pp. 636–648, 2019.

[4] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society A*, vol. 474, p. 20180335, 2018.

[5] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Transactions on Automatic Control*, vol. 66, pp. 1702–1717, 2020.

[6] J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented road map," *IEEE Control Systems Magazine*, vol. 39, pp. 28–99, 2019.

[7] E.-W. Bai and F. Giri, "Introduction to block-oriented nonlinear systems," *Block-oriented Nonlinear System Identification*, pp. 3–11, 2010.

[8] S. Chen, S. A. Billings, and P. Grant, "Non-linear system identification using neural networks," *International journal of control*, vol. 51, pp. 1191–1214, 1990.

[9] T. W. Chow and Y. Fang, "A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics," *IEEE transactions on industrial electronics*, vol. 45, pp. 151–161, 1998.

[10] J. de Jesús Rubio and W. Yu, "Nonlinear system identification with recurrent neural networks and dead-zone kalman filter algorithm," *Neurocomputing*, vol. 70, pp. 2460–2466, 2007.

[11] M. Forgione, A. Muni, D. Piga, and M. Gallieri, "On the adaptation of recurrent neural networks for system identification," *arXiv preprint arXiv:2201.08660*, 2022.

[12] E. Skomski, S. Vasisht, C. Wight, A. Tuor, J. Drgoňa, and D. Vrabie, "Constrained block nonlinear neural dynamical models," *2021 American Control Conference (ACC)*, pp. 3993–4000, 2021.

[13] D. P. Bertsekas, "Dynamic programming and optimal control, volume 1 of optimization and computation series," *Athena Scientific, Belmont, MA, USA, 3rd edition*, vol. 2, 2005.

[14] S. Adhau, V. V. Naik, and S. Skogestad, "Constrained neural networks for approximate nonlinear model predictive control," *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 295–300, 2021.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] 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," 2017.

[17] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.

[18] M. Schoukens and J. P. Noël, "Three benchmarks addressing open challenges in nonlinear system identification," *IFAC-PapersOnLine*, vol. 50, pp. 446–451, 2017.

[19] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[20] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[21] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, pp. 2397–2404, 2023.

[22] R. Relan, K. Tiels, A. Marconato, and J. Schoukens, "An unstructured flexible nonlinear model for the cascaded water-tanks benchmark," *IFAC-PapersOnLine*, vol. 50, pp. 452–457, 2017.

[23] G. Birpoutsoukis, P. Z. Csurcsia, and J. Schoukens, "Efficient multidimensional regularization for volterra series estimation," *Mechanical Systems and Signal Processing*, vol. 104, pp. 896–914, 2018.

[24] A. Svensson and T. B. Schön, "A flexible state–space model for learning nonlinear dynamical systems," *Automatica*, vol. 80, pp. 189–199, 2017.