Proceedings of Machine Learning Research vol 242:349–360, 2024 6th Annual Conference on Learning for Dynamics and Control

# Parameter-Adaptive Approximate MPC: Tuning Neural-Network Controllers without Retraining

**Henrik Hose** HENRIK.HOSE@DSME.RWTH-AACHEN.DE
**Alexander Gräfe** ALEXANDER.GRAEFE@DSME.RWTH-AACHEN.DE
**Sebastian Trimpe** TRIMPE@DSME.RWTH-AACHEN.DE
*Institute for Data Science in Mechanical Engineering (DSME) RWTH Aachen University, Germany*

**Editors:** A. Abate, M. Cannon, K. Margellos, A. Papachristodoulou

## Abstract

Model Predictive Control (MPC) is a method to control nonlinear systems with guaranteed stability and constraint satisfaction but suffers from high computation times. Approximate MPC (AMPC) with neural networks (NNs) has emerged to address this limitation, enabling deployment on resource-constrained embedded systems. However, when tuning AMPCs for real-world systems, large datasets need to be regenerated and the NN needs to be retrained at every tuning step. This work introduces a novel, parameter-adaptive AMPC architecture capable of online tuning without recomputing large datasets and retraining. By incorporating local sensitivities of nonlinear programs, the proposed method not only mimics optimal MPC inputs but also adjusts to known changes in physical parameters of the model using linear predictions while still guaranteeing stability. We showcase the effectiveness of parameter-adaptive AMPC by controlling the swing-ups of two different real cartpole systems with a severely resource-constrained microcontroller (MCU). We use the same NN across both system instances that have different parameters. This work not only represents the first experimental demonstration of AMPC for fast-moving systems on low-cost MCUs to the best of our knowledge, but also showcases generalization across system instances and variations through our parameter-adaptation method. Taken together, these contributions represent a marked step toward the practical application of AMPC in real-world systems.

**Keywords:** Nonlinear model predictive control, approximate MPC, sensitivity, machine learning

## 1. Introduction

Model predictive control (MPC) is an optimization-based control strategy for nonlinear systems with favorable properties like theoretically guaranteed stability and constraint satisfaction (Rawlings et al., 2017). These properties come at the cost of long computation times for solving nonlinear programs, even on modern, high-performance processors. In practice, this limits applications of MPC, as powerful and expensive hardware is not always available. Therefore, in recent years, approximate MPC (AMPC) with neural networks (NNs) has gained increasing interest (Gonzalez et al., 2023). In AMPC, the behavior of the MPC is imitated by a NN that is often small enough to be evaluated within milliseconds on a microcontroller (MCU). This unlocks the potential of MPC on small, resource-constrained, embedded systems. However, few publications have demonstrated NN approximations of MPCs for fast-moving systems with update rates at the order of milliseconds on low-cost MCUs with real hardware experiments (cf. Sec. 2.1). A particular challenge when deploying AMPC in real-world applications is tuning the AMPC on the physical system to achieve satisfactory performance. This usually involves changing parameters in the MPC (e.g., system model,

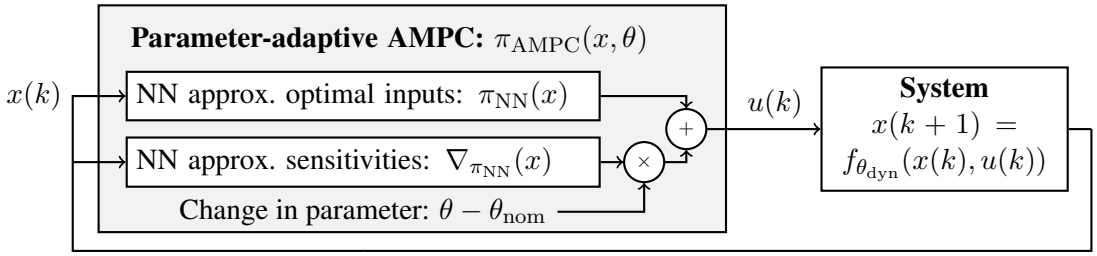


Figure 1: Parameter-adaptive AMPC. *Approximate nominal MPC inputs are linearly adapted to true parameters $\theta$ by approximate sensitivities. The parameters can include parameters of the dynamics model $\theta_{\text{dyn}}$ and other MPC parameters, like weights of the cost function.*

cost function) and retraining the NN approximation. However, every tuning iteration potentially takes days to recompute the dataset. This is because the dataset easily contains tens of thousands to millions of samples from the MPC where each datum corresponds to one solution of the MPC optimization. As a consequence, the whole tuning process can take weeks. While this might be cumbersome yet possible in a research context (Nubert et al., 2020; Carius et al., 2020; Abu-Ali et al., 2022; Leonow et al., 2023), it limits practical applicability of AMPC in real products. This is even more prohibitive when parameters vary slightly for each system instance and the AMPC needs to be retuned every time (Adhau et al., 2019).

**Contribution.** In this paper, we address the issue of cumbersome AMPC tuning by proposing a method that allows for tuning without regenerating a dataset and without retraining the NN. We achieve this using local sensitivities of nonlinear programs with respect to its parameters. The proposed AMPC architecture (see Fig. 1) not only imitates the optimal inputs of the MPC, but also its sensitivities. The architecture allows us to adjust the controller's output online with linear predictions. Thus, the AMPC accounts for time-invariant, known changes in parameters of the dynamics model such as mass or friction, and also others like cost or constraint parameters of the MPC. In summary, we make the following contributions to the field of AMPC.

1. We propose an AMPC architecture which can adjust the output of the trained NN to time-invariant, known changes in dynamics and MPC design parameters using a linear predictor with approximations of the local sensitivities.
2. We derive conditions for the approximation error and maximum parameter change under which the controlled system is stable.
3. We implement the method on a small MCU (STM32G474 running at 170MHz with 96kb of SRAM) that costs only a few dollars and show that it can swing up and balance a real cartpole system. To the best of our knowledge, this is the first real-world implementation of AMPC on a severely resource-constrained, general-purpose MCU controlling a fast and unstable nonlinear system.
4. In hardware experiments, we demonstrate that the parameter-adaptive AMPC generalizes to two cartpole system instances with different parameters without retraining the NN, whereas a naive nominal AMPC fails to generalize.

A video of our experiments is available at https://youtu.be/o1RdiYUH9uY.

## 2. Related Work

This section summarizes essential related work in the context of this paper, divided into three parts. The first part discusses important methods for AMPC and existing implementations of AMPC in embedded systems, highlighting the limited usage of small MCUs for controlling fast and nonlinear real systems. The second part summarizes the few AMPC methods that could include dynamics model variations, highlighting the general lack of parameter-adaptive AMPC methods. The final part reviews prior works on MPC that leverage sensitivities, but for different purposes than dynamics model parameter adaptation as herein.

### 2.1. Approximate model-predictive control

In the past decades, various methods have been proposed to avoid online optimization in embedded MPC. In simple lookup methods, an offline-computed table of optimal inputs is used for online interpolation. Even using the nearest neighbor can perform comparable to imitation learning methods (Florence et al., 2022) and can have theoretical guarantees for robust MPC (Bayer et al., 2016). Through multi-parametric programming, explicit MPC solutions are attainable exactly for (small) linear (Bemporad et al., 2002; Alessio and Bemporad, 2009) and approximately for some nonlinear systems (Johansen, 2004; Bemporad and Filippi, 2006), for which convenient software packages for stand-alone C-code export for MCUs exist (Kvasnica et al., 2015). However, solving multi-parametric programs is complex for general nonlinear and even medium-sized linear systems and lookup tables grow exponentially with the number of optimization variables, making them impractical for MCUs with limited memory.

Possible alternatives that efficiently scale with system dimensions include approximations with NNs, a concept explored for decades (Parisini et al., 1998; Åkesson and Toivonen, 2006). More recently, research focused on theoretical guarantees, imitation learning methods, and validation of NN approximations, cf. a recent overview by Gonzalez et al. (2023). However, only a handful of publications have successfully implemented NN approximations of MPCs in practical hardware experiments, such as robot arm tracking (Nubert et al., 2020), electric motor control (Abu-Ali et al., 2022), or quadruped robot control (Carius et al., 2020); notably, all using laptop-grade CPUs for inference, which would, in principle, also be suitable for real-time optimization.

On the other side, works focussing on the implementation of AMPC on small embedded devices validated their results on simulated plants (Lucia et al., 2020; Wang et al., 2021; Chan et al., 2021; Adhau et al., 2019; Karg and Lucia, 2019). To the best of our knowledge, Leonow et al. (2023) and Xiang et al. (2024) present the only applications of AMPC with NNs on MCUs controlling actual physical systems, albeit with slow dynamics and a control frequency of $2\,\mathrm{Hz}$ in the first and a linear system in the latter case. In this paper, we implement and experimentally test AMPC (evaluated in less than $2\,\mathrm{ms}$) on MCUs controlling a fast physical system (control frequency of $20\,\mathrm{Hz}$).

### 2.2. AMPC with parameter variation

When bringing control to real physical systems, controllers usually need to be fine-tuned on the real plant to achieve optimal performance. In classic MPC, this can be done automatically (Forgione et al., 2020; Paulson et al., 2023) in closed-loop experiments. For AMPC, however, such tuning involves recomputing a large dataset and retraining an NN, a limitation specifically mentioned by Adhau et al. (2019). While AMPC can imitate a robust MPC policy (Nubert et al., 2020) that could

also account for dynamics model parameter variations (Köhler et al., 2020), such robustification schemes are generally involved for nonlinear systems and introduce conservatism. Other AMPC methods that were not originally intended for tuning could be used to adapt to dynamics parameter changes, e.g., warm-starting online optimization (Klaučo et al., 2019; Chen et al., 2022) or retraining the NN in a reinforcement-learning fashion, similar to Bogdanovic et al. (2022). However, implementing them on small MCUs is impractical and online optimization with NN warm-starting can be slow (Vaupel et al., 2020). We overcome this issue by using sensitivities of nonlinear programs to approximately correct the AMPC with a linear prediction. The presented method is fast to compute on MCUs and intuitive to tune due to physical interpretation of dynamics model parameters.

### 2.3. Sensivitiy-based nonlinear MPC

The method proposed in this paper uses the sensitivities of the nonlinear MPC problem to predict online how the optimal inputs change in response to parameter variations. In this section, we briefly introduce sensitivities of nonlinear programs (NLPs) first and then summarize some related works that leverage sensitivities for fast adaptations of MPCs to changes in initial state.

A nonlinear MPC problem is a parametric NLP with parameters $p$. In sensitivity MPC, these parameters are typically the initial state (Büskens and Maurer, 2001) but could also comprise other dynamics model parameters as in this paper, or even cost or constraint parameters. A parametric NLP in standard form with cost function $V$, constraints $g$, and optimization variables $u$ is

$$u^*(p) = \arg\min_u \; V(u,p) \quad \text{s.t.} \quad g(u,p) \leq 0. \tag{1}$$

We call the constraints for which $g(u^*,p) = 0$ active constraints. The classic results of Fiacco (1976) allow applying the implicit function theorem to the gradient of the KKT conditions, which can be used to get first order predictions of the optional solution

$$u^*(p_0 + \Delta p) = u^*(p_0) + \frac{\partial}{\partial p} u^*(p)\Big|_{p=p_0} \Delta p + \mathcal{O}(\|\Delta p\|^2). \tag{2}$$

This holds as long as the set of active constraints does not change, in which case a quadratic program would need to be solved (Kadam and Marquardt, 2004). The gradient $\frac{\partial u^*}{\partial p}$ can be used as a linear predictor of the optimal solution to (1) for parameter variations. Sensitivities are available in some nonlinear optimization software packages used for MPC, e.g., Pyomo (Bynum et al., 2021), CasADi (Andersson and Rawlings, 2018), or acados (Verschueren et al., 2022).

The sensitivity with respect to the initial state has been used extensively in MPC. Classic sensitivity MPC uses it to perform a fast online correction for a precomputed input sequence (Büskens and Maurer, 2001; Diehl et al., 2002); also when active constraints change (Kadam and Marquardt, 2004). In the advanced-step MPC (Zavala and Biegler, 2009; Yang and Biegler, 2013; Jäschke et al., 2014), a nonlinear MPC is solved online for a predicted future initial state. Once numerical optimization concludes, the solution is adjusted to the measured initial state using sensitivities.

Krishnamoorthy (2021, 2023) uses sensitivities with respect to the initial state in AMPC for data augmentation. Linear (offline) predictions around sample points extend the dataset to enhance the accuracy of the NN at a marginal increase in computation times. Similarly, Lüken et al. (2023) use the sensitivities directly in the training with a loss on the gradient to improve prediction accuracies.

While these works indicate that sensitivities can be used for fast online adjustment to changes in the initial state, none of them uses sensitivities to account for changes in dynamics and other MPC parameters as we propose herein.

## 3. Parameter-adaptive AMPC

This section first introduces the nonlinear MPC formulation that is approximated. We then describe the proposed parameter-adaptive AMPC and provide theoretical results on its stability.

We consider general, nonlinear, discrete time dynamical systems

$$x(k+1) = f_{\theta_{\text{dyn}}}(x(k), u(k)), \quad x(0) = x_0, \tag{3}$$

where $k \in \mathbb{N}$ is the discrete time, $x \in \mathbb{R}^n$ the state, $x_0 \in \mathbb{R}^n$ the initial state, $u \in \mathbb{R}^m$ the input, and $f_{\theta_{\text{dyn}}} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ the continuous dynamics function with time-invariant dynamics model parameters $\theta_{\text{dyn}} \in \Theta_{\text{dyn}} \subseteq \mathbb{R}^{q_{\text{dyn}}}$, where $\Theta_{\text{dyn}}$ is the set of possible parameter values. We consider the dynamics parameters to be time invariant, representing, e.g., different instances of the same system class. We consider a nominal system (3) without external disturbances, as is often done in practice, e.g., for approximate MPC by Carius et al. (2020). However, the same parameter-adaption method would also extend to robust AMPC with external disturbances, e.g., by Nubert et al. (2020).

### 3.1. Input robust model predictive control

A standard, nonlinear MPC formulation (Rawlings et al., 2017) for controlling (3) is

$$\begin{aligned} & u_\theta^* = \arg\min_u \textstyle\sum_{\kappa=0}^{N} \ell_\theta(\kappa, x(\kappa|k), u(\kappa|k)) \\ \text{s.t. } & x(0|k) = x(k), \quad x(\kappa+1|k) = f_{\theta_{\text{dyn}}}(x(\kappa|k), u(\kappa|k)), \\ & x(\kappa|k) \in \mathcal{X}_\theta(\kappa), \quad u(\kappa|k) \in \mathcal{U}_\theta(\kappa) \quad \forall \kappa = 0 \dots N, \end{aligned} \tag{4}$$

where $\ell_\theta : \mathbb{N} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is a cost function, and $\mathcal{X}_\theta(\kappa) \subseteq \mathbb{R}^n$ and $\mathcal{U}_\theta(\kappa) \subseteq \mathbb{R}^m$ are the state and input constraints, respectively. The parameter vector $\theta \in \Theta \subseteq \mathbb{R}^q$ contains the parameters of the dynamical system $\theta_{\text{dyn}}$ and possibly other parameters, like weights in the objective function. The set $\Theta$ is the set of all possible parameter values. The first element $u_\theta^*(0|k)$ of the optimal input trajectory $u_\theta^*$ is then applied to the plant. We call this controller $\pi_{\text{MPC}}(x(k), \theta) := u_\theta^*(0|k)$. In practice, there are software packages to formulate and pass the optimization problem (4) to NLP solvers to compute the optimal control trajectory $u^*$. Some software packages (cf. Sec. 2.3) also provide sensitivities for the optimal controls, i.e., gradients $\frac{\partial}{\partial \theta} \pi_{\text{MPC}}(x, \theta)\big|_{\theta_{\text{nom}}}$, which we shall use in the following. We further assume that the MPC (4) stabilizes the plant under input disturbances $d(k) \in \mathbb{R}^m$, i.e., $u(k) = \pi_{\text{MPC}}(x(k), \theta) + d(k)$. The input disturbances model approximation errors of the AMPC, not external disturbances or noise (note that (3) is the nominal system), as similarly proposed by Hertneck et al. (2018). The general formulation of an MPC (4) includes MPCs with or without terminal state costs and constraints. We do not further specify the structure of the MPC, a potential robustification scheme, cost function, and constraints. These depend on the application and their choice is orthogonal to the proposed approximation method. For the following derivations, we require a robust MPC:

**Assumption 1** *There exists a stability notion, a vector norm, a compatible matrix norm $||\cdot||$, and a maximum allowable input disturbance $\eta > 0$ such that it holds for all $\theta \in \Theta$: if for all $k \geq 0$ $||d(k)|| \leq \eta$, the controlled system is stable under this notion.*

An overview of robust MPC methods that can satisfy this assumption is presented by Houska and Villanueva (2019). In the context of AMPC, Hertneck et al. (2018) and Nubert et al. (2020) use an input-robust MPC (cf. Köhler et al. (2020) for a more general version) that can, in principle, satisfy this assumption.

### 3.2. Approximate model predictive control with parameter changes

A general idea of AMPC is to approximate $\pi_{\mathrm{MPC}}$ via supervised learning from a large dataset of samples $(x_i, \pi_{\mathrm{MPC}}(x_i, \theta))$. As the MPC depends on $\theta$, we consider a nominal $\theta_{\mathrm{nom}} \in \Theta$ for training. For example, $\theta_{\mathrm{nom}}$ could be chosen as the center of the relevant set $\Theta$. The learned mapping for nominal dynamics parameters is called $\pi_{\mathrm{NN}} : \mathbb{R}^n \to \mathbb{R}^m$. The approximation error of $\pi_{\mathrm{NN}}$ is $e_\pi(x) := \pi_{\mathrm{NN}}(x) - \pi_{\mathrm{MPC}}(x, \theta_{\mathrm{nom}})$. In addition to approximating the optimal controls by the MPC, as is standard in AMPC, we propose to train an additional NN offline via supervised learning to approximate the sensitivities from samples $(x_i, \frac{\partial}{\partial \theta}\pi_{\mathrm{MPC}}(x_i, \theta)\big|_{\theta_{\mathrm{nom}}})$. These sensitivities are available from some NLP solvers (cf. Sec. 2.3). We call the NN that approximates sensitivities $\nabla_{\pi_{\mathrm{NN}}}(x) : \mathbb{R}^n \to \mathbb{R}^{m \times p}$. The approximation error of this is $e_\nabla(x) := \nabla_{\pi_{\mathrm{NN}}}(x) - \frac{\partial}{\partial \theta}\pi_{\mathrm{MPC}}(x, \theta)\big|_{\theta_{\mathrm{nom}}}$. With the two NN approximations, $\pi_{\mathrm{NN}}(x)$ and $\nabla_{\pi_{\mathrm{NN}}}(x)$, we propose the parameter-adaptive AMPC (see Fig. 1):

$$\pi_{\mathrm{AMPC}}(x, \theta) = \pi_{\mathrm{NN}}(x) + \nabla_{\pi_{\mathrm{NN}}}(x)(\theta - \theta_{\mathrm{nom}}). \tag{5}$$

This parameter-adaptive AMPC (5) uses the approximated optimal controls for the nominal parameters and linearly predicts how these controls change for the real parameters $\theta$. The linear predictor is given by the approximate sensitivities $\nabla_{\pi_{\mathrm{NN}}}(x)$.

In the following, we will derive stability properties of the parameter-adaptive AMPC (5). Based on the classic result on NLP sensitivities by Fiacco (1976), there exists a constant $L$ such that

$$||\pi_{\mathrm{MPC}}(x, \theta_{\mathrm{nom}}) + \tfrac{\partial}{\partial \theta}\pi_{\mathrm{MPC}}(x, \theta)\big|_{\theta_{\mathrm{nom}}}(\theta - \theta_{\mathrm{nom}}) - \pi_{\mathrm{MPC}}(x, \theta)|| \leq L||\theta - \theta_{\mathrm{nom}}||, \tag{6}$$

if the set of active constraints does not change (Zavala and Biegler, 2009). We make this assumption:

**Assumption 2** *The set of active constraints does not change by the linear predictor. That is, the solution to (4) has the same active constraints for $\theta$ and $\theta_{\mathrm{nom}}$.*

Using (6), we can extend the stability of AMPC as per Ass. 1 to the parameter-adaptive AMPC (5):

**Theorem 1** *Let Assumptions 1 and 2 hold. If there exists an $\epsilon > 0$ such that $e_\pi(x) + \epsilon < \eta$, then the dynamical system $f_{\theta_{\mathrm{dyn}}}$ controlled by (5) is stable for all $\theta \in \tilde{\Theta}$ with $\tilde{\Theta} := \{\theta \in \Theta | (\sup_{x \in \mathbb{R}^n} ||e_\nabla(x)|| + L)||\theta - \theta_{\mathrm{nom}}|| < \epsilon\}$.*

**Proof** Because the set of active constraints does not change, we have

$$\begin{aligned}
&||\pi_{\mathrm{AMPC}}(x, \theta) - \pi_{\mathrm{MPC}}(x, \theta)|| = ||\pi_{\mathrm{NN}}(x) + \nabla_{\pi_{\mathrm{NN}}}(x)(\theta - \theta_{\mathrm{nom}}) - \pi_{\mathrm{MPC}}(x, \theta)|| \\
&= ||e_\pi(x) + \pi_{\mathrm{MPC}}(x, \theta_{\mathrm{nom}}) + \left[\tfrac{\partial}{\partial \theta}\pi_{\mathrm{MPC}}(x, \theta)\big|_{\theta_{\mathrm{nom}}} + e_\nabla(x)\right](\theta - \theta_{\mathrm{nom}}) - \pi_{\mathrm{MPC}}(x, \theta)|| \\
&\leq ||e_\pi(x)|| + ||e_\nabla(x)(\theta - \theta_{\mathrm{nom}})|| \\
&\quad + ||\pi_{\mathrm{MPC}}(x, \theta_{\mathrm{nom}}) + \tfrac{\partial}{\partial \theta}\pi_{\mathrm{MPC}}(x, \theta)\big|_{\theta_{\mathrm{nom}}}(\theta - \theta_{\mathrm{nom}}) - \pi_{\mathrm{MPC}}(x, \theta)|| \\
&\leq ||e_\pi(x)|| + ||e_\nabla(x)(\theta - \theta_{\mathrm{nom}})|| + L||\theta - \theta_{\mathrm{nom}}|| \\
&\leq \eta - \epsilon + (||e_\nabla(x)|| + L)||\theta - \theta_{\mathrm{nom}}||.
\end{aligned}$$

Hence, using Assumption 1, the theorem directly follows. ■

Theorem 1 provides a sufficient condition that assures that the AMPC for the nominal parameters $\theta_{\mathrm{nom}}$ generalizes locally to a stabilizing controller for $\theta$ in the set $\tilde{\Theta}$ using the gradient of the

nominal solution. The size of the environment depends on the accuracy of the learned MPC inputs and gradients. In general, the higher the accuracy, the larger the environment. For the gradient, this dependency follows directly from the formula of $\tilde{\Theta}$. For a more accurate $\pi_{\text{NN}}(x)$, $\epsilon$ can be chosen higher, and thus the size of $\tilde{\Theta}$ increases. In practice, global approximation error bounds could be validated, e.g., by statistical methods (Hertneck et al. (2018); Nubert et al. (2020)).

**Remark 1** *In case the set of active constraints changes, accurate sensitivities require solving a quadratic program (Kadam and Marquardt, 2004), or can be avoided by heuristics like soft constraints (Yang and Biegler, 2013). As we will empirically show in Sec. 4.3, neglecting active set changes can still work well in practice for small changes in the parameters.*

## 4. Hardware Implementation: Cartpole Pendulum

This section describes a practical implementation example of parameter-adaptive AMPC with a cartpole pendulum swing-up[1]. First, we systematically test the parameter-adaptive AMPC for parameter variations in simulation. Second, we present hardware experiments to underline our method's practical relevance. We use two different instances of a cartpole pendulum (Fig. 2), one produced by Quanser Inc. (Apkarian et al., 2012), one self-made (Mager et al., 2022). The parameter-adaptive AMPC transfers from nominal parameters to both real system instances without retraining. In both hardware systems, just using approximate controls $\pi_{\text{NN}}$ for nominal parameters fails.

The inverted pendulum is a classic benchmark control system (Boubaker, 2013). The standard cartpole model (cf. Fig. 2) has the state $x = [y, \dot{y}, \alpha, \dot{\alpha}]^\top$, which contains the pendulum angle, cart positions, and corresponding derivatives. The input $u$ is the voltage applied to the cart's motor. Neglecting the dynamics of the motor current[2], the force moving the cart is $F = C_1\dot{y} + C_2 u$, where the constants $C_1$ and $C_2$ capture all velocity-dependent friction as well as motor and transmission constants, respectively. The inertia $J = J_{\text{rod}} + J_{m_{\text{add}}}$ and the mass $m = m_{\text{rod}} + m_{\text{add}}$ of the pendulum are given about its center of mass at a distance $l$ from the point of rotation. The mass of the cart $M = M_{\text{cart}} + J_{\text{mot}}$ includes the reflected inertia of the motor and transmission. Using Euler-Lagrange's equations, we get the equations of motion

$$\begin{aligned} ml\cos(\alpha)\ddot{y} + (ml^2 + J)\ddot{\alpha} - mgl\sin(\alpha) &= -C_3\dot{\alpha} \\ (M+m)\ddot{y} + ml\cos(\alpha)\ddot{\alpha} - ml\sin(\alpha)\dot{\alpha}^2 &= F, \end{aligned} \tag{7}$$

where constant $C_3$ is the velocity-proportional friction coefficient of the pendulum rotational axis.

### 4.1. Implementation of the nonlinear MPC with sensitivities

We implement an MPC (4) with a horizon length of $N = 25$ using collocation and inputs that are step-wise constant for 160 ms. We impose constraints on the cart position $y \in [y_{\text{min}}, y_{\text{max}}]$ and the control $u \in [u_{\text{min}}, u_{\text{max}}]$ and choose a stage cost $\ell_\theta(\kappa, x(\kappa|k), u(\kappa|k)) = E_{\text{kin}} - E_{\text{pot}} + p^2 + 0.01u^2$ with potential energy $E_{\text{pot}}$ and kinetic energy $E_{\text{kin}}$ of the system, which is known to work well for pendulum swing-up tasks (Magni et al., 2002; Boubaker, 2013). We impose a terminal constraint $x(N|k) \in \mathcal{X}(N)$ to avoid set-valued MPC solutions (ambiguities) in our dataset that would cause a single feedforward NN to average through different solutions (Carius et al., 2020; Li et al.,

1. Code available at: https://github.com/hshose/Adaptive-AMPC-Cartpole
2. This is reasonable for small motor inductances in the hardware (Apkarian et al., 2012).

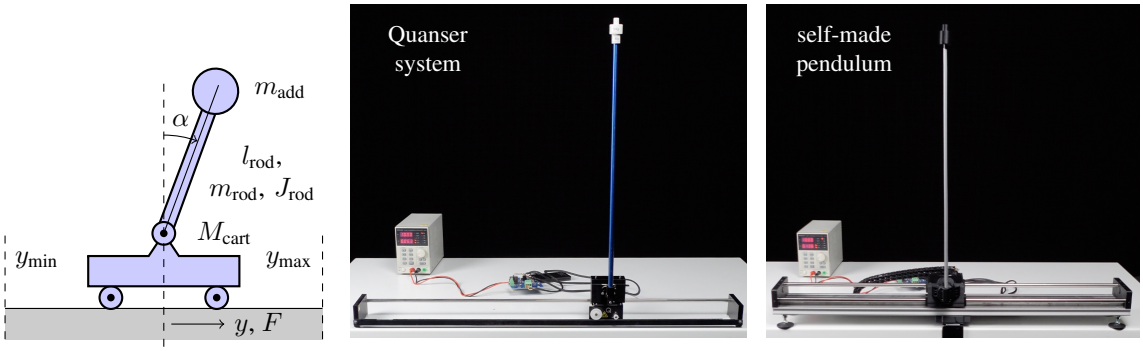


Figure 2: The cartpole inverted pendulum system (left) and the hardware pendulums used. *The Quanser pendulum (center) and the self-made pendulum (right) have significantly different parameters $\theta$ from each other and from $\theta_{\text{nom}}$. A video of our experiments is available at https://youtu.be/o1RdiYUH9uY.*

2022). To avoid ambiguities from local minima, we reinitialize the NLP solver multiple times with random states and only consider the minimal cost value from all successful runs. We compute sensitivities for the parameters $\theta = [m_{\text{add}}, M, C_1, C_2, C_3]^\top$ with nominal values $\theta_{\text{nom}} = [0.02\,\text{kg}, 0.506\,\text{kg}, -3.96\,\text{N}\,\text{s}\,\text{m}^{-1}, 1.3\,\text{N}\,\text{V}^{-1}, 0.0002\,\text{N}\,\text{m}\,\text{s}\,\text{rad}^{-1}]^\top$ from direct measurements or manufacturer datasheets. We found that the true parameters $\theta$ identified on trajectory data by minimizing a squared loss for the prediction error along an MPC horizon to have significant deviations among several instances of the system, i.e., as depicted in Fig. 2 for the Quanser system $\theta_{\text{nom}} + [-0.02\,\text{kg}, 0.36\,\text{kg}, 0.17\,\text{N}\,\text{s}\,\text{m}^{-1}, -0.82\,\text{N}\,\text{V}^{-1}, 0.002\,\text{N}\,\text{m}\,\text{s}\,\text{rad}^{-1}]^\top$ and for the self-made pendulum $\theta_{\text{nom}} + [0.01\,\text{kg}, 0.5\,\text{kg}, -1.42\,\text{N}\,\text{s}\,\text{m}^{-1}, 0.5\,\text{N}\,\text{V}^{-1}, 0.01\,\text{N}\,\text{m}\,\text{s}\,\text{rad}^{-1}]^\top$.

The MPC optimization problem is formulated using Pyomo (Bynum et al., 2021) and solved using IPOPT (Wächter and Biegler, 2006) with sensitivities through the sIPOPT extension (Pirnay et al., 2011). The dataset used for the training contains optimal inputs for 368 000 randomly sampled initial states with 1% of initial states densely sampled around the upright position. Computation of the dataset takes over 12 000 CPU core hours[3] due to random reinitializations and high numerical accuracy for sensitivities. This shows that without the proposed method, tuning a nominal AMPC by recomputing the dataset would require large amounts of resources and time.

### 4.2. Implementation of the parameter-adaptive AMPC

We use fully-connected feedforward NNs with 50 neurons per layer, tangent-hyperbolic activations, and 5 and 8 layers for $\pi_{\text{NN}}(\cdot)$ and $\nabla_{\pi_{\text{NN}}}(\cdot)$, respectively. While ReLU activations are well suited to represent piecewise affine solutions of linear MPC, tangent-hyperbolic activations yield smooth policies for nonlinear systems which is favored in practical applications Carius et al. (2020); Nubert et al. (2020). Training both NNs in Jax (Bradbury et al., 2018) with a mixture of Minkowski and linear loss takes 4 hours[4]. For hardware experiments, the parameter-adaptive AMPC is implemented on a STM32G474 MCU[5]. We use the embedded C++ library builder *modm* for hardware abstraction. The NN forward pass is implemented as single-precision floating point matrix-vector multiplication

3. computed in parallel with Intel Xeon Platinum 8160 "SkyLake" CPU at 2.1 GHz
4. on an Nvidia RTX4070
5. Arm Cortex-M4 core at 170MHz with 96kByte of SRAM and 512kByte flash

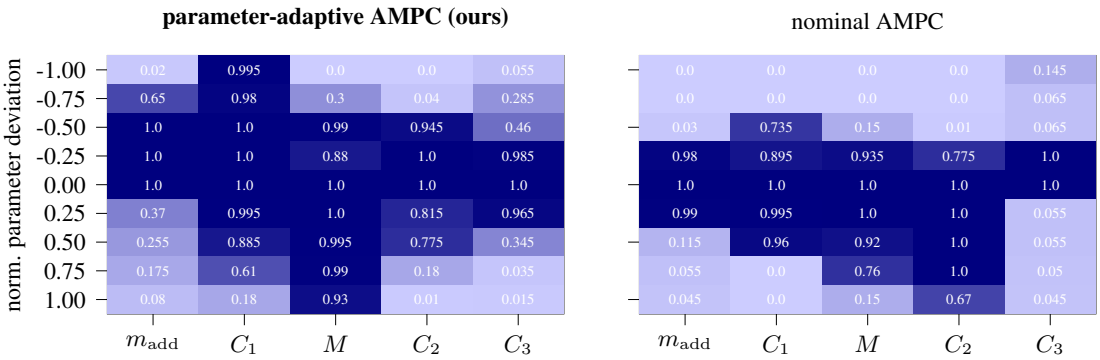


Figure 3: Approximate linear predictor effectiveness: *Each cell indicates the fraction of closed-loop simulations from random initial states where the pendulum is stabilized upright, given a deviation in parameters (scaled to* $\pm 1$ *in this plot). The use of approximate linear predictors enables the NN to stabilize the system across a broader parameter range.*

with a lookup table for the tangent hyperbolic and takes less than 2ms to evaluate for both NNs. Due to the symmetry of the problem, we can reset $\alpha \leftarrow \text{mod}(\alpha+\pi, 2\pi)-\pi$ before evaluating the AMPC in practical experiments. Hyperparameters in this example were chosen for simplicity and based on engineering intuition, thus further tuning, quantization, or pruning could improve performance.

### 4.3. Simulation results: study of parameter variations

We test parameter-adaptive AMPC in simulation by systematically changing each parameter individually within the bounds $\theta - \theta_{\text{nom}} \in \pm[0.04\,\text{kg}, 1\,\text{kg}, 9\,\text{N}\,\text{s}\,\text{m}^{-1}, 1\,\text{N}\,\text{V}^{-1}, 0.06\,\text{N}\,\text{m}\,\text{s}\,\text{rad}^{-1}]^\top$ while keeping all other parameters at nominal values. We evaluate for the same random initial conditions if both AMPCs successfully perform the swing-up and stabilization within constraints. As shown in Fig. 3, the parameter-adaptive AMPC (our method) stabilizes the system across a broader range of parameter values compared to the nominal controller. Interestingly, the nominal controller is able to stabilize the system for a broader range of values in parameter $C_2$, which we attribute to the linear predictions becoming inaccurate. However, because one is free to choose the parameters in the AMPC, if the real pendulum dynamics lie in this range, one can just choose to use the nominal parameters to control the pendulum.

### 4.4. Hardware results: transfer to system instances without retraining

We test parameter-adaptive AMPC on two real cartpole systems (see Fig. 2). A video of our experiments can be found at https://youtu.be/o1RdiYUH9uY. The controller with nominal parameters does not perform a swing-up and reaches a limit cycle on both the Quanser and self-made systems. When using parameters identified from real trajectory data for each system instance (see Sec. 4.1), the parameter-adaptive AMPC (our method) reliably performs a successful swing-up, as depicted for an exemplary trajectory in Fig. 4. In addition, the corrected parameter values for the Quanser pendulum fail on the self-made pendulum where the pendulum starts rotating indefinitely (depicted in orange in Fig. 4, right) or the cart violates its constraints. There is a notable residual offset of the cart position in our experiments, which is due to an unmodeled friction effect in the pendulum bearing. However, as can be seen in our video, the cart does not violate its constraints over a long horizon, but slowly drives back and forth on its rail.

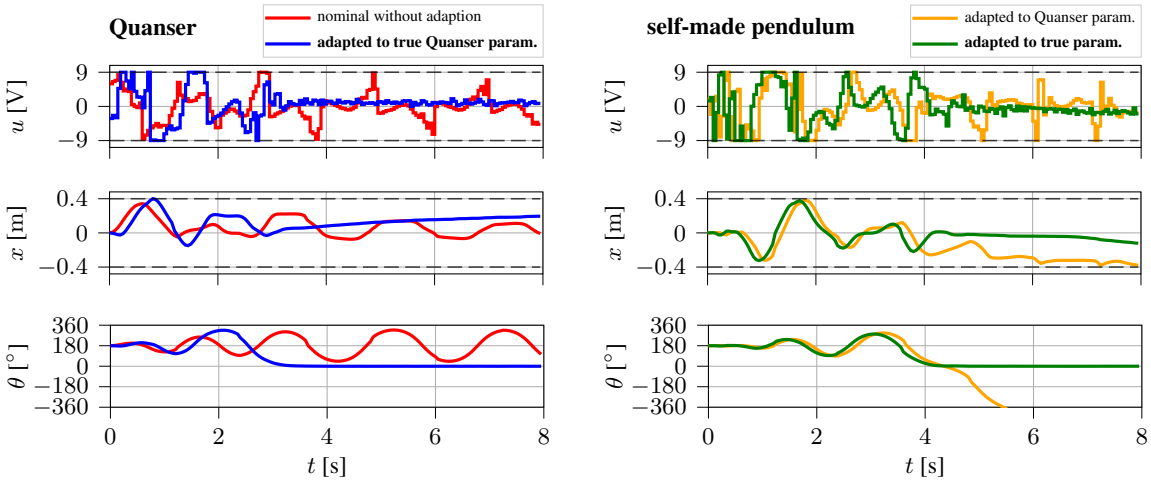


Figure 4: Closed-loop evaluation on Quanser and self-made cartpole pendulum hardware: *The parameter-adaptive AMPC with true identified system parameters stabilizes both pendulums, Quanser (blue) and self-made (green), while no adaption (red) or parameters from the wrong system instance (orange) fail during the swing-up. Nominal, Quanser, and self-made system have significantly different parameters. The slow drift in the cart position for both systems is due to unmodeled stick-slip effects in the pendulum bearing near zero velocity. However, as our video shows on a long horizon, the cart slowly oscillates on its rail. Notably, with parameter-adaptive AMPC, the systems satisfy constraints.*

## 5. Conclusion

In this work, we presented an approach that allows for tuning parameters of AMPC online without retraining. Alongside the output of an MPC, parameter-adaptive AMPC also learns the gradient of this output with respect to MPC parameters offline. Using a linear predictor, we can then adapt the output of the AMPC to parameter changes online. Our work resulted in the, to the best of our knowledge, first real-world implementation of AMPC on a resource-constrained MCU controlling a fast nonlinear system. Our simulation and hardware experiments demonstrate the efficacy of parameter-adaptive AMPC. Throughout experiments, we noticed that tuning is very intuitive due to the physical meaning of dynamics parameters. Further, identified parameters from real data work well in practice. As a result, the tuning process for the AMPC took only minutes from first try to a successful swing-up. Without parameter adaption, this would have required weeks for recomputing datasets. As such, the method is a significant step towards real-world applications of AMPC.

We identified two significant implementation challenges. First, despite multiple reinitializations of the NLP solver to find a global optimum, there remain outliers in the dataset due to local minima. Second, accurate sensitivities require additional fine-tuning of solver settings and high numerical accuracy, thus prolonging dataset generation in practice. Hence, future work should face these challenges by developing methods that cope with the non-uniqueness of MPC solutions and improve sensitivity calculation. As the results suggest, parameter-adaptive AMPC enables practical applications for systems where classic AMPC would be too cumbersome to tune, e.g., in robotics. We plan to investigate such new applications and the generalization of our method to high dimensional systems in future research.

## Acknowledgments

This work is funded in part by the German Research Foundation (DFG) – RTG 2236/2 (UnRAVeL) and the DFG priority program 1914 (grant TR 1433/1-2). Simulations were performed with computing resources granted by RWTH Aachen University under project rwth1500. We thank Andrés Posada Moreno and Christian Fiedler for helpful discussions.

## References

Mohammad Abu-Ali, Felix Berkel, Maximilian Manderla, Sven Reimann, Ralph Kennel, and Mohamed Abdelrahem. Deep learning-based long-horizon MPC: robust, high performing, and computationally efficient control for PMSM drives. *IEEE Transactions on Power Electronics*, 2022.

Saket Adhau, Sayli Patil, Deepak Ingole, and Dayaram Sonawane. Embedded implementation of deep learning-based linear model predictive control. In *Sixth Indian control conference (ICC)*, 2019.

Bernt M Åkesson and Hannu T Toivonen. A neural network model predictive controller. *Journal of Process Control*, 2006.

Alessandro Alessio and Alberto Bemporad. A survey on explicit model predictive control. *Nonlinear Model Predictive Control: towards New Challenging Applications*, 2009.

Joel AE Andersson and James B Rawlings. Sensitivity analysis for nonlinear programming in CasADi. In *IFAC-PapersOnLine, 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018*, 2018.

Jacob Apkarian, Herve Lacheray, and Peter Martin. Student workbook IP02 base unit experiment for matlab/simulink users. *Quanser Inc.*, 2012.

Florian A Bayer, Florian D Brunner, Mircea Lazar, Marc Wijnand, and Frank Allgöwer. A tube-based approach to nonlinear explicit MPC. In *55th Conference on Decision and Control (CDC)*. IEEE, 2016.

Alberto Bemporad and Carlo Filippi. An algorithm for approximate multiparametric convex programming. *Computational optimization and applications*, 2006.

Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 2002.

Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization. *Frontiers in Robotics and AI*, 2022.

Olfa Boubaker. The inverted pendulum benchmark in nonlinear control theory: a survey. *International Journal of Advanced Robotic Systems*, 2013.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Christof Büskens and Helmut Maurer. Sensitivity analysis and real-time optimization of parametric nonlinear programming problems. In *Online Optimization of Large Scale Systems*. 2001.

Michael L. Bynum, Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. *Pyomo–optimization modeling in Python*. Springer Science & Business Media, 2021.

Jan Carius, Farbod Farshidian, and Marco Hutter. MPC-Net: A first principles guided policy search. *IEEE Robotics and Automation Letters*, 2020.

Kimberly J Chan, Joel A Paulson, and Ali Mesbah. Deep learning-based approximate nonlinear model predictive control with offset-free tracking for embedded applications. *American Control Conference (ACC)*, 2021.

Steven W Chen, Tianyu Wang, Nikolay Atanasov, Vijay Kumar, and Manfred Morari. Large scale model predictive control with neural networks and primal active sets. *Automatica*, 2022.

Moritz Diehl, H Georg Bock, Johannes P Schlöder, Rolf Findeisen, Zoltan Nagy, and Frank Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 2002.

Anthony V Fiacco. Sensitivity analysis for nonlinear programming using penalty methods. *Mathematical programming*, 1976.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Proceedings of the 5th Conference on Robot Learning*, 2022.

Marco Forgione, Dario Piga, and Alberto Bemporad. Efficient calibration of embedded mpc. *IFAC-PapersOnLine, 21st IFAC World Congress*, 53(2):5189–5194, 2020.

Camilo Gonzalez, Houshyar Asadi, Lars Kooijman, and Chee Peng Lim. Neural networks for fast optimisation in model predictive control: a review. *arXiv preprint arXiv:2309.02668*, 2023.

Michael Hertneck, Johannes Köhler, Sebastian Trimpe, and Frank Allgöwer. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2018.

Boris Houska and Mario E Villanueva. Robust optimization for MPC. *Handbook of model predictive control*, 2019.

Johannes Jäschke, Xue Yang, and Lorenz T Biegler. Fast economic model predictive control based on NLP-sensitivities. *Journal of Process Control*, 2014.

Tor A Johansen. Approximate explicit receding horizon control of constrained nonlinear systems. *Automatica*, 2004.

Jitendra V Kadam and Wolfgang Marquardt. Sensitivity-based solution updates in closed-loop dynamic optimization. In *IFAC Proceedings Volumes, 7th IFAC Symposium on Dynamics and Control of Process Systems 2004*, 2004.

Benjamin Karg and Sergio Lucia. Learning-based approximation of robust nonlinear predictive control with state estimation applied to a towing kite. In *18th European Control Conference (ECC)*, 2019.

Martin Klaučo, Martin Kalúz, and Michal Kvasnica. Machine learning-based warm starting of active set methods in embedded model predictive control. *Engineering Applications of Artificial Intelligence*, 2019.

Johannes Köhler, Raffaele Soloperto, Matthias A Müller, and Frank Allgöwer. A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, 2020.

Dinesh Krishnamoorthy. A sensitivity-based data augmentation framework for model predictive control policy approximation. *IEEE Transactions on Automatic Control*, 2021.

Dinesh Krishnamoorthy. An improved data augmentation scheme for model predictive control policy approximation. *IEEE Control Systems Letters*, 2023.

Michal Kvasnica, Juraj Holaza, Bálint Takács, and Deepak Ingole. Design and verification of low-complexity explicit MPC controllers in MPT3. In *European Control Conference (ECC)*, 2015.

Sebastian Leonow, Raphael Dyrska, and Martin Mönnigmann. Embedded implementation of a neural network emulating nonlinear MPC in a process control application. 2023.

Yun Li, Kaixun Hua, and Yankai Cao. Using stochastic programming to train neural network approximation of nonlinear MPC laws. *Automatica*, 2022.

Sergio Lucia, Denis Navarro, Benjamin Karg, Hector Sarnago, and Oscar Lucia. Deep learning-based model predictive control for resonant power converters. *IEEE Transactions on Industrial Informatics*, 2020.

Lukas Lüken, Dean Brandner, and Sergio Lucia. Sobolev training for data-dfficient approximate nonlinear MPC. In *IFAC Proceedings Volumes, 22nd IFAC World Congress*, 2023.

Fabian Mager, Dominik Baumann, Carsten Herrmann, Sebastian Trimpe, and Marco Zimmerling. Scaling beyond bandwidth limitations: wireless control with stability guarantees under overload. *ACM Transactions on Cyber-Physical Systems (TCPS)*, 2022.

Lalo Magni, Riccardo Scattolini, and Karl Johan Åström. Global stabilization of the inverted pendulum using model predictive control. In *IFAC Proceedings Volumes, 15th IFAC World Congress*, 2002.

Julian Nubert, Johannes Köhler, Vincent Berenz, Frank Allgöwer, and Sebastian Trimpe. Safe and fast tracking on a robot manipulator: robust MPC and neural network control. *IEEE Robotics and Automation Letters*, 2020.

Thomas Parisini, M Sanguineti, and R Zoppoli. Nonlinear stabilization by receding-horizon neural regulators. *International Journal of Control*, 1998.

Joel A Paulson, Farshud Sorourifar, and Ali Mesbah. A tutorial on derivative-free policy learning methods for interpretable controller representations. In *2023 American Control Conference (ACC)*, pages 1295–1306. IEEE, 2023.

Hans Pirnay, Rodrigo López-Negrete, and Lorenz T Biegler. sIPOPT reference manual. Carnegie Mellon University, 2011.

James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017.

Yannic Vaupel, Nils C Hamacher, Adrian Caspari, Adel Mhamdi, Ioannis G Kevrekidis, and Alexander Mitsos. Accelerating nonlinear model predictive control through machine learning. *Journal of process control*, 2020.

Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 2022.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 2006.

Daming Wang, Zheng John Shen, Xin Yin, Sai Tang, Xifei Liu, Chao Zhang, Jun Wang, Jose Rodriguez, and Margarita Norambuena. Model predictive control using artificial neural network for power converters. *IEEE Transactions on Industrial Electronics*, 2021.

Yangxiao Xiang, Henry Shu-Hung Chung, and Hongjian Lin. On the use of dualrelu ann for approximating explicit model predictive control for buck converters. In *2024 IEEE Applied Power Electronics Conference and Exposition (APEC)*, pages 2822–2827. IEEE, 2024.

Xue Yang and Lorenz T Biegler. Advanced-multi-step nonlinear model predictive control. *Journal of process control*, 2013.

Victor M Zavala and Lorenz T Biegler. The advanced-step NMPC controller: optimality, stability and robustness. *Automatica*, 2009.