2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Computationally efficient predictive control based on ANN state-space models

Jan H. Hoekstra, Bence Cseppentő, Gerben I. Beintema, Maarten Schoukens, Zsolt Kollár, and Roland Tóth

***Abstract*— Artificial neural networks (ANN) have been shown to be flexible and effective function estimators for the identification of nonlinear state-space models. However, if the resulting models are used directly for nonlinear model predictive control (NMPC), the resulting nonlinear optimization problem is often overly complex due to the size of the network, requires the use of high-order observers to track the states of the ANN model, and the overall control scheme does not exploit the available autograd tools for these models. In this paper, we propose an efficient approach to auto-convert ANN state-space models to linear parameter-varying (LPV) form and solve predictive control problems by successive solutions of linear model predictive problems, corresponding to quadratic programs (QPs). Furthermore, we show how existing deep-learning methods, such as SUBNET that uses a state encoder, enable efficient implementation of MPCs on identified ANN models. Performance of the proposed approach is demonstrated by a simulation study on an unbalanced disc system.**

I. Introduction

Over the past decades, intensive research has been conducted on *model predictive control* (MPC) due to advantages of predictive methods over reactive feedback solutions and reliable constraint handling, making MPC a core technology in many industrial sectors [1]. However, current technological developments and increasing performance expectations are pushing system designs towards exhibiting more dominant nonlinear behavior, giving rise to an increasing need for reliable MPC solutions for *nonlinear* (NL) systems.

Despite an increasing range of highly competitive NMPC solutions, e.g, [2], [3], a serious obstacle that one encounters in their application in practice is the required existence of an accurate model of the underlying system. As obtaining models of increasingly complex system designs is becoming infeasible via previously-used first principles-based methods, data-driven techniques started to gain serious importance in practice. To accomplish accurate data-driven modelling of complex NL systems, *machine learning* methods have recently appeared to provide a reliable approach. Especially *artificial neural networks* (ANN) have been shown to be flexible in capturing complicated nonlinear dynamic relationships and effective in dealing with the high-complexity of the involved estimation problem. For control applications, ANN *state-space* (SS) models [4] are particularly attractive, and with the recent introduction of encoder-based methods, such as [5], [6], estimation of them can be reliably and computationally efficiently accomplished in practice.

While NMPC methods have been applied to various ANN models, e.g., in [7], [8], these approaches are mainly based on the application of existing general NMPC techniques and linearization-based methods. For example, it is not complicated to replace the nonlinear system model in any NMPC scheme with an ANN-SS model and solve the resulting problem with general solvers, e.g., [9]. However, ANN-based models have many nonlinear activation functions which all need to be evaluated and linearized by the solver without being aware of the underlying structure, often resulting in excessive computational complexity. Furthermore, the use of high-order observers is also required to track the states of the ANN model, increasing further the computational load.

In this paper, we propose a computationally efficient iterative MPC scheme for ANN-SS models through the use of automated *linear parameter varying* (LPV) conversion based on the *Fundamental Theorem of Calculus* (FTC) and iterative *quadratic programming* (QP). The computational complexity of this conversion is minimized through a computational structure that exploits the use of highly efficient autograd methods and multi-threading-based computing. Also, the encoder obtained with the identification of the ANN-SS model is used as a direct observer for state estimation. Our contributions can be summarized as follows:

- Approximation-free auto conversion of ANN-SS models to an LPV form through the FTC, efficiently computed via autograd methods and multi-threaded computing.
- Direct formulation of successive QP problems using the obtained optimal solution from the previous iteration step as the scheduling for the LPV prediction model.

The remainder of the paper first introduces the ANN-SS models in Section II, followed by the corresponding NMPC problem and the proposed LPV conversion scheme in Sections III and IV. The use of the LPV conversion to solve the NMPC efficiently via an iterative scheme is covered in Section V. An unbalanced disk simulation example is used to demonstrate the performance of the proposed MPC approach and compare it to a standard NMPC solution in Section VI.

II. Deep-learning based state-space models

Consider a *discrete-time* (DT) nonlinear system in the form:

$$x_{k+1} = f(x_k, u_k), \tag{1a}$$

$$y_k = h(x_k) + e_k, \tag{1b}$$

This research was supported by the Eötvös Loránd Research Network (grant. number: SA-77/2021) and the Gedeon Richter PhD scholarship of excellence by the Gedeon Richter Talentum Foundation.

B. Cseppentő and Zs. Kollár are with the Dept. of Measurement and Information Systems at the Budapest University of Technology and Economics, Hungary, while the rest of the authors are with the Control Systems group in the Dept. of Electrical Engineering at the Eindhoven University of Technology, The Netherlands. Roland Tóth is also affiliated with the Institute for Computer Science and Control, Budapest, Hungary.

Corresponding author: B. Cseppentő (cseppento@mit.bme.hu)

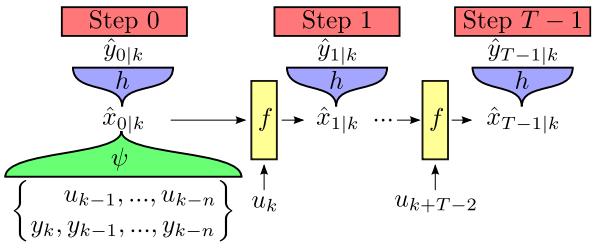

Fig. 1. SUBNET structure: the subspace encoder ψ_θ estimates the initial state at time index k based on past inputs and outputs, then the state is propagated through f_θ and h_θ multiple times until a simulation length T.

where $x_k \in \mathbb{R}^{n_\mathrm{x}}$ is the state, $u_k \in \mathbb{R}^{n_\mathrm{u}}$ is the input, $y_k \in \mathbb{R}^{n_\mathrm{y}}$ is the output signal of the system at time moment $k \in \mathbb{Z}$ with e_k an i.i.d. white noise process representing measurement noise, and the state-transition function $f : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}^{n_\mathrm{x}}$ and output function $h : \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}^{n_\mathrm{y}}$ are bounded functions.

Using a data sequence $\mathcal{D}_N = \{(y_k, u_k)\}_{k=1}^{N}$ generated by (1), we aim to identify a DT model of the form

$$\hat{x}_{k+1} = f_\theta\left(\hat{x}_k, u_k\right), \tag{2a}$$
$$\hat{y}_k = h_\theta\left(\hat{x}_k\right), \tag{2b}$$

where $\hat{x}_k \in \mathbb{R}^{n_{\hat{\mathrm{x}}}}$ is the model state, $\hat{y}_k \in \mathbb{R}^{n_\mathrm{y}}$ is the model output, and $f_\theta : \mathbb{R}^{n_{\hat{\mathrm{x}}}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}^{n_{\hat{\mathrm{x}}}}$ with $h_\theta : \mathbb{R}^{n_{\hat{\mathrm{x}}}} \to \mathbb{R}^{n_\mathrm{y}}$ are feedforward multi-layer artificial neural networks[1] with $\theta \in \mathbb{R}^{n_\theta}$ collecting the activation weights as model parameters. Under these considerations, (2) represents a recurrent neural network, also called an ANN-SS model [4].

In the literature, many approaches have been introduced to identify ANN-SS models, e.g., [6], [10]. One recent approach that provides computationally efficient estimation of (2) under statistical consistency guarantees is the SUBNET approach [5]. For computational efficiency, SUBNET uses the truncated prediction cost as the objective function to be minimized during identification which also enables the use of batch optimisation methods. This truncation corresponds to a forward simulation of (2) over multiple subsections of length T of the available data as depicted in Figure 1, starting from randomly selected time instances $k \in \{n+1, \ldots, N-T\}$. In order to compute such simulation batches, SUBNET co-estimates an "encoder" function ψ_θ together with f_θ and h_θ, which provides a direct estimate of the initial state $\hat{x}_{0|k}$ based on past input and output data, i.e., $\hat{x}_{0|k} = \psi_\theta(u_{k-n}^{k-1}, y_{k-n}^{k})$ where $u_k^{k+\tau} = [\ u_k^\top \ \cdots \ u_{k+\tau}^\top]^\top$ for $\tau \geq 0$ and $y_k^{k+\tau}$ is defined similarly. Next, we formulate the predictive control problem we intend to solve for identified ANN-SS models in the form of (2).

III. Nonlinear predictive control problem

Traditional NMPC corresponds to repeatedly solving an *optimal control problem* (OCP) on a finite prediction horizon using the current observation of the state variables x_k as an initial value, then applying the first element of the acquired input sequence to the system, see [1]. However, we consider the true NL system equations (1) to be unknown and to have only an approximate ANN-SS model (2) with estimated parameters $\hat{\theta}$. Considering a regulation objective w.r.t. a set point $x_\mathrm{ref} \in \mathbb{R}^{n_\mathrm{x}}$, NMPC problem based on (2) and under a quadratic cost is formulated as follows:

$$\min_{u_{0|k}^{N_\mathrm{p}-1}} \ \sum_{i=1}^{N_\mathrm{p}} \ell_i(\hat{x}_{i|k} - x_\mathrm{ref}, u_{i-1|k} - u_\mathrm{ref}), \tag{3a}$$
$$\text{s.t.} \ \ \hat{x}_{i+1|k} = f_{\hat{\theta}}(\hat{x}_{i|k}, u_{i|k}), \quad i \in \mathbb{I}_0^{N_\mathrm{p}-1}, \tag{3b}$$
$$\hat{y}_{i|k} = h_{\hat{\theta}}(\hat{x}_{i|k}), \quad i \in \mathbb{I}_1^{N_\mathrm{p}}, \tag{3c}$$
$$u_\mathrm{min} \leq u_{i|k} \leq u_\mathrm{max}, \quad i \in \mathbb{I}_0^{N_\mathrm{p}-1}, \tag{3d}$$
$$y_\mathrm{min} \leq \hat{y}_{i|k} \leq y_\mathrm{max}, \quad i \in \mathbb{I}_1^{N_\mathrm{p}}, \tag{3e}$$
$$\hat{x}_{0|k} = \hat{x}_k, \tag{3f}$$

where subscript $(i|k)$ means the predicted value of the variable at $i+k$ in the optimisation problem when the current time is $k \in \mathbb{Z}$, $\ell_i(x,u) = x^\top Q_i x + u^\top R_i u$ is the quadratic stage cost where Q_i are positive definite and R_i are positive semi-definite weighting matrices defining the user-chosen performance specification for the controller, $\mathbb{I}_{\tau_1}^{\tau_2} = \{k \in \mathbb{Z} \mid \tau_1 \leq k \leq \tau_2\}$ is an index set, while $y_\mathrm{min}, \ldots, u_\mathrm{max}$ correspond to box constraints[2] on the outputs and the inputs where $\leq$ and $\geq$ is considered element-wise, and N_p is the length of the prediction horizon (considered to be equivalent with the control horizon). As the model of the true system is unknown and the identified NL-SS model can be on arbitrary state basis, constraints imposed on the states would be meaningless. Therefore, only output constraints in (3e) are considered. On the other hand, to realize the set-point objective, it is assumed that a set point target $(x_\mathrm{ref}, u_\mathrm{ref})$ is synthesised for example based on the approach in [11]. Considering the tracking objective in terms of x_ref also makes possible to ensure stability of the predictive controller for example via terminal ingredients [11], however, due to the lack of space, this is not detailed in the paper.

Even under these considerations, the main challenge faced by existing NMPC methods applied on (3) is that fast online solutions of this nonlinear problem are required under potentially large state-dimensions and heavily complex functions $f_{\hat{\theta}}$ and $h_{\hat{\theta}}$, which are required to be passed analytically to the existing NMPC solvers. Hence, it becomes attractive to develop a dedicated predictive control solution that can exploit `autograd` tools for ANN-SS models to provide accelerated solutions for (3).

IV. Conversion to surrogate LPV form

To provide an efficient solution of (3), we aim to convert the ANN-SS model (2) with estimated parameters $\hat{\theta}$ to an LPV form. We will show in Section V that this form enables to solve (3) via a chain of computationally efficient QPs. To realize this objective, we need an efficient auto-conversion of an ANN-SS model (2) to an LPV-SS form

[1] Consider that each hidden layer is composed from m activation functions $\phi : \mathbb{R} \to \mathbb{R}$ in the form of $z_{i,j} = \phi(\sum_{l=1}^{m_{i-1}} \theta_{\mathrm{w},i,j,l} z_{i-1,l} + \theta_{\mathrm{b},i,j})$ where $z_i = \mathrm{col}(z_{i,1}, \ldots, z_{i,m_i})$ is the latent variable representing the output of layer $1 \leq i \leq q$. Here, $\mathrm{col}(\centerdot)$ denotes the composition of a column vector. For f_θ with q hidden-layers and linear input and output layers, this means $f_\theta(\hat{x}_k, u_k) = \theta_{\mathrm{w},q+1} z_q(k) + \theta_{\mathrm{b},q+1}$ and $z_0(k) = \mathrm{col}(\hat{x}_k, u_k)$.

[2] The use of box constraints is an arbitrary choice, in fact, any polyhedral set can be considered.

$$\hat{x}_{k+1} = A(p_k)\hat{x}_k + B(p_k)u_k, \tag{4a}$$
$$y_k = C(p_k)\hat{x}_k, \tag{4b}$$

where p_k is the so-called *scheduling variable*, dependent on $\hat{x}_k$ and u_k in terms of the *scheduling map* $p_k = \mu(\hat{x}_k, u_k)$. Note that (4) corresponds to $f_{\hat{\theta}}(\hat{x}, u) = A(\mu(\hat{x}, u))\hat{x} + B(\mu(\hat{x}, u))u$ and $h_{\hat{\theta}}(\hat{x}, u) = C(\mu(\hat{x}, u))\hat{x}$, which are non-unique factorizations of the nonlinearities. The map μ is often synthesized under the expectation that $A, \dots, C$ belong to a given function class, e.g. affine, polynomial, rational, etc.

In the literature, a wide range of LPV conversion methods for general NL-SS models is available, however, many of these methods are either based on complex analytical formulas and restricted to narrow system classes, e.g., [12], [13] or follow a data-driven conversion approach, e.g. [14], [15], which makes them difficult to automate and inefficient for conversion of ANN models. A novel method that has been recently derived in [16] is based on the *Fundamental Theorem of Calculus* (FTC) and, as we show here, can be used to exploit `autograd` tools and parallel computation to recast general ANN-SS models as (4).

The core idea is that if given a continuously differentiable function $g : \mathbb{R}^n \to \mathbb{R}^m$, then based on the FTC [16]:

$$g(\eta) - g(0) = \left(\int_0^1 \frac{dg}{d\eta}(\lambda\eta)\, d\lambda \right) \eta, \tag{5}$$

where $\frac{dg}{d\eta}(\lambda\eta) \in \mathbb{R}^{m\times n}$ is the Jacobian of g evaluated in $\lambda\eta$. Provided that functions f and h in (1) are differentiable, choosing $\eta = [\ \hat{x}^\top \ \ u^\top\ ]^\top$ gives via the FTC that

$$\tilde{f}_{\hat{\theta}}(\hat{x}, u) = \underbrace{\left(\int_0^1 \frac{\partial f}{\partial \hat{x}}(\lambda\hat{x}, \lambda u)\, d\lambda \right)}_{A(\hat{x},u)} \hat{x} + \underbrace{\left(\int_0^1 \frac{\partial f}{\partial u}(\lambda\hat{x}, \lambda u)\, d\lambda \right)}_{B(\hat{x},u)} u,$$
$$\tilde{h}_{\hat{\theta}}(\hat{x}) = \underbrace{\left(\int_0^1 \frac{\partial h}{\partial \hat{x}}(\lambda\hat{x})\, d\lambda \right)}_{C(\hat{x})} \hat{x}, \tag{6}$$

where $\tilde{f}_{\hat{\theta}}(\hat{x}, u) = f_{\hat{\theta}}(\hat{x}, u) - f_{\hat{\theta}}(0, 0)$ with $\tilde{h}_{\hat{\theta}}$ similarly defined. With $p_k = [\ \hat{x}_k^\top \ \ u_k^\top\ ]^\top$ and μ the identity function, this gives an LPV form of the ANN-SS model (2) with affine terms $V = f_{\hat{\theta}}(0, 0)$ and $W = h_{\hat{\theta}}(0)$ as

$$\hat{x}_{k+1} = A(p_k)\hat{x}_k + B(p_k)u_k + V, \tag{7a}$$
$$y_k = C(p_k)\hat{x}_k + W. \tag{7b}$$

While calculating the Jacobians and solving the integrals to obtain $A, \dots, C$ is a difficult task analytically, we will show that for an MPC implementation, these matrix functions are needed to be computed only for specific values of x and u, hence algorithmic differentiation and a suitable numerical integration scheme (e.g., Simpson rule) can be efficiently used for this purpose. Furthermore, for ANNs, where the activation functions are not everywhere differentiable over their argument space, but they are Lipschitz continuous, `autograd` methods still provide a subgradient, see [17]. Next, we show how this LPV surrogate form of the identified ANN-SS model can be used to solve (3) efficiently.

V. Iterative ANN-MPC method

In this section, we develop an iterative optimisation algorithm to solve the ANN-SS model based predictive problem (3) inspired[3] by the so called "quasi" LPV-MPC approach [11]. The core idea is that, at any given time moment k, for a fixed scheduling sequence $\{p_{i+k}\}_{i=0}^{N_\mathrm{p}}$, (7) is used to formulate a linear MPC problem that can be solved rapidly as a QP. Then the resulting control sequence $\{u_{i+k}\}_{i=0}^{N_\mathrm{p}-1}$ is used to forward simulate (1) to compute a new sequence $p_{i+k} = [\ \hat{x}_{i+k}^\top \ \ u_{i+k}^\top\ ]^\top$ on which a new linear MPC problem is formulated and solved. Execution of this iteration multiple times until a converging u_{i+k} trajectory is reached ensures that the LPV representation sufficiently well approximates the NL system along a (locally) optimal solution trajectory of (3) for which the solution of the linear MPC problem coincides with the solution of (3).

A. Reformulation to a quadratic problem

To exploit the above sketched idea, the NMPC problem (3) at time moment $k \in \mathbb{Z}$, for a given fixed scheduling sequence $\{p_{i|k}\}_{i=0}^{N_\mathrm{p}}$ is reformulated as the LPV-MPC problem

$$\min_{\breve{u}_{0|k}^{N_\mathrm{p}-1}} \ \sum_{i=1}^{N_\mathrm{p}} \ell_i(\breve{x}_{i|k} - x_\mathrm{ref}, \breve{u}_{i-1|k} - u_\mathrm{ref}), \tag{8a}$$
$$\text{s.t.} \ \ \breve{x}_{i+1|k} = A(p_{i|k})\breve{x}_{i|k} + B(p_{i|k})\breve{u}_{i|k} + V, \ i \in \mathbb{I}_0^{N_\mathrm{p}-1} \tag{8b}$$
$$\breve{y}_{i|k} = C(p_{i|k})\breve{x}_{i|k} + W, \qquad i \in \mathbb{I}_1^{N_\mathrm{p}}, \tag{8c}$$
$$u_\mathrm{min} \le \breve{u}_{i|k} \le u_\mathrm{max}, \qquad i \in \mathbb{I}_0^{N_\mathrm{p}-1}, \tag{8d}$$
$$y_\mathrm{min} \le \breve{y}_{i|k} \le y_\mathrm{max}, \qquad i \in \mathbb{I}_1^{N_\mathrm{p}}, \tag{8e}$$
$$\breve{x}_{0|k} = x_k. \tag{8f}$$

Using linearity of (8b) and (8c) along $\{p_{i|k}\}_{i=0}^{N_\mathrm{p}}$ gives

$$\breve{X}_{k+1} = \Phi\,(P_k)\,\breve{x}_{0|k} + \Gamma\,(P_k)\,\breve{U}_k + F_0(P_k)V, \tag{9a}$$
$$\breve{Y}_{k+1} = \Lambda\,(P_{k+1})\,\breve{X}_{k+1} + H, \tag{9b}$$

where $\breve{X}_{k+1} = \mathrm{col}(\{\breve{x}_{i|k}\}_{i=1}^{N_\mathrm{p}}) = [\ \breve{x}_{1|k}^\top \ \cdots \ \breve{x}_{N_\mathrm{p}|k}^\top\ ]^\top$ and $\breve{U}_k = \mathrm{col}(\{\breve{u}_{i|k}\}_{i=0}^{N_\mathrm{p}-1})$ with $\breve{Y}_{k+1}$ and P_k similarly defined. $\Phi\,(\mathrm{P}_k)$ and $\Gamma\,(\mathrm{P}_k)$ contain products of $A(p_{i|k})$ and $B(p_{i|k})$ according to the structure discussed in [11], and

$$F_0(P_k) = I_{n_\mathrm{x}} + \begin{bmatrix} 0 & A^\top(p_{1|k}) & \sum_{\tau=1}^{N_\mathrm{p}-1} \prod_{i=1}^{\tau} A^\top(p_{i|k}) \end{bmatrix}^\top$$

where I_{n_x} is an identity matrix of dimension n_x. Furthermore, $\Lambda\,(P_{k+1}) = \mathrm{diag}(C(p_{1|k}), \dots, C(p_{N_\mathrm{p}|k}))$ and $H = \mathrm{col}(W, \dots, W)$. Based on (9), the objective function (8a) can be written as

$$J_k = \breve{U}_k^\top G\,(P_k)\,\breve{U}_k + F^\top(P_k)\,\breve{U}_k, \tag{10}$$

similar to [11], where $G\,(P_k) = 2\left(\Psi + \Gamma^\top(P_k)\,\Omega\Gamma\,(P_k)\right)$ and

$$F(P_k) = 2\left(\Gamma\left(P_k\right)^\top \Omega(\Phi(P_k)\hat{x}_{0|k} + \right.$$
$$\left. + F_0(P_k)V - X_\mathrm{ref}) - \Psi^\top U_\mathrm{ref}\right) \tag{11}$$

[3] An important difference w.r.t. [11] that instead of a linearisation based LPV model, we employ exact LPV conversion based on factorization (6).

with $\Omega = \mathrm{diag}(Q_1, \ldots, Q_{N_\mathrm{p}})$, $\Psi = \mathrm{diag}(R_1, \ldots, R_{N_\mathrm{p}})$, $X_\mathrm{ref} = \mathrm{col}(x_\mathrm{ref}, \ldots, x_\mathrm{ref})$ and $U_\mathrm{ref} = \mathrm{col}(u_\mathrm{ref}, \ldots, u_\mathrm{ref})$. This gives that, along a fixed trajectory of $\{p_{i|k}\}_{i=0}^{N_\mathrm{p}}$, the unconstrained part of the predictive problem for the ANN-SS model can be efficiently recast as a single QP (10) in the decision variables $\breve{U}_k$.

B. Reformulation of the constraints

To add the constraints (8d) and (8e) to the QP form (10) of the optimization problem, they need to be rewritten in the variable $\breve{U}_k$. We start by writing the constraints as

$$M_i \breve{y}_{i+1|k} + E_i \breve{u}_{i|k} \leq b_i, \quad \forall i \in \mathbb{I}_0^{N_\mathrm{p}} \tag{12}$$

where

$$M_i = \begin{bmatrix} 0_{2n_\mathrm{u} \times n_\mathrm{y}} \\ -I_{n_\mathrm{y}} \\ I_{n_\mathrm{y}} \end{bmatrix}, \; E_i = \begin{bmatrix} -I_{n_\mathrm{u}} \\ I_{n_\mathrm{u}} \\ 0_{2n_\mathrm{y} \times n_\mathrm{u}} \end{bmatrix}, \; b_i = \begin{bmatrix} -u_\mathrm{min} \\ u_\mathrm{max} \\ -\hat{y}_\mathrm{min} \\ \hat{y}_\mathrm{max} \end{bmatrix}.$$

Putting (12) together gives

$$\mathcal{M} \breve{Y}_{k+1} + \mathcal{E} \breve{U}_k \leq c, \tag{13}$$

where the matrices are given as $\mathcal{M} = \mathrm{diag}(M_1, \ldots, M_{N_\mathrm{p}})$, $\mathcal{E} = \mathrm{diag}(E_1, \ldots, E_{N_\mathrm{p}})$, and $c = \mathrm{col}(b_1, \ldots, b_{N_\mathrm{p}})$. Substituting (9) into (13) gives

$$L\left(P_k, P_{k+1}\right) \breve{U}_k \leq c + W\left(P_k, P_{k+1}\right), \tag{14}$$

which is a linear constraint in $\breve{U}_k$ with matrices

$$L\left(P_k, P_{k+1}\right) = \mathcal{M} \Lambda\left(P_{k+1}\right) \Gamma\left(P_k\right) + \mathcal{E}, \tag{15a}$$

$$W\left(P_k, P_{k+1}\right) = -\mathcal{M}\left(\Lambda\left(P_{k+1}\right)\left(\Phi\left(P_k\right) \hat{x}_{0|k}\right.\right. \tag{15b}$$

$$\left.\left. + F_0\left(P_k\right) V\right) + H\right). \tag{15c}$$

To avoid feasibility problems due to model inaccuracies, the constraint can also be softened by a slack variable.

C. Iterative algorithm

As we discussed in Sections V-A and V-B, the original NMPC problem (3) can be efficiently reformulated as the QP (10) with a linear constraint (14) along a given scheduling sequence $\{p_{i|k}\}_{i=0}^{N_\mathrm{p}}$. If this scheduling sequence would be taken as the optimal solution of (3), i.e., $p_{i|k} = \mu(\hat{x}_{i|k}^\mathrm{NL}, \hat{u}_{i|k}^\mathrm{NL})$, the optimal solution of (8) would coincide with $\{\hat{x}_{i+1|k}^\mathrm{NL}, \hat{u}_{i|k}^\mathrm{NL}\}_{i=0}^{N_\mathrm{p}-1}$. Hence, by starting from an initial sequence of $\{\hat{x}_{i+1|k}, \hat{u}_{i|k}\}_{i=0}^{N_\mathrm{p}-1}$ used to compute $\{p_{i|k}\}_{i=0}^{N_\mathrm{p}}$, the idea is to converge through successive LPV formulations of the NL problem towards the optimal solution of (3) by iteratively solving (8) via (10) with (14) and use the obtained input sequence $\{\hat{u}_{i|k}\}_{i=0}^{N_\mathrm{p}-1}$ to simulate (2) and, based on the resulting $\{\hat{x}_{i|k}\}_{i=1}^{N_\mathrm{p}}$, update $\{p_{i|k}\}_{i=0}^{N_\mathrm{p}}$. The iteration is continued till a converging input trajectory is achieved and then $u_{0,k}$ is applied to the system in a receding horizon sense. The resulting iterative scheme is summarized in Algorithm 1. Iterations often quickly converge in practice and after the computation of the first control cycle at $k = 0$, the next cycles can be bootstrapped by taking the optimal trajectory obtained in the previous cycle to initialise p, i.e., $\{p_{i|k} = \mu(\breve{x}_{i+1|k-1}, \breve{u}_{i+1|k-1})\}_{i=0}^{N_\mathrm{p}-2}$, $p_{N_\mathrm{p}-1|k} = \mu(\breve{x}_{N_\mathrm{p}|k-1}, \breve{u}_{N_\mathrm{p}-1|k-1}) = p_{N_\mathrm{p}|k}$. Note that $p_{N_\mathrm{p}|k}$ is only used to compute C in (8c) for (8e) and C is only dependent on the x part of p.

Algorithm 1 Iterated LPV-MPC solution of (3)

1: **initialization**: Let $k = 0$ and $\{\hat{x}_{i|k} = \hat{x}_0, \hat{u}_{i|k} = 0\}_{i=0}^{N_\mathrm{p}}$
2: **while** $k \leq N_\mathrm{sim}$ **do**
3: **repeat**
4: **update** $p_{i|k}$ by $\mu(\hat{x}_{i|k}, \hat{u}_{i|k})$ for $i \in \mathbb{I}_{i=0}^{N_\mathrm{p}}$
5: **calculate** $A(p_{i|k}), \ldots, C(p_{i|k})$ via (6)
6: **solve** (8) to obtain $\{\breve{u}_{i|k}\}_{i=0}^{N_\mathrm{p}-1}$
7: **simulate** (2) with $\hat{u}_{i|k} \leftarrow \breve{u}_{i|k}$ to obtain $\{\hat{x}_{i|k}\}_{i=1}^{N_\mathrm{p}}$
8: **until** $\{\hat{u}_{i|k}\}_{k=0}^{N_\mathrm{p}-1}$ has converged
9: **apply** $u_k = \hat{u}_{1|k}$ and observe $\hat{x}_{k+1}$
10: **let** $k \leftarrow k + 1$
11: **propagate** $(\hat{x}_{i|k-1}, \hat{u}_{i|k-1})$ to initialize $(\hat{x}_{i|k}, \hat{u}_{i|k})$
12: **end while**

D. Target selection

In order to realize a set-point tracking objective for the ANN-SS model based MPC problem (3), $(x_\mathrm{ref}, u_\mathrm{ref})$ are required to be determined. In case the objective is defined in terms of a reference $r \in \mathbb{R}^{n_\mathrm{y}}$ that the output needs to follow, the corresponding $(x_\mathrm{ref}, u_\mathrm{ref})$ setpoint can be determined via a target selector, for example the method discussed in [11]:

$$\min_{x_\mathrm{ref}, u_\mathrm{ref}} \frac{1}{2} \left\| C\left(p_\mathrm{ref}\right) x_\mathrm{ref} + W - r \right\|_Q^2 + \left\| u_\mathrm{ref} \right\|_R^2 \tag{16a}$$

subject to

$$\begin{bmatrix} I - A\left(p_\mathrm{ref}\right) & -B\left(p_\mathrm{ref}\right) \\ C\left(p_\mathrm{ref}\right) & 0 \end{bmatrix} \begin{bmatrix} x_\mathrm{ref} \\ u_\mathrm{ref} \end{bmatrix} = \begin{bmatrix} V \\ r - W \end{bmatrix} \tag{16b}$$

$$y_\mathrm{min} \leq C\left(p_\mathrm{ref}\right) x_\mathrm{ref} + W \leq y_\mathrm{max} \tag{16c}$$

$$u_\mathrm{min} \leq u_\mathrm{ref} \leq u_\mathrm{max}, \tag{16d}$$

where (16) can be solved as a QP iteratively by following the procedure in Section V-A and exploiting that $p_\mathrm{ref} = \mu(x_\mathrm{ref}, u_\mathrm{ref})$. Due to the efficient QP formulation, the selector can be also executed online in case r varies with k.

E. Encoder based state observer

To solve the optimization problem (8), the initial state $\hat{x}_{0|k} = \hat{x}_k$ is required. However, from the real system (1), we can only obtain a noise-contaminated output measurement y_k. Therefore we require an observer that estimates $\hat{x}_k$ from past values of u_k, and the past and current values of y_k. For this purpose, various methods can be applied, but if the SS-ANN has been obtained via the SUBNET method, then we can make use the estimated encoder for the model, see Figure 1. This encoder is trained under noisy data such that the multiple shooting based prediction loss of the entire model is minimized under noisy measurements of y_k. Hence the trained encoder provides filtering and state reconstruction for a reliable estimate of $\hat{x}_k$ and can be used to calculate the initial state as $\breve{x}_{0|k} = \psi_{\hat{\theta}}(u_{k-n}^{k-1}, y_{k-n}^{k})$ in (8f).

F. Computational structure

To set up the QP at each k time moment, the matrices required for (10) and (14) are needed to be rapidly computed, which involves computation of $A(p_{i|k}), B(p_{i|k}), C(p_{i|k})$ in terms of (6) along a given $\{p_{i|k}\}_{i=0}^{N_\mathrm{p}}$. The involved integrals are computed by a numerical integration scheme (e.g., Simpson rule) corresponding to n evaluation of Jacobians of the ANNs composing the model. Hence, in every iteration in Algorithm 1, $N_\mathrm{p} \cdot n$ Jacobians are needed to be determined. Due to the lack of dependence between Jacobian calculation, this can be done through `autograd` methods effectively, and the corresponding n calculations of the Jacobians can be multi-threaded resulting in a highly efficient computational scheme shown in Figure 2.

G. Convergence properties

In the literature, it has been established that the iteratively solved LPV-MPC scheme is equivalent to standard Newton-based *sequential quadratic progrmaming* (SQP) if the LPV model (7) is obtained using purely Jacobian linearization. This corresponds to the case when, in (6), the integration is dropped and the corresponding A, B, C matrices are approximated only by the Jacobian evaluated at $(\hat{x}_k, u_k)$ [18]. Under this simplification, it is sufficient to borrow results from the analysis of Newton SQPs [19], which are locally convergent if all inequality constraints are predefined. In our algorithm, the latter condition is satisfied by the inequality constraints (8d) and (8e). Furthermore, for space limitations, the cost function (8a) does not contain terminal ingredients, but they can be easily incorporated in the problem. In that case, a feasible terminal set would have to be defined to ensure convergence. Extension of the Newton SQP properties for the exact factorisation provided by (6) are expected to hold, while, due to lack of approximation error in (6), this novel variant of SQP is expected to have a faster convergence rate.

VI. Example

Consider a simulator of a nonlinear unbalanced disc system [20]. The motion model of the system is

$$\dot{\theta}_t = \omega_t, \tag{17a}$$

$$\dot{\omega}_t = -\frac{Mgl}{J}\sin(\theta_t) - \frac{1}{\tau}\omega_t + \frac{K_\mathrm{m}}{\tau}v_t, \tag{17b}$$

$$y_k = \theta_{kT_\mathrm{s}} + e_k, \tag{17c}$$

where $t \in \mathbb{R}$ is the continous time, θ is the angle of the disk w.r.t. the bottom position and measured clockwise, ω is the angular velocity of the disk, v is the system input in V actuated in a *zero-order hold* (ZOH) sense, y is the DT output measurement with sampling time $T_\mathrm{s} = 0.1$ while e is a discrete white noise process with $e_k \sim \mathcal{N}(0, \sigma_\mathrm{e}^2)$. The physical parameters $M, \ldots, K_\mathrm{s}$ in (17) are taken from [20].

By applying RK4-based numerical integration with T_s, (17) is simulated for a ZOH input signal v_t generated by a DT multisine u_k with 16666 components in the range $[0, 5]$ Hz with a uniformly distributed phase in $[0, 2\pi]$. The resulting u is limited to an amplitude of 3 V to prevent the unbalanced disc from going over the top. The sampled output measurements y_k are perturbed by e_k resulting in a *signal-to-noise ratio* (SNR) of 30 dB. This provides a data sequence $\mathcal{D}_N$ of size $N = 10^5$. The data sequence is split into estimation, validation, and test data sets with size $N_\mathrm{est} = 6 \cdot 10^4$, $N_\mathrm{val} = 2 \cdot 10^4$, $N_\mathrm{test} = 2 \cdot 10^4$, respectively.

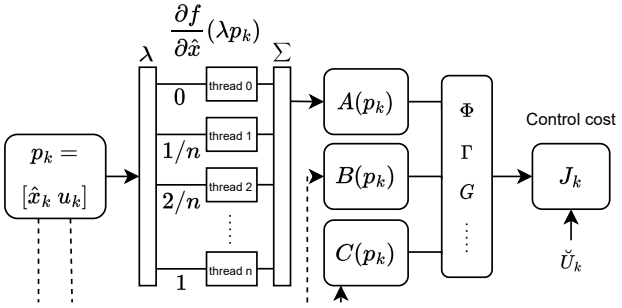

Fig. 2. Rapid calculation of the LPV MPC objective function J_k in (10) is executed via autograd calculation of the gradients and numerical evaluation of the integrals in (6) using multi-threading. Each thread computes an evaluation of the derivative term for a value of λ in the numerical integration scheme and afterward, the results of these threads are efficiently summed together to compute each matrix in J_k.

TABLE I

Hyperparameters for identifying the ANN-SS model.

hidden layers	nodes	n_x	n_u n_y	n_a n_b	epochs	batch size
2	64	2	1	4	250	256

An ANN-SS model is identified as in [21] with the hyperparameters shown in Table I. The resulting model has *normalized root mean square* error of 3.2% on the test data set, corresponding to an accurate DT model of (17).

For the MPC, we select $Q_i = \mathrm{diag}(10^3, 10)$, $R_i = 1$, with $-\hat{y}_\mathrm{min} = \hat{y}_\mathrm{max} = 1.2$, $-u_\mathrm{min} = u_\mathrm{max} = 4$, and control horizon $N_\mathrm{p} = 10$. LPV conversion (6) for calculation of the MPC is set up by using Simpson's rule for integration and `functorch` [22] for differentiation with $d\lambda = 0.05$ as the integration step size. The QP problem in terms of minimization of (10) under (14) is solved with `osqp` [23]. Convergence in Step 8 of Algorithm 1 is defined as $\|\breve{U}_k^{(j)} - \breve{U}_k^{(j-1)}\|_2 \leq 0.1$ with $\breve{U}_k^{(j)}$ denoting the optimal solution of (10) under (14) at iteration j.

As a comparison, we use ipopt [9] with `CasADi` and SQP with `acados` [2] to solve the NMPC problem (3) with the same weighting matrices Q and R. Fig. 3 shows the response of system (17) controlled by the proposed MPC algorithm, ipopt, and SQP using the identified ANN-SS model, while tracking a piece-wise constant reference signal for the disk angle. The proposed MPC solution provides reliable tracking of the reference and when compared to the ipopt and SQP methods, the difference between the obtained responses is relatively small. We can also see that all OCP controllers are capable of tracking the reference as long as the input required is within the region on which the encoder and the ANN-SS model were trained. If the required input falls outside this region, a small steady-state error occurs. This can be explained by model extrapolation errors. Generally, the same extrapolation problem is expected from any identified black-box model on the considered data set.

While the proposed LPV-MPC, ipopt and SQP methods

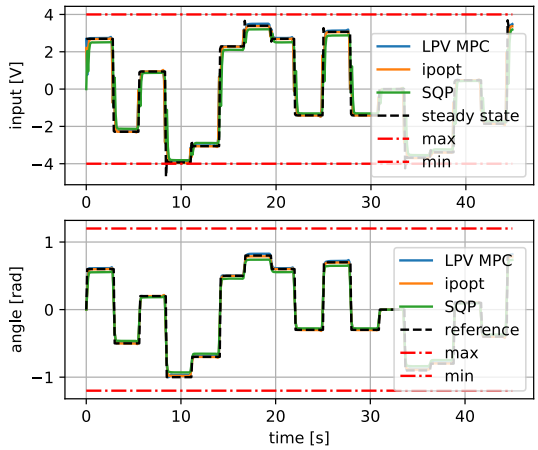

Fig. 3. Reference tracking of the unbalanced disk with the proposed LPV-MPC algorithm (Algorithm 1) compared to ipopt and SQP based solutions of the NMPC problem (3) using the identified ANN-SS model. The plot shows the piece-wise constant reference, the noiseless output of (17), the resulting input commands, and the input and output constraints.

TABLE II
COMPUTATIONAL TIME
(INTEL(R) CORE(TM) I7-7700HQ CPU @ 2.80GHZ)

	Max time	Average time	St. dev.	Solver time
LPV-MPC	65.0 ms	14.1 ms	7.07 ms	4.79 ms
CasADi ipopt	100 ms	18.1 ms	5.75 ms	16.3 ms
acados SQP	3.92 ms	1.16 ms	0.586 ms	0.0659 ms

perform similarly, the required computation times, given in Table II, show that the proposed QP iterations are much more efficient than the `ipopt` solution, while the embedded SQP solution is much faster than the other methods. Furthermore, out of the 450 simulation steps, 338 required only a single iteration, 110 had two iterations, and only 2 required three steps to converge. The overall computation time is dominated by the LPV conversion, while the solver takes relatively little time. For a better comparison between methods, an embedded implementation should be used for both the proposed MPC and ipopt. An embedded implementation of the LPV-MPC algorithm is expected to have a computation time close to the SQP implementation, due to the main computation burden being similar for both algorithms. For SQP this is the linearization of the continuity equation [2] around a trajectory, while for the LPV-MPC this is the auto-conversion. However, the auto-conversion avoids linearization error of the SQP, potentially allowing for faster convergence with only a slight increase of the computational time of a iteration which can be mitigated as discussed in Section V-F.

VII. CONCLUSIONS

We have proposed an efficient iterative LPV-MPC solution starting from identified ANN-SS models, based on two key ingredients: (i) approximation-free conversion of ANN-SS models to an LPV form through the Fundamental Theorem of Calculus, which can be efficiently computed via autograd methods and multi-threaded numerical integration, (ii) direct formulation of successive QP problems using the obtained optimal solution from the previous iteration step as the scheduling for the LPV prediction model. We have shown that the computational load of the proposed MPC solution is lower than a general NMPC method applied with the ANN-SS model thanks to the approximation-free LPV form and the computational toolchain that profits from the available autograd tools. The approach also benefits from the encoder structure provided by recent ANN-SS estimation methods, which enables reliable initial state calculation for the MPC scheme using noisy input-output measurements.

REFERENCES

[1] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Pub., 2017.

[2] R. Verschueren, et. al., "Acados—a Modular Open-Source Framework for Fast Embedded Optimal Control," *Math. Prog. Comp.*, 2022.

[3] D. Limon, A. Ferramosca, I. Alvarado, and T. Alamo, "Nonlinear MPC for tracking piece-wise constant reference signals," *IEEE Trans. Aut. Cont.*, vol. 63, p. 3735–3750, 2018.

[4] J. A. K. Suykens, B. L. R. De Moor, and J. Vandewalle, "Nonlinear system identification using neural state space models, applicable to robust control design," *Int. J. of Control*, vol. 62, no. 1, pp. 129–152, 1995.

[5] G. I. Beintema, M. Schoukens, and R. Tóth, "Deep subspace encoders for nonlinear system identification," *Automatica*, vol. 156, p. 111210, 10 2023. [Online]. Available: http://arxiv.org/abs/2210.14816

[6] M. Forgione and D. Piga, "Continuous-time system identification with neural networks: Model structures and fitting criteria," *Eu. J. of Cont.*, vol. 59, pp. 69–81, 2021.

[7] M. Lazar and O. Pastravanu, "A neural predictive controller for non-linear systems," *Math. and Comp. in Sim,*, vol. 60, pp. 315–324, 2002.

[8] M. Lawrynczuk, *Computationally Efficient Model Predictive Control Algorithms: A Neural Network Approach*. Springer, 2014.

[9] A. Wächter and L. Biegler, "On the implementation of an interior-filter line-search algorithm for large-scale nonlinear programming," *Math. Prog.*, vol. 106, pp. 25–56, 2006.

[10] D. Masti and A. Bemporad, "Learning nonlinear state–space models using autoencoders," *Automatica*, vol. 129, p. 109666, 2021.

[11] P. Cisneros, "Quasi-linear model predictive control: Stability, modelling and implementation," Ph.D. dissertation, Technical University Hamburg, 2021.

[12] S. M. Hashemi, H. S. Abbas, and H. Werner, "Low-complexity linear parameter-varying modeling and control of a robotic manipulator," *Con. Eng. Pract.*, 2012.

[13] A. Marcos and G. J. Balas, "Development of linear-parameter-varying models for aircraft," *J. of Guid., Cont. and Dyn.*, vol. 27, no. 2, pp. 218–228, 2004.

[14] A. Kwiatkowski and H. Werner, "PCA-based parameter set mappings for LPV models with fewer parameters and less overbounding," *IEEE Trans. on Cont. Sys. Tech.*, vol. 16, no. 4, pp. 781–788, 2008.

[15] A. Sadeghzadeh, B. Sharif, and R. Tóth, "Affine linear parameter-varying embedding of non-linear models with improved accuracy and minimal overbounding," *IET Cont. Theory & App.*, vol. 14, pp. 3363–3373, 2020.

[16] P. J. Koelewijn, "Analysis and control of nonlinear systems with stability and performance guarantees: A linear parameter-varying approach," Ph.D. dissertation, Eindhoven University of Technology, 2023.

[17] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[18] C. Hespe and H. Werner, "Convergence properties of fast quasi-LPV model predictive control," in *Proc. of the 60th IEEE Conf. on Decision and Cont.*, 2021, pp. 3869–3874.

[19] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, p. 1–51, 1995.

[20] B. Kulcsár, J. Dong, J. W. van Wingerden, and M. Verhaegen, "LPV subspace identification of a DC motor with unbalanced disc," in *Proc. of the 15th IFAC Symp. on Sys. Id.*, 2009, pp. 856–861.

[21] G. Beintema, R. Toth, and M. Schoukens, "Nonlinear state-space identification using deep encoder networks," in *Proc. of the 3rd Conf. on Learn. for Dyn. and Cont.*, ser. Proceedings of Machine Learning Research, vol. 144, 2021, pp. 241–250.

[22] R. Z. Horace He, "functorch: Jax-like composable function transforms for pytorch," https://github.com/pytorch/functorch, 2021.

[23] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Math. Prog. Comp.*, vol. 12, no. 4, pp. 637–672, 2020.