Received 9 June 2022; revised 2 September 2022 and 27 October 2022; accepted 30 October 2022. Date of publication 10 November 2022; date of current version 2 December 2022. Recommended by Senior Editor Minghui Zhu.

Digital Object Identifier 10.1109/OJCSYS.2022.3221063

Convex Neural Network-Based Cost Modifications for Learning Model Predictive Control

KATRINE SEEL [1,2], ARASH BAHARI KORDABAD [1], SÉBASTIEN GROS [1], AND JAN TOMMY GRAVDAHL [1] (Senior Member, IEEE)

[1]Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7032 Trondheim, Norway
[2]SINTEF Digital, Mathematics and Cybernetics, 0373 Oslo, Norway

CORRESPONDING AUTHOR: KATRINE SEEL (e-mail: katrine.seel@ntnu.no).

This work was supported by the Research Council of Norway.

ABSTRACT Developing model predictive control (MPC) schemes can be challenging for systems where an accurate model is not available, or too costly to develop. With the increasing availability of data and tools to treat them, learning-based MPC has of late attracted wide attention. It has recently been shown that adapting not only the MPC model, but also its cost function is conducive to achieving optimal closed-loop performance when an accurate model cannot be provided. In the learning context, this modification can be performed via parametrizing the MPC cost and adjusting the parameters via, e.g., reinforcement learning (RL). In this framework, simple cost parametrizations can be effective, but the underlying theory suggests that rich parametrizations in principle can be useful. In this paper, we propose such a cost parametrization using a class of neural networks (NNs) that preserves convexity. This choice avoids creating difficulties when solving the MPC problem via sensitivity-based solvers. In addition, this choice of cost parametrization ensures nominal stability of the resulting MPC scheme. Moreover, we detail how this choice can be applied to economic MPC problems where the cost function is generic and therefore does not necessarily fulfill any specific property.

INDEX TERMS Dissipativity, economic nonlinear model predictive control, neural networks, reinforcement learning.

I. INTRODUCTION

Learning-based model predictive control (MPC) has become a popular field of research. In general terms, this category describes the combination of recent advances within the field of machine learning and MPC. An important motivation for learning-based MPC is the potential of designing optimal controllers even in face of model errors and uncertainties.

One direction of learning-based MPC is the use of learning to build the MPC prediction model. In that context, neural networks (NNs) have typically been used for learning an approximation of the system dynamics from data, which is then used as the prediction model in the MPC scheme, see e.g. [1], [2], [3]. However, it is in general difficult to conclude regarding the closed-loop optimality of the resulting MPC scheme.

Another approach to learning-based MPC, is using cost modifications to handle model imperfection. In [4], it was proved that an MPC scheme can deliver the optimal policy for a system, even if the model in the MPC is inaccurate. This can be achieved under some fairly mild conditions via modifications of the cost of the MPC scheme, which compensates for model inaccuracy. This idea can be applied in practice by parametrizing the MPC cost and constraints, and using learning techniques to adapt the parameters. In that context, reinforcement learning (RL) has been extensively investigated as a learning tool for tuning the cost, constraints and MPC model, see [5], [6], [7], [8], [9].

RL aims to find the optimal policy that minimizes the expected value of the infinite-horizon sum of costs [10]. Existing RL methods are usually divided into two categories,

that is either value-based or policy-based methods. Value-based methods aim at fitting a parameterized approximation of the optimal action-value function, whereas policy-gradient methods parameterize and learn the optimal policy directly. Data-driven techniques are then used to find the optimal function parameters. For both types of methods, NNs are often used as they are generic function approximators. Recently, MPC schemes have also been exploited as function approximators for RL. This approach allows one to use the extensive theory underlying MPC to discuss the closed-loop properties of the resulting policy [11], [12], [13]. We take advantage of the following result, and will use both value-based and policy-based methods.

The idea of compensating model inaccuracy with cost modifications, has successfully been tested in [4], [14], [15], using fairly simple parametrizations of the cost. However, the theory underlying this result suggests that in principle the cost parametrization should be "rich," i.e. it should be able to capture fairly generic functions. Rich parametrizations of the cost in the context of economic nonlinear MPC (ENMPC) were first considered in [16]. In this paper, we elaborate on this early investigation and propose a more complete framework to provide such a rich parametrization. More specifically, we propose to use a class of NNs that preserve convexity. This choice has two important benefits. First, ensuring convexity of the MPC cost alleviates the difficulties inherent to solving MPC schemes numerically using sensitivity-based solvers. Second, the stage cost in the MPC scheme must be lower-bounded by a $\mathcal{K}_\infty$-function to ensure stability. A convex function can be designed to satisfy this lower bound, and in turn ensures nominal stability of the resulting MPC scheme.

In this paper, we will apply the framework of convex cost modifications to ENMPC problems. ENMPC is concerned with optimizing performance rather than penalizing deviations from a given reference. This means that the cost function not necessarily can be lower-bounded by a $\mathcal{K}_\infty$-function [17]. Dissipativity theory is a fundamental tool in order to understand the stability of ENMPC. For dissipative problems, there exist corresponding tracking MPC schemes that yield the same policy as the ENMPC scheme. As tracking MPC schemes use quadratic cost functions, they intrinsically satisfy the lower bound. Dissipativity of a problem is verified through the existence of a storage function that satisfies the dissipativity inequality [18]. Finding a valid storage function for the general problem is hard, but may be captured using RL techniques as suggested in [4] and justified in [19]. As we are focusing on economic problems, we use deterministic systems as a proof of concept, for which general dissipativity theory is valid.

Even for dissipative problems, the ideal modified stage cost may not be convex. This means that enforcing convexity as a means to ensure stability, may impose limitations on the learned cost function. In [20], the authors showed that for a dissipative problem, a tracking MPC with a quadratic stage cost is locally equivalent to ENMPC. In other words, a convex cost approximation is at least valid locally.

The main contribution of the paper is the introduction of convex NNs as cost modifications in MPC schemes with imperfect prediction models. Using the MPC scheme as a function approximator for the value function and the policy, we will use RL to adjust the cost parameters, including the NN weights, in pursuance of the optimal economic policy. The second contribution of the paper is the combination of RL methods for when neither value-based nor policy-based RL methods alone are sufficient. We let one simulation example serve as a proof of concept, and then consider a second simulation example to benchmark the addition of convex cost modifications against the standard quadratic cost parametrization.

The paper is structured as follows. Section II introduces the problem statement and presents general theory on dissipativity leading to necessary assumptions for stability. This is followed by a section on convex cost parametrizations. In Section IV the requirements from Section III are specified in terms of NNs. Section V then outlines how RL can be used for updating the parameters in the parameterized MPC scheme. A method that combines value-based and policy-based RL methods is detailed in Section VI. To illustrate the presented theory, two numerical examples are presented in Section VII. Finally, conclusions are given in Section VIII.

II. BACKGROUND AND PROBLEM STATEMENT

We consider discrete-time, constrained dynamic systems of the form:

$$s_{k+1} = f(s_k, a_k), \quad h(s_k, a_k) \leq 0, \tag{1}$$

where k denotes the discrete time step, $s_k \in \mathbb{X} \subseteq \mathbb{R}^n$ denotes the state and $a_k \in \mathbb{U} \subseteq \mathbb{R}^m$ denotes the input. The (possibly nonlinear) dynamics are defined by $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. The function $h(s_k, a_k)$ describes a mixed input-state constraint. We propose how to formulate stable MPC schemes by parameterizing the cost function, for an economic problem where the stage cost $L : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ is indefinite, using a (potentially) inaccurate model of the system, $\tilde{f}$.

A. ECONOMIC NMPC

ENMPC is a framework for optimal control of dynamical systems with respect to a generic economic objective, typically performance-oriented, and usually referred to as economic. The objective may address the energy, time or financial cost of running a system. The following section will describe the ENMPC formulation and recall dissipativity theory as a tool to analyze stability. The following standard assumptions for ENMPC are used in the rest of the paper. The set of states $s \in \mathbb{X}$ and inputs $a \in \mathbb{U}$ define the set of feasible state-input pairs as follows

$$\mathbb{Z} = \{(s, a) \in \mathbb{X} \times \mathbb{U} \mid f(s, a) \in \mathbb{X},\ h(s, a) \leq 0\}, \tag{2}$$

for which we need the following assumption.

Assumption 1 (Properties of constraint sets): The set $\mathbb{Z}$ is compact and non-empty.

Assumption 2 (Continuity of cost and system): The functions $f(\cdot)$ and $L(\cdot)$ are continuous on $\mathbb{Z}$.

The optimal steady state pair (s_e, a_e) is defined as follows:

$$(s_e, a_e) \in \arg\min_{(s,a)\in\mathbb{Z}} \{L(s,a) \mid s = f(s,a)\}. \tag{3}$$

We define a *shifted stage cost* as:

$$\ell(s,a) = L(s,a) - L(s_e, a_e), \tag{4}$$

so that $\ell(s_e, a_e) = 0$.

Let π denote a deterministic policy, that maps the state to an action, $\pi : \mathbb{R}^n \to \mathbb{R}^m$. The optimal policy $\pi^\star$ is then given by the solution to the following infinite-horizon problem

$$V^\star(s) = \min_{\pi} \quad \sum_{k=0}^{\infty} \ell(x_k, \pi(x_k)) \tag{5a}$$

$$\text{s.t} \quad x_{k+1} = f(x_k, \pi(x_k)), \quad x_0 = s, \tag{5b}$$

$$h(x_k, \pi(x_k)) \leq 0, \tag{5c}$$

where x_k denotes the predicted state, not to be confused with the true state s_k, so that $(x_k)_{k=1}^{\infty}$ is the predicted state trajectory for the system starting at some initial state $x_0 = s$, subject to policy π. The optimal action-value function is defined as follows

$$Q^\star(s,a) = \ell(s,a) + V^\star(f(s,a)). \tag{6}$$

The action-value function and the value function are related through the underlying Bellman equations [21]

$$\pi^\star(s) \in \arg\min_a Q^\star(s,a), \; V^\star(s) = \min_a Q^\star(s,a). \tag{7}$$

B. STRICT DISSIPATIVITY

A generic economic stage cost can make it challenging to establish closed-loop stability of the resulting MPC scheme. Satisfaction of the dissipativity conditions, entails that the ENMPC scheme can be recast as a tracking MPC scheme, for which closed-loop stability properties are straightforward to prove [18]. The concept of strict dissipativity is defined using a storage function λ. A function $c : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a $\mathcal{K}$-function if it is strictly increasing and $c(0) = 0$. If a $\mathcal{K}$-function is such that $\lim_{b\to\infty} c(b) = \infty$, it is a $\mathcal{K}_\infty$-function.

Definition 1 (Strict dissipativity): The system (1) with stage cost $L(\cdot)$ is strictly dissipative if there exists a storage fuction $\lambda : \mathbb{X} \to \mathbb{R}$ satisfying

$$\lambda(f(s,a)) - \lambda(s) \leq -\rho(||s - s_e||) + \ell(s,a), \tag{8}$$

where $\rho \in \mathcal{K}_\infty$ and $||\cdot||$ denotes the Euclidean norm.

Assumption 3 (Strict dissipativity): The system (1) is strictly dissipative.

For the storage function, we make the following assumption.

Assumption 4 (Continuity of storage function): The storage function $\lambda(\cdot)$ is continuous on $\mathbb{Z}$.

Without loss of generality, we can add a constant to the storage function, in order to ensure that $\lambda(s_e) = 0$, without invalidating inequality (8). If λ exists, we can define the rotated stage cost as

$$\bar{\ell}(s,a) = \ell(s,a) + \lambda(s) - \lambda(f(s,a)). \tag{9}$$

Combining (8) and (9) then yields

$$\rho(||s - s_e||) \leq \bar{\ell}(s,a), \quad \bar{\ell}(s_e, a_e) = 0. \tag{10}$$

For a strictly dissipative problem, the ENMPC scheme is equal to a tracking MPC, using the rotated stage cost $\bar{\ell}$. As the rotated stage cost is zero at the optimal steady state and lower-bounded by a $\mathcal{K}_\infty$-function, the closed-loop system is stable [17]. The corresponding tracking MPC is formulated as

$$V^\star(s) = \min_{\pi} \; -\lambda(s) + \sum_{k=0}^{\infty} \bar{\ell}(x_k, \pi(x_k)) \tag{11a}$$

$$\text{s.t} \quad (5b), (5c). \tag{11b}$$

To formulate the finite-horizon MPC, we introduce the finite-horizon stage cost, $\hat{\ell}$, and add a terminal cost, according to

$$V^\star(s) = \min_{u,x} \; -\lambda(s) + T(x_N) + \sum_{k=0}^{N-1} \hat{\ell}(x_k, u_k) \tag{12a}$$

$$\text{s.t} \quad x_{k+1} = f(x_k, u_k), \quad x_0 = s, \tag{12b}$$

$$h(x_k, u_k) \leq 0, \tag{12c}$$

$$x_N \in \mathbb{X}_f, \tag{12d}$$

where N is the horizon length, $T : \mathbb{X}_f \to \mathbb{R}$ is a penalty on the terminal state and $\mathbb{X}_f$ is a compact terminal region containing the steady state operating point in its interior. The resulting input sequence is the vector $u = \{u_0, \ldots, u_{N-1}\}$, and $x = \{x_0, \ldots, x_N\}$ is the corresponding state trajectory. Note that we use $\hat{\ell}$ to denote the finite-horizon stage cost, to clearly distinguish it from the infinite-horizon stage cost $\bar{\ell}$, as these may not be the same. The stage cost $\hat{\ell}$ must be selected such that it satisfies (10). For the terminal cost, we make the following assumptions.

Assumption 5 (Continuity of terminal cost): The terminal cost $T(\cdot)$ is continuous on $\mathbb{X}_f$.

Assumption 6 (Stability assumption): There exists a compact terminal region $\mathbb{X}_f \subseteq \mathbb{X}$, containing the point s_e in its interior, and terminal control law $\kappa_f : \mathbb{X}_f \to \mathbb{U}$ such that

$$T(f(s, \kappa_f(s))) - T(s) \leq -\hat{\ell}(s, \kappa_f(s)), \tag{13}$$

$\forall s \in \mathbb{X}_f$ and $(s, \kappa_f(s)) \in \mathbb{Z}$. Moreover, $T(s_e) = 0$ and $T(s) > 0 \; \forall s \in \mathbb{X}_f \setminus \{s_e\}$.

Remark 1: This assumption requires that for each $s \in \mathbb{X}_f$, $f(s_k, \kappa_f(s)) \in \mathbb{X}_f$, i.e. the set $\mathbb{X}_f$ is a control invariant set.

Assumption 6 is a standard assumption used with the purpose of analyzing the stability of the resulting closed-loop system.

Theorem II.1: Let Assumptions 1–6 hold. Then the steady state solution s_e is an asymptotically stable equilibrium point of the system (1) using input a where $a = u_0^*$ and u_0^* is the first element in the optimal solution to (12).

Proof: Note that the term $-\lambda(s)$ does not affect the optimal solution in (12), but shapes the action-value and value function. The rest of the proof is a standard result, and can be found in e.g., [17].

C. PARAMETERIZED TRACKING MPC

We propose to use a finite-horizon MPC problem to approximate the value function (5), where the cost function is parameterized with parameters θ, according to

$$V_\theta(s) = \min_{x,u} \; -\lambda_\theta(s) + T_\theta(x_N) + \sum_{k=0}^{N-1} \hat{\ell}_\theta(x_k, u_k) \tag{14a}$$

$$\text{s.t } x_{k+1} = \tilde{f}(x_k, u_k), \quad x_0 = s, \tag{14b}$$

$$h(x_k, u_k) \leq 0, \tag{14c}$$

$$x_N \in \mathbb{X}_f, \tag{14d}$$

where $\tilde{f}(x_k, u_k)$ is a potentially inaccurate prediction model. The resulting policy is given by the first element in the input sequence

$$\pi_\theta(s) = \bar{u}_0^*(s, \theta), \tag{15}$$

where $\bar{u}^*(s, \theta)$ is the optimal solution to (14). Using the MPC scheme as a function approximator, entails using RL to update the parameters θ in order to shape the value function estimate (14) and improve the policy (15). For stable economic problems, using rich parametrizations for the cost function enables the MPC scheme to deliver the optimal policy even with an inaccurate prediction model. This is stated in Theorem 1 in [4]. The following assumption is needed to ensure the stability of the closed-loop system.

Assumption 7: The stage cost $\hat{\ell}_\theta(s, a)$ satisfies

$$\rho(||s - s_e||) \leq \hat{\ell}_\theta(s, a), \quad \hat{\ell}_\theta(s_e, a_e) = 0, \tag{16}$$

where the optimal steady state pair (s_e, a_e) may be part of the parametrization θ.

Because the MPC scheme in (14) may use an inaccurate prediction model, we propose to parameterize the steady state. This is not a new idea, but closely resembles an approach the real-time optimization (RTO) community refers to as modifier adaptation, see e.g. [22]. Our case differs in that we use RL to adjust the modifiers, or as we call them, parameters.

We note that as long as the MPC scheme is using an inaccurate prediction model $\tilde{f}(s, a)$, we can only guarantee nominal stability of the resulting MPC scheme, i.e. stability with respect to the MPC model. In order to guarantee the stability of the true system, we would have to apply robust techniques. Robust techniques for MPC with RL, is treated in [8]. The authors in [8] describe a robust technique for MPC that uses a nominal prediction model. A tube-based approach considers the system stochasticity and the model uncertainties, and is used to perform a suitable tightening of the constraints. Because we are considering economic MPC problems, re-cast as tracking MPC schemes, the techniques from [8] directly extend to the proposed parameterized MPC scheme. For the sake of brevity, we have not treated robust techniques further in this paper.

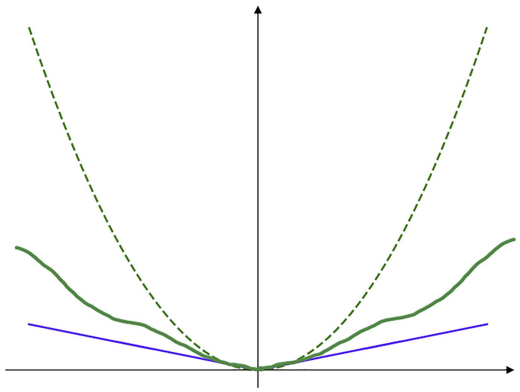


FIGURE 1. Generic cost function (green) lower-bounded by a $\mathcal{K}_\infty$-function (blue), with a quadratic approximation (dashed green).

Remark 2: The easiest way to ensure (nominal) stability of (14) is to use the so-called zero terminal equality i.e. $\mathbb{X}_f = \{s_e\}$, for which the terminal cost can be omitted. However, this is very restrictive and may yield feasibility issues. The terminal constraint may also be defined using an inequality constraint defined by a terminal region. For the standard quadratic stage and terminal cost, the terminal region can be approximated. For more generic stage and terminal cost approximations, the terminal region may be hard to find. However, in practice we may use a general positive definite terminal cost, select N "large enough," and ensure stability without terminal constraints.

Proposition 1: The parameterized MPC scheme in (14), with a stage cost satisfying Assumption 7, using either a stabilizing terminal cost, i.e. satisfying Assumption 6, with terminal constraints or a general positive definite terminal cost and a long enough horizon N, will be stabilizing for all θ.

Proof: This is a standard result, and proofs are given in e.g. [17] and [23].

III. CONVEX COST PARAMETRIZATIONS

For proving nominal closed-loop stability of the MPC scheme, the stage cost must be lower-bounded by a $\mathcal{K}_\infty$-function. A generic cost function lower-bounded by a $\mathcal{K}_\infty$-function is illustrated in Fig. 1. The $\mathcal{K}_\infty$ lower bound in principle entails no restrictions regarding the convexity of the cost function. On the other hand, convexity may be selected as a tool to show that the same lower bound holds. The first reason for selecting convexity is that we are able to describe and therefore parameterize generic convex functions. Second, it is easier to handle convex functions in the MPC scheme. Moreover, a strictly convex function will satisfy the $\mathcal{K}_\infty$ lower bound by enforcing its global minimum to be zero at zero and the function to be radially unbounded.

For ENMPC schemes that are locally stabilizing, it can be shown that the modified stage cost obtained using dissipativity theory, is locally quadratic [20]. The quadratic approximation

of the stage cost is also illustrated in Fig. 1. In fact, the local quadratic approximation of the cost can be computed by solving a semi-definite program (SDP) [20]. As convex functions also describe quadratic functions, we can establish that the choice of a convex stage cost is at least valid locally.

A. STAGE COST

In order to satisfy strict dissipativity as stated in Assumption 3, the approximated stage cost must satisfy the lower bound from Assumption 7 and continuity from Assumption 2. Without loss of generality, we let $(s_a, a_e) = (0, 0)$ so that $\bar{\ell}(0, 0) = 0$. The following Lemma lists the function requirements for the stage cost.

Lemma III.1: Let $p(s, a) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be a strictly convex function. If the minimum of the function is $p(0, 0) = 0$, and $p(s, a)$ is radially unbounded with respect to s, i.e. $p(s, a) \to \infty$ as $s \to \infty$, then $\rho(\|s\|) \leq p(s, a)$ for some $\rho \in \mathcal{K}_\infty$.

Proof: If the function is strictly convex, it will only have one global minimum. If this is at $p(0, 0) = 0$, it means that $\forall (s, a) \neq (0, 0)\ p(s, a) > 0$, so that $p(s, a) \geq \beta(||s||)$, where $\beta \in \mathcal{K}$. Because $p(s, a)$ is radially unbounded with respect to s, we can also state that $p(s, a) \geq \rho(||s||)$, where $\rho \in \mathcal{K}_\infty$.

B. TERMINAL COST

The terminal cost must satisfy continuity from Assumption 5, and should be positive definite according to Assumption 6.

C. STORAGE FUNCTION

The storage function must satisfy continuity from Assumption 4, and zero at the optimal state, $\lambda(s_e) = 0$, as a result of (9).

IV. NNS FOR COST MODIFICATION

NNs are known to be universal function approximators. Consequently, a multilayered NN can represent any continuous function under mild assumptions, see e.g. [24]. More uniquely, NNs also perform well for high dimensions, where traditional approximation methods tend to perform poorly. We propose to combine NNs with quadratic functions to parameterize the cost. The quadratic function is a good initial guess as we know it is locally valid and we know how to compute it [20]. NNs are then added as an attempt to capture everything beyond the locally valid quadratic function.

In this section we will describe both regular NNs as well as convex NNs, and how these can be combined with quadratic functions to provide nominal stability of the MPC scheme by construction.

A. REGULAR NEURAL NETWORKS

A (not necessarily convex) feedforward neural network (FNN) with F layers, for which $i = 0, \ldots, F-1$, can be formulated as

$$z_{i+1} = \sigma_i(W_i z_i + b_i), \quad v(s) = z_F, \tag{17}$$

where $z_0 = s$ is the network input, $z_i \in \mathbb{R}^{q_i \times 1}$ denotes the hidden state of layer i, $z_{i+1} \in \mathbb{R}^{q_{i+1} \times 1}$ denotes the hidden state of the next layer, so that W_i is a matrix of size $\mathbb{R}^{q_{i+1} \times q_i}$ containing the weights of layer i and $b_i \in \mathbb{R}^{q_{i+1} \times 1}$ are bias terms. The nonlinear activation function used in layer i is denoted by σ_i, and operates element-wise.

An FNN will be used to modify the parametrization of the storage function. From [20] we know that locally a quadratic storage function is sufficient to show strict dissipativity. We therefore combine the quadratic function with an FNN, according to

$$\lambda_\theta(s) = v_\theta(s) + \theta_{\lambda_0} + (s - \theta_{s_e})^\top D(\theta)(s - \theta_{s_e}), \tag{18}$$

where $v_\theta(s)$ is the FNN as defined in (17), θ_{s_e} is the parameterized steady state, θ_{λ_0} is a parameter that will be tuned so that $\lambda_\theta(\theta_{s_e}) = 0$ and $D(\theta)$ is a matrix with entries from the parameter vector θ. All NN weights $W_{0:F-1}$ and bias terms $b_{0:F-1}$ are part of the parameter vector θ.

The terminal cost is typically modelled with a quadratic function, and here combined with an FNN

$$T_\theta(s) = v_\theta(s)^2 + \theta_{T_0} + (s - \theta_{s_e})^\top (B(\theta)^\top B(\theta) + \epsilon I)(s - \theta_{s_e}), \tag{19}$$

where $B(\theta)$ is a parameter matrix, and θ_{T_0} is tuned to shift $T_\theta(\theta_{s_e})$ to zero. We use $B(\theta)^\top B(\theta) + \epsilon I$, where ϵ is a small positive constant, to ensure that the quadratic term is positive definite. For this reason we also square the output from the FNN.

Remark 3: In (19) the terminal cost is modelled with an FNN that does not preserve convexity. To ensure nominal stability the terminal cost should be at least positive definite. However, for optimization reasons, it may be beneficial to model also the terminal cost using a convex NN, as detailed next.

B. CONVEX NEURAL NETWORKS

In order to build stable MPC schemes, the parameterized stage cost must satisfy Assumption 7. Making general FNNs respect the lower bound, would entail constraining the majority of the network's weights, giving an in practice intractable optimization problem for most applications. Instead, we select convex NNs to parameterize the stage cost. In addition to the stability argument, a convex cost function is expected to alleviate difficulties when using sensitivity-based solvers. This section outlines how convex NNs can be adapted so that Assumption 7 is satisfied.

In recent years several convex NN architectures have been developed. Using convex NNs as cost modifications in MPC has, to the best of the authors' knowledge, not been done before. We consider a class of fully input convex neural networks (ICNNs) first proposed in [25]. It was proven in [26] that ICNNs are universal approximators of convex Lipschitz functions. Alternative convex NN architectures exist, as described in e.g. [27], that are richer function approximators than ICNNs, but usually require a larger number of parameters. The specific type of ICNN is selected because it offers

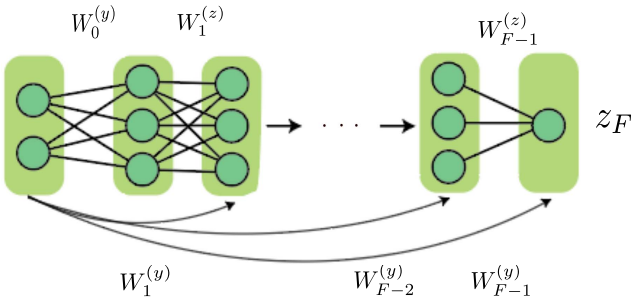


FIGURE 2. Input convex neural network.

a simpler parametrization and training process, and requires fewer parameters.

Let $y = \{s, a\}$. An ICNN with F layers as described in [25], for which $i = 0, \ldots, F-1$, can be formulated as

$$z_{i+1} = \sigma_i(W_i^{(z)} z_i + W_i^{(y)} y + b_i), \quad g(y) = z_F. \tag{20}$$

The ICNN is similar to the FNN in (17), to the exception of the additional input weights $W_i^{(y)} \in \mathbb{R}^{q_{i+1} \times (n+m)}$ and the input to the ICNN $y \in \mathbb{R}^{(n+m)\times 1}$, that here enters every hidden layer. Also, the ICNN requires the activation σ_i to be a convex and non-decreasing nonlinear activation function. An example of this is given in Section IV-C. For the input layer we have that $W_0^{(z)} = 0$. The network is visualized in Fig. 2.

Proposition 2: The function g is convex in y provided that all terms in $W_{1:F-1}^{(z)}$ are non-negative, and all functions σ_i are convex and non-decreasing.

Proof: This is straightforward to prove, using the fact that non-negative sums of convex functions are also convex and that the composition of a convex and convex non-decreasing function is convex [28].

By limiting all weights $W_{1:F-1}^{(z)}$ to be non-negative, the function $g(y)$ will be a convex function with respect to its input. We model the stage cost as

$$\hat{\ell}_\theta(y) = g_\theta(y) + \theta_{\ell_0} + (y - \theta_e)^\top (M(\theta)^\top M(\theta) + \epsilon I)(y - \theta_e), \tag{21}$$

where θ_{ℓ_0} is a dedicated parameter used to shift $\hat{\ell}_\theta(\theta_e)$ to zero and θ_e contains the parameterized steady state, i.e. $\theta_e = (\theta_{s_e}, \theta_{a_e})$. Moreover, we use $M(\theta)^\top M(\theta) + \epsilon I$ to ensure that the quadratic term is positive definite. The quadratic term will ensure that for a (not necessarily strictly) convex function, θ_e will be the only global minimum. The quadratic term will also ensure that the resulting function is radially unbounded with respect to s. In addition to the function being convex, we also need the global minimum of the function to be zero at steady state, i.e.

$$\hat{\ell}_\theta(\theta_e) = 0, \quad \nabla_y \hat{\ell}_\theta(\theta_e) = 0. \tag{22}$$

This is satisfied by construction as the parameters are updated. Next, we will formally establish that the stage cost is lower-bounded by a $\mathcal{K}_\infty$-function. As a result, the parameterized MPC scheme ensures nominal stability by construction.

Theorem IV.1: Let (22) hold for $\hat{\ell}_\theta(s, a)$ modelled as in (21). Then the parameterized stage cost (21) satisfies

$$\rho(||s - \theta_{s_e}||) \leq \hat{\ell}_\theta(s, a), \quad \hat{\ell}_\theta(\theta_{s_e}, \theta_{a_e}) = 0. \tag{23}$$

Proof: The ICNN term, $g_\theta(y) + \theta_{l_0}$, and the quadratic term, $(y - \theta_e)^\top (M(\theta)^\top M(\theta) + \epsilon I))(y - \theta_e)$, are both convex functions. The addition of the quadratic term ensures that the stage cost becomes strictly convex, so that it will have at most one global minimum. The quadratic term also ensures that the cost will be radially unbounded. Because (22) holds, the only global minimum will be at $(\theta_{s_e}, \theta_{a_e})$, and consequently the stage cost satisfies Lemma III.1, and $\hat{\ell}_\theta(\theta_{s_e}, \theta_{a_e}) = 0$.

C. CHOICE OF ACTIVATION FUNCTIONS

Convexity of the ICNN is dictated by proposition 2, which requires convex and non-decreasing activation functions. By selecting a smoothed version of the rectified linear unit (ReLU), such as the softplus function, the specified convexity properties are ensured. The fact that this function is also continuously differentiable, may ease optimization of the MPC problem. The softplus function is given by

$$\sigma(x) = \ln(1 + \exp(x)). \tag{24}$$

For the cost terms modelled by the FNNs, we have no limitations on the choice of activation functions, except the requirement on continuity. As stated in Section II-A, all cost terms should be continuous functions. For the NN terms this is dictated by the choice of activation function, and this is satisfied for all the most popularly used activation functions.

V. RL FOR PARAMETER UPDATES

For dissipative economic problems, finding a storage function that allows us to recast the ENMPC as a stable tracking MPC, is a difficult problem. Recently, new methods have been proposed to build these conditions, either via sum-of-squares programming [29], or via techniques borrowed from RL [4]. The latter approach allows one to build ENMPC schemes that are optimal and whose stability is established by construction rather than by verification. This framework also introduces the additional flexibility to tackle non-dissipative problems. Indeed, for non-dissipative problems, the proposed framework can be used to find a controller that as closely as possible resembles the optimal unstable policy for the problem at hand. In this section, we will consider how RL can be used to perform parameter updates of parameterized MPC schemes such that the requirements outlined in Section IV, are ensured.

A. Q-LEARNING

Q-learning is an RL method based on learning the optimal action-value function $Q^\star(s, a)$ [10]. Using MPC as a function approximator, we can estimate the optimal Q-function by constraining the first action in the input sequence, according to

$$Q_\theta(s, a) = \min_{x,u} \; -\lambda_\theta(s) + T_\theta(x_N) + \sum_{k=0}^{N-1} \hat{\ell}_\theta(x_k, u_k) \tag{25a}$$

$$\text{s.t} \quad (14b) - (14d), \tag{25b}$$

$$u_0 = a. \tag{25c}$$

It can be shown that the Q-function estimate from (25), the value function estimate (14) and the policy (15) satisfies the Bellman equations [4]. Q-learning seeks to provide the solution to the following least-squares problem

$$\min_{\theta} \ \mathbb{E}\left[\frac{1}{2}(Q_\theta(s,a) - Q^\star(s,a))^2\right]. \tag{26}$$

One approach to solving this, is updating the parameters using temporal difference (TD) methods. For the undiscounted setting the parameter update is (see e.g. [10])

$$\delta_k = \ell(s_k, a_k) + V_\theta(s_{k+1}) - Q_\theta(s_k, a_k), \tag{27a}$$

$$\Delta\theta_Q = \alpha\delta_k \nabla_\theta Q_\theta(s_k, a_k), \tag{27b}$$

where α denotes the learning step size, δ_k the TD error at time step k and $\Delta\theta_Q = \theta_{k+1} - \theta_k$.

For Q-learning techniques it should be mentioned that there is no guarantee to find the optimal policy. This is because the parameter update of Q-learning methods is not designed to optimize closed-loop performance directly. Instead, Q-learning aims to fit Q_θ as closely as possible to $Q^\star$, and assumes that $Q_\theta \approx Q^\star$ results in $\pi_\theta \approx \pi^\star$. However, there are no guarantees that the former approximation implies the latter, and for certain shapes of Q-functions, it still may be challenging to capture the optimal policy, even with an almost correct Q-function estimate. In this scenario, policy-based methods such as deterministic policy gradient methods may be more suited.

B. DETERMINISTIC POLICY GRADIENT METHODS

The lack of convergence guarantees for Q-learning methods has motivated the need for alternative methods with more formal convergence guarantees. Using policy gradient methods, the parameters are updated towards improving the performance of the policy irrespective of the action-value function accuracy. The policy performance index used in the undiscounted setting, for a deterministic policy, is defined as follows

$$J_\theta(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{k=0}^{\infty} \ell(s_k, a_k) \mid a_k = \pi_\theta(s_k)\right]. \tag{28}$$

The expectation $\mathbb{E}_{\pi_\theta}[\cdot]$ can be estimated by taking the mean of trajectories generated by the policy π_θ. The policy gradient in the deterministic case is given by [30]

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \pi_\theta(s) \nabla_a Q_\theta(s,a)|_{a=\pi_\theta(s)}\right]. \tag{29}$$

The condition for optimality is $\nabla_\theta J(\pi_\theta) = 0$, and the parameters are then updated in the direction of policy improvement, according to

$$\Delta\theta_J = -\alpha\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \pi_\theta(s) \nabla_a Q_\theta(s,a)|_{a=\pi_\theta(s)}\right]. \tag{30}$$

The deterministic policy gradient is used to derive a range of actor-critic algorithms [30].

C. SENSITIVITY ANALYSIS

The gradients needed for Q-learning are found from sensitivity analysis of the parameterized MPC scheme in (25). The Lagrange function for the optimization problem is

$$\mathcal{L}_\theta = \Phi_\theta + \nu^\top G + \mu^\top H, \tag{31}$$

where Φ_θ is the cost (25a), H gathers the inequality constraints and G the equality constraints in (25). The variables ν and μ are Lagrange multipliers associated with the equality constraints and inequality constraints respectively. Let p label the primal decision variables and let $\eta = \{p, \nu, \mu\}$. The solution to the MPC problem (25) is then given by $\eta^\star$. The gradient of $Q_\theta(s,a)$, needed in (27b), is then

$$\nabla_\theta Q_\theta(s,a) = \nabla_\theta \mathcal{L}_\theta(s, \eta^\star). \tag{32}$$

The policy gradient $\nabla_\theta \pi_\theta$ required in (30), is found by considering the MPC scheme in (14). The primal-dual Karush Kuhn Tucker (KKT) conditions are given by

$$R = \begin{bmatrix} \nabla_p \mathcal{L}_\theta \\ G \\ \text{diag}(\mu)\text{H} \end{bmatrix} = 0, \tag{33}$$

where $\text{diag}(\mu)$ is a diagonal matrix with entries μ. Using the implicit function theorem, it follows that

$$\nabla_\theta \pi_\theta(s) = -\nabla_\theta R(\eta^*, s, \theta) \nabla_\eta R(\eta^*, s, \theta)^{-1} \frac{\partial \eta}{\partial u_0}. \tag{34}$$

The other gradient needed in policy gradient methods, $\nabla_a Q_\theta(s,a)$, can under certain conditions be derived using a simple approximator of the action-value function. For further details the reader is referred to [30].

D. CONSTRAINED RL STEPS

In order to ensure convexity of the ICNN, we need to perform constrained RL steps so that selected weights in the network stay non-negative. This applies to the hidden state weights $W^{(z)}_{1:F-1}$ in (20). We also use the constrained RL update to ensure that all NNs are zero at steady state, and to ensure that the global minimum of the stage cost is zero at steady state i.e. that (22) holds.

Let d denote the proposed step by RL at time step k, i.e. we could have either $d = \Delta\theta_Q$ in case of using Q-learning or $d = \Delta\theta_J$ if using policy gradient methods. We can then define the following optimization problem to constrain d, so that the parameter update respects the constraints as detailed above

$$\min_{\Delta\theta} \ \frac{1}{2}\|\Delta\theta\|^2 - d^\top \Delta\theta \tag{35a}$$

$$\text{s.t} \quad w_i + \Delta\theta_i \geq 0 \quad \text{for} \quad i = 1, \ldots, r, \tag{35b}$$

$$\hat{\ell}_{\theta_{k+1}}(\theta_{k+1,e}) = 0, \tag{35c}$$

$$\nabla_y \hat{\ell}_{\theta_{k+1}}(\theta_{k+1,e}) = 0, \tag{35d}$$

$$\lambda_{\theta_{k+1}}(\theta_{k+1,s_e}) = 0, \tag{35e}$$

$$T_{\theta_{k+1}}(\theta_{k+1,s_e}) = 0, \tag{35f}$$

where $\Delta\theta = \theta_{k+1} - \theta_k$ is the constrained update of the entire parameter vector and w_i are the constrained elements in the ICNN weight matrices $W^{(z)}_{1:F-1}$. The r constrained weights w_i are also part of the parameter vector θ, and we have that $w_i = \theta_{i,k}$. It can be shown that if there exists a constrained parameter update $\Delta\theta$ at $k = 0$ by solving the optimization problem (35), such that (35b)-(35f) are satisfied, then there must exist a feasible solution $\Delta\theta\ \forall k > 0$.

Remark 4: The constraints in (35) are not necessarily ensured at $k = 0$ for the initial values of the parameters θ, even for NNs that are pre-trained to quadratic functions. To guarantee that these constraints hold at $k = 0$, the constraints can be enforced either during or after pre-training.

Lemma V.1: If learning converges, the constrained parameter update found from solving (35) will yield the true optimal parameters θ, i.e. the parameter values that minimize the function with gradient step d.

Proof: Let $\Psi(\theta)$ denote the function that RL is trying to minimize, i.e. $d = -\alpha\nabla\Psi(\theta)$. We formulate the original optimization problem as

$$\min_{\theta} \ \alpha\Psi(\theta) \tag{36a}$$

$$\text{s.t} \ \ Z(\theta) = 0, \tag{36b}$$

$$\Gamma(\theta) \leq 0, \tag{36c}$$

where Z and Γ are matrices that gather the equality and inequality constraints in (35) respectively. For (36) the stationarity of the KKT conditions is given as

$$\alpha\nabla\Psi(\theta) + \phi^\top\nabla Z + \xi^\top\nabla\Gamma = 0, \tag{37}$$

where ϕ and ξ are the multipliers associated with equality and inequality constraints, respectively. The stationarity of the KKT conditions for (35) are

$$\Delta\theta - d + \phi^\top\nabla Z(\theta + \Delta\theta) + \xi^\top\nabla\Gamma(\theta + \Delta\theta) = 0. \tag{38}$$

As learning converges, $\Delta\theta \approx 0$. Using this, and the fact that $d = -\alpha\nabla\Psi(\theta)$, we see that we obtain the same expression for stationarity of the KKT conditions. Note that one can also show that the primal/dual feasibility conditions and the complementary slackness condition are the same. The two optimization problems therefore share the same optimal values of θ.

VI. COMBINING Q-LEARNING AND POLICY GRADIENT METHODS

Policy-based methods have several advantages over value-based RL methods. First and most important, these methods are more reliable when it comes to improving the policy, as they are designed based on optimality of the closed-loop policy. Second, certain types of policy-based methods are also known to be more sample-efficient than Q-learning [10]. However, there may be parameters that the MPC policy gradient will be insensitive to. Mathematically this entails that certain parameters lie in the null space of the policy gradient.

This is especially relevant for rich parametrizations, as they contain more parameters. Although certain parameters may not influence the optimal policy, we may still want to update them, in order to e.g. capture the correct shape of the value and action-value function. In this context we propose to embed Q-learning, as a measure to handle the parameters that the MPC policy may not be sensitive to. As the Q-function and the policy are jointly unique functions, the parameters should affect at least one of these functions.

A. NULL SPACE METHOD

For an ENMPC problem recast as a tracking MPC, policy gradient methods will not be sufficient for tuning the cost parametrization, as the MPC policy is insensitive to the storage function. Hence, we will use a policy gradient method and combine it with Q-learning using a null space method. More specifically we aim at using a policy gradient method to update parts of the parameters in order to converge to the correct policy, and perform Q-learning steps in the null space of the policy gradient to shape the action-value function with the remaining parameters. For a parametrization that is rich enough, the correct action-value function should be captured without conflicting with the policy approximation. Whereas the use of both Q-learning and policy gradient methods for adjusting a parameterized MPC scheme is well established, we now introduce a new method for combining RL algorithms.

In order to formulate the null space of the policy gradient update, we consider an approximation of the Hessian of the policy gradient, given by the Fisher information matrix [31]:

$$\nabla^2_\theta J_\theta \approx \mathbb{E}_{\pi_\theta}[\nabla_\theta\pi_\theta(s)\nabla_\theta\pi_\theta(s)^\top]. \tag{39}$$

Alternatively, a more accurate approximation of the Hessian can be found in [32]. We then define the null space of $\nabla^2_\theta J_\theta$ as the matrix $\mathcal{N}$, such that $\nabla^2_\theta J_\theta\mathcal{N} = 0$. The parameter update resulting from Q-learning (27) is then projected to the null space of the policy gradient according to

$$\Delta\theta^{\mathcal{N}}_Q = \mathcal{N}(\mathcal{N}^\top\mathcal{N})^\dagger\mathcal{N}^\top\Delta\theta_Q, \tag{40}$$

where $\cdot^\dagger$ denotes the Moore-Penrose pseudo-inverse. The full parameter update resulting from combining policy gradient and Q-learning is then given by

$$\Delta\theta = \Delta\theta_J + \Delta\theta^{\mathcal{N}}_Q. \tag{41}$$

VII. NUMERICAL EXAMPLES

In this section we propose two numerical examples to illustrate the proposed method. The first example is a seemingly simple case of an economic linear quadratic regulator (LQR), i.e. an LQR with weighting matrices that are not positive definite. The example becomes challenging because of the shape of the action-value function that calls for a combination of RL methods in order to capture both the correct policy and value function. The second simulation example is a chemical reactor with nonlinear dynamics and an economic cost function. The ENMPC scheme is recast as a tracking MPC scheme, using a parameterized cost and storage function. We combine the

TABLE I NNs for LQR Example

Cost term	NN class	Num. hidden layers	Num. neurons	Pre-trained
$\hat{\ell}_\theta(s,a)$	ICNN	2	16, 16	✓
$\hat{\ell}_\theta(s,a)$	FNN	2	16, 16	✓
$\lambda_\theta(s)$	FNN	2	16, 16	✓
$T_\theta(x_N)$	FNN	2	16, 16	✗

convex NN-based cost modifications and quadratic functions for all cost terms, ensuring nominal stability of the MPC, and benchmark its performance against the standard quadratic cost parametrization.

A. ECONOMIC LQR

We consider an economic LQR for a system with dynamics

$$s_{k+1} = 0.1s_k + a_k, \tag{42}$$

and stage cost

$$L(s,a) = -s^2 + 10a^2. \tag{43}$$

For the sake of satisfying Assumption 1, we introduce the following artificial constraints

$$-100 \leq a \leq 100, \quad -100 \leq s \leq 100. \tag{44}$$

For the set of states that never activate the constraints, we can solve the Riccati equation for the discrete system, and obtain the optimal value function and policy. For the dynamics (42) and stage cost (43) in the unconstrained case, the optimal value function and policy is

$$V^\star(s) = -1.0113s^2, \ \pi^\star(s) = 0.0113s. \tag{45}$$

In the first set of simulations, we will make a comparison of the ICNN and the FNN. For this purpose, we will assume that both the true dynamics and the optimal steady state pair are available. We formulate the following finite-horizon linear MPC scheme

$$\min_{x,u} \ -\lambda_\theta(s) + T_\theta(x_N) + \sum_{k=0}^{N-1} \hat{\ell}_\theta(x_k, u_k) \tag{46a}$$

$$\text{s.t} \ \ x_{k+1} = 0.1x_k + u_k \quad x_0 = s, \tag{46b}$$

$$-100 \leq x_k \leq 100, \tag{46c}$$

$$-100 \leq u_k \leq 100, \tag{46d}$$

with prediction horizon $N = 10$. For this example, we used NNs to model all cost terms in (46). In order to get suitable initial values of the weights, we pre-trained the NNs to quadratic functions. This was done using Keras in Python [33]. The architecture used to parameterize each cost term, is reproduced in Table 1. We stress that this simulation example is mainly providing a proof of concept, and therefore that the choices related to the architectures of the NNs have not been optimized. System (42) is simulated from random initial conditions on the interval $[-1, 1]$, for episodes of length 10.

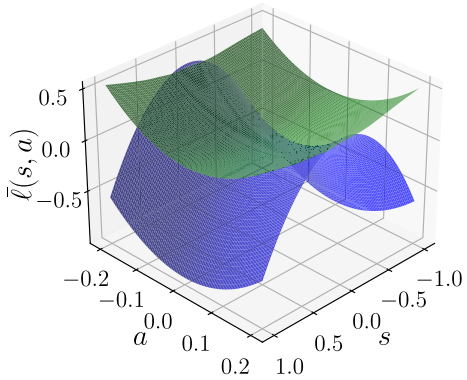


FIGURE 3. Resulting rotated stage cost (9) when using an ICNN (green) and an FNN (blue) to model the stage cost respectively.

For this example, regular Q-learning manages to capture the Q-function fairly accurately, but struggles to capture the optimal policy. This is likely explained by the shape of the Q-function, which turns out to be fairly insensitive to the policy, causing small errors in the Q-function to give large errors in the resulting policy. This clearly illustrates a known weakness in Q-learning, and we therefore resort to a combination of Q-learning and deterministic policy gradient methods using a null space method as described in Section VI. The gradient of Q needed to formulate the policy gradient can be computed from data using a range of algorithms. For convenience, we use the true Q-function to formulate the gradient needed in (29), that is

$$Q_\theta(s,a) = -s^2 + 10a^2 + (0.1s + a)^2 P, \tag{47}$$

where P is the solution to the Riccati equation. The derivative with respect to the action is then

$$\nabla_a Q_\theta(s,a) = 0.2Ps + 2(10 + P)a. \tag{48}$$

Furthermore, we use gradient descent with learning rate $\alpha = 0.02$ for both the Q-learning and policy gradient update. A total of 2500 episodes were simulated in order to update the parameters, yielding a total of 2.5×10^4 learning samples. For the same hyperparameter values, we tested learning using both an ICNN and an FNN to model the stage cost. Given the learned storage function, we are able to obtain the rotated cost given by (9). This is plotted for the ICNN and the FNN in Fig. 3. Because the curvature is much larger in the action dimension, we have adjusted the axes in order to highlight the curvature in s-direction.

We see that with an FNN to model the stage cost, learning may fail to capture the correct storage function, and consequently we obtain a stage cost that is not lower-bounded by a $\mathcal{K}_\infty$-function. We stress that for a subset of simulations, using FNNs would also produce stage costs that are lower-bounded by a $\mathcal{K}_\infty$-function. In other words, with FNNs you may risk that learning fails to modify the cost, whereas when using an ICNN, we successfully learned the modified cost in every simulation.

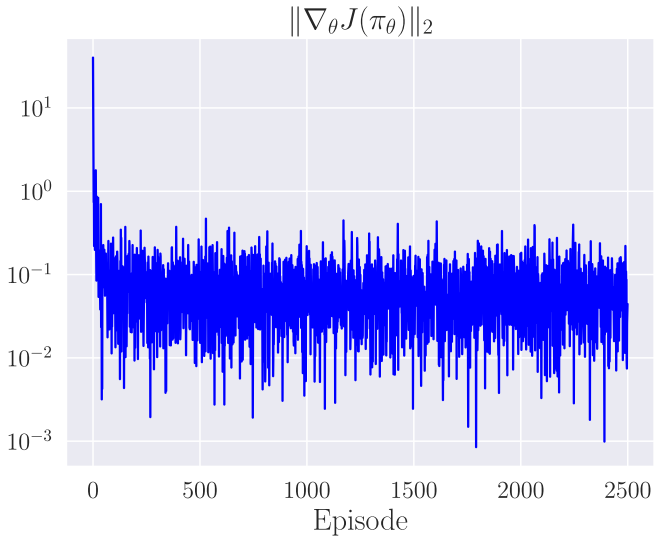


FIGURE 4. Policy gradient over episodes.

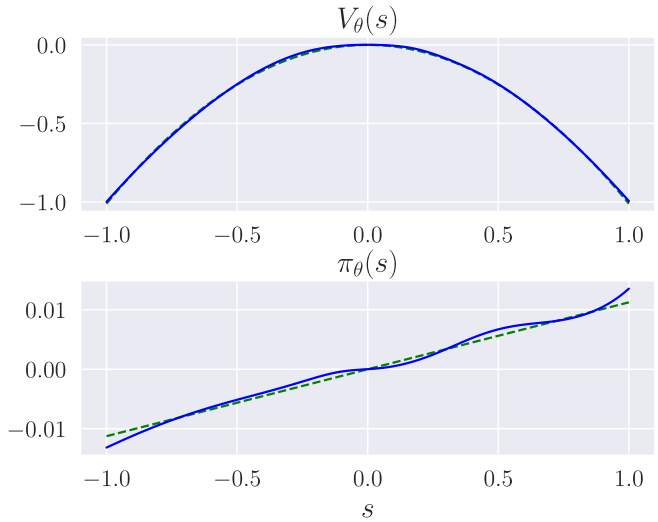


FIGURE 5. Approximated value function and policy using only NNs to parameterize all cost terms (blue). Optimal value function and policy for the unconstrained case (green, dashed).

For the second set of simulations, we will demonstrate that MPC with the proposed cost parametrization, successfully learns the optimal policy. For this demonstration, we assume that both the true dynamics and the optimal steady state are unknown. We used the following inaccurate prediction model in the MPC scheme:

$$x_{k+1} = 0.098x_k + 1.02u_k \tag{49}$$

We parameterized the steady state parameters, and wrongly initialized the parameters with $\theta_e = (0.3, 0.3)$. We ran a total of 2500 episodes of length 10, resulting in 2.5×10^4 learning samples, using $\alpha = 0.01$. In Fig. 4 we have plotted the evolution of the policy gradient over episodes. We note that this is noisy due to random choices of initial conditions.

In Fig. 5 we have plotted the approximation of the value function and the policy, using the final updated values of the parameters. We see that the value function is captured accurately, whereas the policy approximation has some inaccuracies. This is likely improved by increasing the number of learning samples.

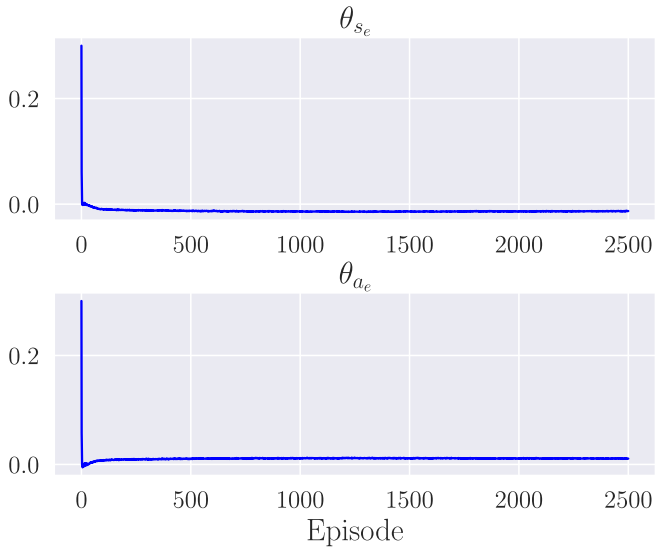


FIGURE 6. Evolution of steady state parameters.

In Fig. 6 we have plotted the evolution of the steady state parameters over episodes. We see that the steady state parameters converge very close to the optimal steady state, namely $(\theta_{s_e}, \theta_{a_e}) = (0, 0)$.

B. ENMPC: CHEMICAL REACTOR

The next simulation example has nonlinear dynamics and an economic stage cost, and we expect a quadratic cost function to be valid only locally. It is therefore suitable for testing the addition of NNs to a quadratic cost parametrization. We consider a continuously stirred tank reactor (CSTR), with an economic cost as described in [34]. The CSTR describes a non-isothermal reactor, where an exothermic reaction, converting reactant A to product B, takes place. The dynamics are given as

$$\dot{C}_A = \frac{F}{V_R}(C_{A0} - C_A) - k_0 e^{-E/RT} C_A^2 \tag{50a}$$

$$\dot{T} = \frac{F}{V_R}(T_0 - T) - \frac{\Delta H k_0}{\rho_R C_p} e^{-E/RT} C_A^2 + \frac{q}{\rho_R C_p V_R}, \tag{50b}$$

where T is the temperature in the reactor, C_A is the concentration of the reactant A, F is the flow rate and q is the heat rate. The same quantities constitute the states and inputs, i.e. $s = [C_A, T]$ and $a = [F, q]$ respectively. The additional parameters are listed in the appendix. The inputs are constrained according to

$$[0, -2 \times 10^5] \leq a \leq [10, 2 \times 10^5]. \tag{51}$$

The economic cost is

$$L = -\omega F(C_{A0} - C_A) + \beta q, \tag{52}$$

TABLE II NNs for CSTR Example

Cost term	NN class	Num. hidden layers	Num. neurons	Pre-trained
$\hat{\ell}_\theta(s,a)$	ICNN	2	16, 16	✗
$\lambda_\theta(s)$	FNN	2	16, 16	✗
$T_\theta(x_N)$	ICNN	2	16, 16	✗

where $\omega = 1.7 \times 10^4$ and $\beta = 1$ so that the production rate and energy consumption will be balanced. The dynamics in (50) and cost in (52) describe a non-dissipative problem, i.e. the closed-loop system does not converge to the optimal steady state. With the proposed MPC scheme we will learn a stable controller by design, i.e. we will obtain a controller that matches the true, unstable economic policy as closely as possible while maintaining stability. To get the dynamics on the form of (1), the equations in (50) were discretized using the Euler method with a step size of 0.02 hours. According to (3), the optimal steady state of the system is

$$s_e = [0.7572, 497.71], \quad a_e = [10, 1.38557 \times 10^5]. \tag{53}$$

We assume that we only have an inaccurate prediction model available, which we define by introducing the following errors in the dynamics described by (50):

$$\dot{C}_A = 0.85\frac{F}{V_R}(C_{A0} - 0.85C_A) - 1.2k_0 e^{-E/R0.95T}C_A^2 \tag{54a}$$

$$\dot{T} = 0.85\frac{F}{V_R}(T_0 - T) - \frac{\Delta H 1.2k_0}{\rho_R C_p}e^{-E/R0.95T}C_A^2 + \frac{0.8q}{\rho_R C_p V_R}. \tag{54b}$$

The MPC scheme is formulated with the inaccurate prediction model and a parameterized cost according to

$$\min_{x,u} \; -\lambda_\theta(s) + T_\theta(x_N) + \sum_{k=0}^{N-1} \hat{\ell}_\theta(x_k, u_k) \tag{55a}$$

$$\text{s.t } x_{k+1} = \tilde{f}(x_k, u_k), \quad x_0 = s, \tag{55b}$$

$$h(u_k) \leq 0, \tag{55c}$$

with a prediction horizon of $N = 10$, where $\tilde{f}(x_k, u_k)$ is the discretized version of (54). The stage and terminal cost were modelled with quadratic functions and convex cost modifications as in (21), whereas for the storage function we used a quadratic function and an FNN as in (18). Because the true model is unknown, we also parameterized the steady state. The steady state parameters were initialized with values found by evaluating (3) for the inaccurate prediction model described by (54). The architecture for each NN is specified in Table 2. The quadratic terms were initialized with $M(\theta) = I_{n+m}, B(\theta) = I_n, D(\theta) = 10 \times I_n$, where I is the identity matrix. The NN weights and bias terms were initialized to small, random numbers. All parameters were updated with Q-learning until convergence. Because the combination of quadratic functions and NNs will introduce many parameters, that may also differ by orders of magnitude, we used Adam optimization for updating the parameters. Adam optimization differs from gradient descent by computing individual learning rates for each parameter. The hyperparameters typically require little tuning, and we used standard values, see [35]. Because the states and inputs span different orders of magnitude, we used input normalization to force the NN input variables into the range [0, 1]. Input normalization used correctly is known to reduce estimation errors as well as speed up convergence, see for instance [36].

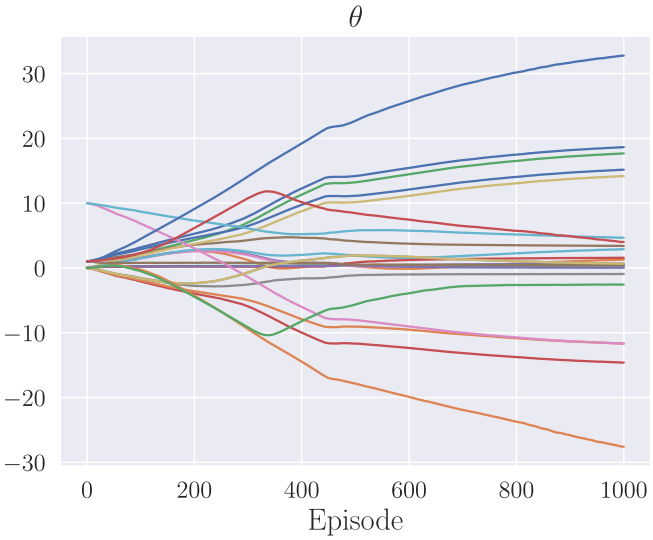


FIGURE 7. Evolution of the quadratic parameters during the first 1000 episodes.

The closed-loop system was first simulated with quadratic cost terms, i.e. we parameterize the cost with only the quadratic terms in (18), (19) and (21). We simulated for 1000 episodes of length 60, with a learning rate of $\alpha = 1 \times 10^{-3}$, until performance converged. The evolution of the quadratic parameters over the episodes is plotted in Fig. 7. The result from the first 1000 episodes was used as a benchmark for the rich cost parameterization.

The rich cost parametrization was obtained by adding NNs to the already trained quadratic cost terms after 1000 episodes. We then continued learning for 500 more episodes of shorter length, as we were mainly hoping to improve performance in the transients. The closed-loop performance during learning is plotted for all episodes in Fig. 8. We note that, as for the previous simulation example, this plot is noisy due to the random choice of initial conditions. The closed-loop performance initially worsens, before improving and converging for the quadratic cost parameterization. After 1000 episodes we see that introducing the NNs creates a new peak in performance, before converging to a slightly smaller i.e. better mean than before. The addition of the NN-based cost modifications clearly gave a modest improvement in performance. This is most likely because the quadratic cost terms minimize the economic objective almost as much as possible while maintaining stability, and we find that the closed-loop performance is close to the optimal economic performance. For this example we

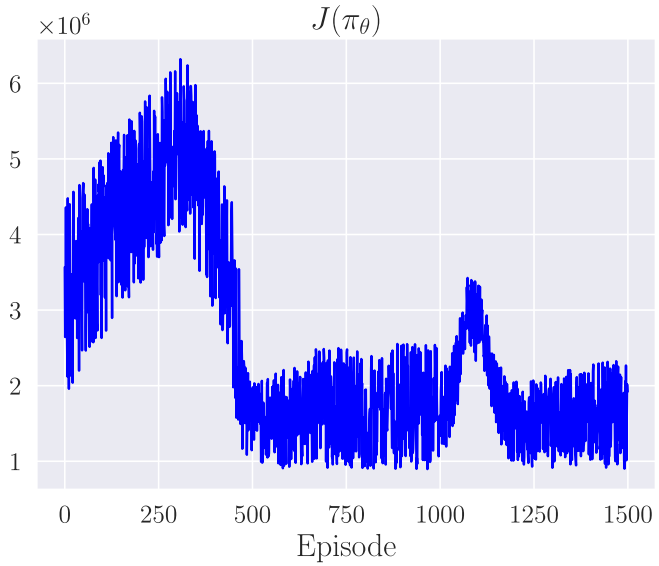


FIGURE 8. Closed-loop performance during training of the quadratic parameters (0-1000 episodes), and of the quadratic and NN parameters (after 1000 episodes).

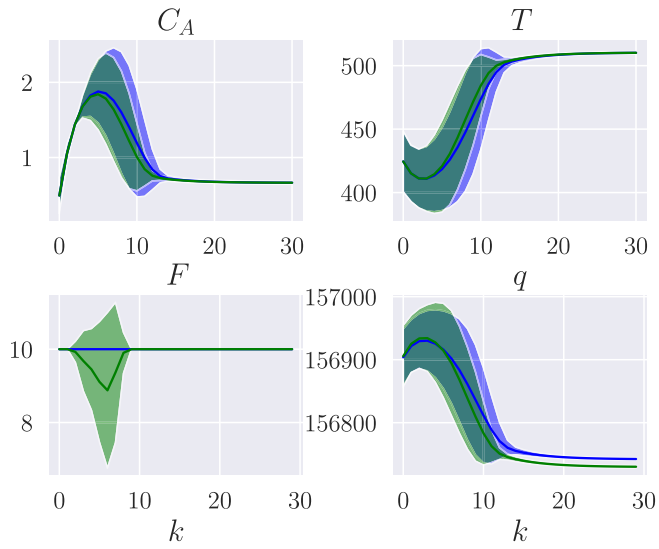


FIGURE 9. Simulation with learned controllers, using a quadratic parameterization (blue) and a combination of quadratic functions and NNs (green).

will not be able to achieve the optimal economic behaviour as we are learning a stable controller by construction and consider a non-dissipative problem.

In order to further evaluate the performance of each cost parametrization, we have also compared their performance in closed-loop using the final values of the updated parameters. In order to show that the learned controller is robust to model error, we added parametric uncertainty in the pre-exponential rate factor k_0. For each random initial condition, we also drew a new value of $\tilde{k}_0$ where $\tilde{k}_0 \sim \mathcal{N}(k_0, \sigma_k^2)$ with $\sigma_k = 2.1156 \times 10^5$. In Fig. 9 we have plotted the mean and two standard deviations of simulations from 50 randomly selected

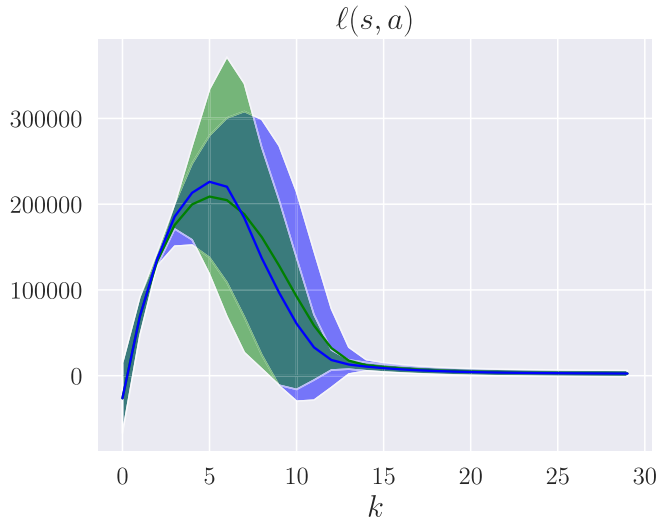


FIGURE 10. Economic cost in simulations with learned controllers, using a quadratic cost parameterization (blue) and a combination of quadratic functions and NNs (green).

initial conditions and values of $\tilde{k}_0$. We see that the addition of the NNs does not alter the closed-loop trajectories much, except noticeably for the flow rate F. Also, both controllers converge to slightly different steady states than that obtained by evaluating (3) for the true system. However, we found that the shifted economic cost, $\ell(s, a)$, of the original and the learned steady state is practically the same. This is plotted in Fig. 10. In Fig. 10 we see clearly that, as previously stated, at steady state the quadratic cost parametrization is sufficient to obtain the optimal economic cost. Although the effect of adding NN-based cost modifications on performance was limited in this case, we see that the small improvement that does occur happens in the transients.

VIII. CONCLUSION

In this paper we have considered the use of convex cost modifications, using neural networks (NNs). We have applied this framework to economic nonlinear model predictive control (ENMPC). By invoking dissipativity theory, we have recast the ENMPC as a tracking MPC scheme, with the additional storage function. Convexity properties of the learned stage cost have been leveraged in order to ensure the appropriate lower bound necessary to establish nominal closed-loop stability, as well as alleviating numerical difficulties when solving the MPC problem. We have outlined how reinforcement learning (RL) can be used to adjust the parametrized cost, including the weights of the NN, so that the tracking MPC scheme delivers the optimal policy, even with an inaccurate prediction model. For a challenging case of economic linear quadratic regulator (LQR), we have demonstrated how a combination of RL methods can be used to update the parameters, so that both the correct policy and value function is learned. For a nonlinear chemical reactor, we have benchmarked the combination of quadratic functions and NNs, against a standard quadratic parametrization. For this

particular simulation example, the addition of NN-based cost modifications resulted in a small improvement in closed-loop performance. This suggests that for many applications quadratic functions are rich enough. Future work will involve identifying examples where richer cost functions are expected to improve performance substantially.

APPENDIX

TABLE III CSTR Parameter Definitions and Values

Symbol	Description	Value
C_{A0}	Feed concentration of A	3.5 kmol/m^3
T_0	Feedstock temperature	300 K
V_R	Reactor fluid volume	1.0 m^3
E	Activation energy	5.04×10^4 kJ/kmol
k_0	Pre-exponential rate factor	8.46×10^6 m^3/kmolh
ΔH	reaction enthalpy change	-1.16×10^4 kJ/kmol
C_p	Heat capacity	0.231 kJ/kgK
ρ_R	Density	1000 kg/m^3
R	Gas constant	8.314 kJ/kmolK

ACKNOWLEDGMENT

The authors gratefully acknowledge the support by the industry partners Borregaard, Elkem, Hydro, Yara and the Research Council of Norway through the project Towards Autonomy in Process Industries (TAPI), project number 294544, and the project Safe Reinforcement Learning using MPC (SARLEM), project number 300172.

REFERENCES

[1] N. Lanzetti, Y. Z. Lian, A. Cortinovis, L. Dominguez, M. Mercangöz, and C. Jones, "Recurrent neural network based MPC for process industries," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 1005–1010.

[2] F. Bünning, A. Schalbetter, A. Aboudonia, M. H. D. Badyn, P. Heer, and J. Lygeros, "Input convex neural networks for building MPC," in *Proc. Learn. Dyn. Control*, 2021, pp. 251–262.

[3] K. Seel, E. I. Grøtli, S. Moe, J. T. Gravdahl, and K. Y. Pettersen, "Neural network-based model predictive control with input-to-state stability," in *Proc. Amer. Control Conf.*, 2021, pp. 3556–3563.

[4] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Trans. Autom. Control*, vol. 65, no. 2, pp. 636–648, Feb. 2020.

[5] S. Gros and M. Zanon, "Learning for mpc with stability & safety guarantees," *Automatica*, vol. 146, 2022, Art. no. 110598.

[6] M. Zanon and S. Gros, "Safe reinforcement learning using robust mpc," *IEEE Trans. Autom. Control*, vol. 66, no. 8, pp. 3638–3652, 2020.

[7] S. Gros and M. Zanon, "Bias correction in reinforcement learning via the deterministic policy gradient method for MPC-based policies," in *Proc. Amer. Control Conf.*, 2021, pp. 2543–2548.

[8] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Trans. Autom. Control*, vol. 66, no. 8, pp. 3638–3652, Aug. 2021.

[9] S. Gros, M. Zanon, and A. Bemporad, "Safe reinforcement learning via projection on a safe set: How to achieve optimality?," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8076–8081, 2020.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA:MIT Press, 2018.

[11] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 6059–6066.

[12] S. Gros and M. Zanon, "Reinforcement learning for mixed-integer problems based on MPC," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5219–5224, 2020.

[13] M. Zanon, S. Gros, and A. Bemporad, "Practical reinforcement learning of stabilizing economic MPC," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 2258–2263.

[14] W. Cai, A. B. Kordabad, H. N. Esfahani, A. M. Lekkas, and S. Gros, "MPC-based reinforcement learning for a simplified freight mission of autonomous surface vehicles," in *Proc. IEEE 60th Conf. Decis. Control*, 2021, pp. 2990–2995.

[15] A. B. Martinsen, A. M. Lekkas, and S. Gros, "Combining system identification with reinforcement learning-based MPC," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8130–8135, 2020.

[16] A. B. Kordabad and S. Gros, "Q-learning of the storage function in economic nonlinear model predictive control," *Eng. Appl. Artif. Intell.*, vol. 116, 2022, Art. no. 105343.

[17] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. San Francisco, CA, USA: Nob Hill Publishing, 2017.

[18] D. Angeli, R. Amrit, and J. B. Rawlings, "On average performance and stability of economic model predictive control," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1615–1626, Jul. 2012.

[19] A. B. Kordabad and S. Gros, "Verification of dissipativity and evaluation of storage function in economic nonlinear MPC using Q-learning," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 308–313, 2021.

[20] M. Zanon, S. Gros, and M. Diehl, "A tracking MPC formulation that is locally equivalent to economic MPC," *J. Process Control*, vol. 45, pp. 30–42, 2016.

[21] D. Bertsekas, *Dynamic Programming and Optimal Control: Volume I*, vol. 1. Belmont, MA, USA:Athena Scientific, 2012.

[22] B. Chachuat, B. Srinivasan, and D. Bonvin, "Adaptation strategies for real-time optimization," *Comput. Chem. Eng.*, vol. 33, no. 10, pp. 1557–1567, 2009.

[23] A. Jadbabaie and J. Hauser, "On the stability of receding horizon control with a general terminal cost," *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 674–678, May 2005.

[24] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[25] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 146–155.

[26] Y. Chen, Y. Shi, and B. Zhang, "Optimal control via neural networks: A convex approach," in *Proc. 31st Int. Conf. Mach. Learn.*, 2018. [Online]. Available: https://openreview.net/forum?id=H1MW72AcK7

[27] G. C. Calafiore, S. Gaubert, and C. Possieri, "Log-sum-Exp neural networks and posynomial models for convex and log-log-convex data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 827–838, Mar. 2020.

[28] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[29] S. Pirkelmann, D. Angeli, and L. Grüne, "Approximate computation of storage functions for discrete-time systems using sum-of-squares techniques," *IFAC-PapersOnLine*, vol. 52, no. 16, pp. 508–513, 2019.

[30] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, vol. 1, pp. 605–619. [Online]. Available: http://proceedings.mlr.press/v32/silver14.pdf

[31] S. M. Kakade, "A natural policy gradient," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, 2001, pp. 1531–1538.

[32] A. B. Kordabad, H. N. Esfahani, W. Cai, and S. Gros, "Quasi-Newton iteration in deterministic policy gradient," in *Proc. IEEE Amer. Control Conf.*, 2022, pp. 2124–2129.

[33] F. Chollet et al., "Keras," 2015. [Online]. Available: https://github.com/fchollet/keras

[34] X. Li, L. Zhang, M. Nakaya, and A. Takenaka, "Application of economic MPC to a CSTR process," in *Proc. IEEE Adv. Inf. Manage., Commun., Electron. Automat. Control Conf.*, 2016, pp. 685–690.

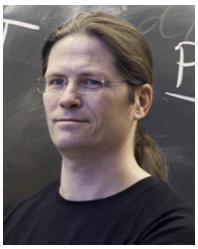
[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proc. Int. Conf. Learn. Representations*, 2014.

[36] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Trans. Nucl. Sci.*, vol. 44, no. 3, pp. 1464–1468, Jun. 1997.

KATRINE SEEL received the M.Sc. degree in marine cybernetics in 2017 from the Norwegian University of Science and Technology, Trondheim, Norway, where she is currently working toward the Ph.D. degree with the Department of Engineering Cybernetics. She holds a part-time position with SINTEF Digital, Analytics and Artificial Intelligence Group. Her research interests include combining model predictive control with learning methods, such as reinforcement learning.

ARASH BAHARI KORDABAD received the B.Sc. and M.Sc. degrees in mechanical engineering from the University of Tabriz, Tabriz, Iran, and the Sharif University of Technology, Tehran, Iran, in 2017 and 2019, respectively. Since February 2020, he has been working toward the Ph.D. degree with the Department of Cybernetics Engineering, Norwegian University of Science and Technology, Trondheim, Norway. His research interests include reinforcement learning, model predictive control, and optimization for autonomous vehicles, smart-grid applications.

SÉBASTIEN GROS received the Ph.D. degree from the Swiss Federal Institute of Technology Lausanne, Switzerland, in 2007. After a journey by bicycle from Switzerland to the Everest base camp in full autonomy, he joined an R&D Group hosted at Strathclyde University, Glasgow, U.K., focusing on wind turbine control. In 2011, he joined the University of KU Leuven, Leuven, Belgium, where his main research focus was on optimal control and fast MPC for complex mechanical systems. In 2013, he joined the Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden, where he became an Associate Professor in 2017. He is currently a full Professor with NTNU, Norway and a Guest Professor with Chalmers. His main research interests include numerical methods, real-time optimal control, reinforcement learning, and the optimal control of energy-related applications.



JAN TOMMY GRAVDAHL (Senior Member, IEEE) received the Siv.ing and Dr.ing degrees in engineering cybernetics from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 1994 and 1998, respectively. Since 2005, he has been a Professor with the Department of Engineering Cybernetics, NTNU, where he also was the Head of Department in 2008–2009. He has supervised the graduation of 160 M.Sc. and 15 Ph.D. candidates. He has authored or coauthored five books and more than 250 papers in international conferences and journals. His research interests include mathematical modeling and nonlinear control in general, in particular applied to turbomachinery, marine vehicles, spacecraft, robots, and high-precision mechatronic systems. Since 2020, he has been an Associate Editor for IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY. He was a Senior Editor of the *IFAC journal Mechatronics* in 2016–2020, and an Associate Editor for the journal *Simulation Modelling Practice and Theory* in 2007–2011. He has been on the Editorial Board and IPC for numerous international conferences. In 2000 and again in 2017, he was the recipient of the IEEE Transactions on Control Systems Technology Outstanding Paper Award.