Safe and Efficient Model Predictive Control Using Neural Networks: An Interior Point Approach

Daniel Tabas and Baosen Zhang

***Abstract*— Model predictive control (MPC) provides a useful means for controlling systems with constraints, but suffers from the computational burden of repeatedly solving an optimization problem in real time. Offline (explicit) solutions for MPC attempt to alleviate real time computational challenges using either multiparametric programming or machine learning. The multiparametric approaches are typically applied to linear or quadratic MPC problems, while learning-based approaches can be more flexible and are less memory-intensive. Existing learning-based approaches offer significant speedups, but the challenge becomes ensuring constraint satisfaction while maintaining good performance. In this paper, we provide a neural network parameterization of MPC policies that explicitly encodes the constraints of the problem. By exploring the interior of the MPC feasible set in an unsupervised learning paradigm, the neural network finds better policies faster than projection-based methods and exhibits substantially shorter solve times. We use the proposed policy to solve a robust MPC problem, and demonstrate the performance and computational gains on a standard test system.**

I. Introduction

Model predictive control (MPC) [1] is a powerful technique for controlling systems that are subject to state and input constraints, such as agricultural [2], automotive [3], and energy systems [4]. However, many applications require fast decision-making which may preclude the possibility of repeatedly solving an optimization problem online [5].

A popular approach for accelerating MPC is to move as much computation offline as possible [5], [6]. These techniques, known as explicit MPC, involve precomputing the solution to the MPC problem over a range of parameters or initial conditions. Most of the research effort has focused on problems with linear dynamics and constraints, and linear or quadratic cost functions. In this case, the explicit MPC solution is a piecewise affine (PWA) function defined over a polyhedral partition of the state constraints. However, many of the applications of interest have cost functions that are not necessarily linear or quadratic, or even convex. Further, the memory required to store the partition and affine functions can be prohibitive even for modestly-sized problems.

In order to reduce the complexity of explicit MPC, the optimal offline solution can be approximated. Approximations generally fall into two categories: partition-based solutions [7]–[9] that generate piecewise control laws over coarser state space partitions, and learning-based solutions [10]–[13] that use function approximation to compactly represent the optimal MPC policy. In this paper, we focus on the latter with the key contribution of ensuring constraint satisfaction while exploring all feasible policies.

Constraint satisfaction is crucial in many engineering applications, and the ability of MPC to enforce constraints is a major factor in its popularity. However, it is not straightforward to guarantee that a learning-based solution will satisfy constraints. The main challenge arises from the fact that while neural networks can limit their outputs to be in simple regions, there is no obvious way of forcing complex constraint satisfaction at the output. In [10], [14], supervised and unsupervised learning were used to approximate the solution of MPCs, but did not provide any feasibility guarantees. By contrast, [11] trains an NN using a policy gradient approach, and guarantees feasibility by projecting the NN output into the feasible action set. However, this extra optimization step slows down the speed of online implementation, making it difficult to use in applications that require high-frequency solutions [15]. Supervised learning approaches that provide safety guarantees [12], [13] rely on a choice of MPC oracle that is not obvious when persistent disturbances are present.

In this paper, we propose an NN architecture for approximating explicit solutions to finite-horizon MPC problems with linear dynamics, linear constraints, and arbitrary differentiable cost functions. The proposed architecture guarantees constraint satisfaction without relying on projections or MPC oracles. By exploring the *interior* of the feasible set, we demonstrate faster training and evaluation, and comparable closed-loop performance relative to other NN architectures.

The proposed approach has parallels in interior point methods for convex optimization [16]. Interior point methods first solve a *Phase I* problem to find a strictly feasible starting point. This solution is used to initialize the *Phase II* algorithm for optimizing the original problem. Our approach accelerates both phases. The Phase I solution is given by a simple function (e.g., affine map) and the Phase II problem is solved using an NN architecture that can encode arbitrary polytopic constraints (Fig. 1).

The Phase II solution builds on a technique first proposed in [17], which uses a *gauge map* to establish equivalence between compact, convex sets. With respect to [17], the current work has three novel aspects. First, the reinforcement learning (RL) algorithm in [17] only uses information about the constraints, and does not use information about the cost

This work is partially supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1762114, NSF grants ECCS-1930605 and ECCS-2023531. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Authors are with the department of Electrical and Computer Engineering, University of Washington, Seattle, WA, United States. {dtabas, zhangbao}@uw.edu.

function or dynamics. The resulting policy is safe, but can exhibit suboptimal performance. The MPC formulation in the current paper gives rise to a training algorithm that can exploit knowledge about the system, improving performance. Second, the MPC formulation permits explicit consideration for future time steps. The RL formulation cannot optimize entire trajectories due to the presence of constraints. This inability to "look ahead" again limits the performance of the RL algorithm. Finally, the previous work required a Phase I that used a linear feedback to find a strictly feasible point. A linear feedback, however, may not exist for some problems. The current work proposes a more general class of Phase I solutions (piecewise affine), while providing a way to manage the complexity of the Phase I solution.

We demonstrate the effectiveness of the proposed technique on a 3-state test system, and compare to standard projection- and penalty-based approaches for learning with constraints. The results show that the proposed technique achieves Pareto efficiency in terms of closed-loop performance and online computation effort. All code is available at `github.com/dtabas/gauge_networks`.

Notation: The p-norm ball for $p \geq 1$ is $\mathbb{B}_p = \{z \mid \|z\|_p \leq 1\}$. A *polytope* $\mathcal{P} \subset \mathbb{R}^n := \{z \in \mathbb{R}^n \mid Fz \leq g\}$ is the (bounded) intersection of a finite number of halfspaces. Scaling of polytopes by a factor $\lambda > 0$ is defined as $\lambda\mathcal{P} = \{\lambda z \in \mathbb{R}^n \mid Fz \leq g\} = \{z \in \mathbb{R}^n \mid Fz \leq \lambda g\}$. Given a matrix F and a vector g, the ith row of F is denoted $F^{(i)T}$ and the ith component of g is $g^{(i)}$. The interior of any set $\mathcal{Q}$ is denoted **int** $\mathcal{Q}$. The value of a variable y at a time interval t is denoted y_t. A state or control trajectory of length τ is written as the vector $\mathbf{x} = \left[x_1^T, \dots, x_\tau^T\right]^T \in \mathbb{R}^{n\tau}$ or $\mathbf{u} = \left[u_0^T, \dots, u_{\tau-1}^T\right]^T \in \mathbb{R}^{m\tau}$. The column vector of all ones is $\mathbf{1}$. The symbol $\circ$ denotes function composition.

II. Problem Formulation

In this paper, we consider the problem of regulating discrete-time dynamical systems of the form

$$x_{t+1} = Ax_t + Bu_t + d_t \tag{1}$$

where $x_t \in \mathbb{R}^n$ is the system state at time t, $u_t \in \mathbb{R}^m$ is the control input, and $d_t \in \mathbb{R}^n$ is an uncertain input that captures exogenous disturbances and/or linearization error (if the true system dynamics are nonlinear) [18]. We assume the pair (A, B) is stabilizable. The input constraints (actuation limits) are $\mathcal{U} = \{u \in \mathbb{R}^m \mid F_u u \leq g_u\}$ while the state constraints arising from safety-critical engineering considerations are $\mathcal{X} = \{x \in \mathbb{R}^n \mid F_x x \leq g_x\}$.

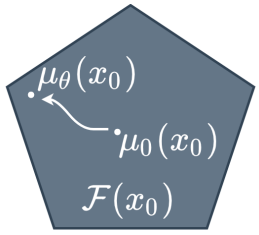

Fig. 1. Illustration of the interior point approach to learning-based MPC. The set $\mathcal{F}(x_0)$ represents the MPC feasible set, while $\mu_0(x_0)$ and $\mu_\theta(x_0)$ are control input sequences representing solutions to the Phase I and Phase II problems, respectively. The neural network μ_θ moves the Phase I solution to a more optimal solution.

We consider the problem of operating the system (1) using finite-horizon model predictive control. The goal is to choose, given initial condition $x_0 \in \mathcal{X}$, a sequence of inputs $\mathbf{u}$ of length τ that minimizes the cost of operating the system while respecting the operational constraints.

However, since the disturbances d_t are unknown ahead of time, the designer must carefully consider how to achieve both optimality and constraint satisfaction. Robust MPC literature contains many ways to handle the presence of disturbances in both the cost and constraints [19]. For example, the *certainty-equivalent* approach [5] considers only the nominal system trajectory, while the *min-max* approach [9] considers the worst-case disturbance. Interpolating between these two extremes, the *tube-based* approach [20] considers the cost of a nominal trajectory while guaranteeing that the true trajectory satisfies constraints. A *stochastic* point of view in [21] considers the disturbance as a random variable and minimizes the expected cost while providing probabilistic guarantees for constraint satisfaction.

In most robust MPC formulations, the set of possible disturbances is modeled as either a finite set, a bounded set, or a probability distribution [22]. In this paper, we assume the disturbances lie in a closed and bounded set $\mathcal{D} := \{d \in \mathbb{R}^n \mid F_d d \leq g_d\}$. In order to ensure constraint satisfaction, we operate the system within a *robust control invariant set* (RCI) $\mathcal{S} \subseteq \mathcal{X}$, defined as a set of initial conditions for which there exists a feedback policy in $\mathcal{U}$ keeping all system trajectories in $\mathcal{S}$, under any disturbance sequence in $\mathcal{D}$ [23]. In our simulations, we used approximately-maximal RCIs computed with the semidefinite program from [24].

With $\mathcal{S} := \{x \in \mathbb{R}^n \mid F_s x \leq g_s\}$, we define the *target set* $\mathcal{T}$ as $\{x \in \mathbb{R}^n \mid x + d \in \mathcal{S},\ \forall\ d \in \mathcal{D}\} = \{x \in \mathbb{R}^n \mid F_s x \leq \tilde{g}_s\}$ where for each row i, $\tilde{g}_s^{(i)} = g_s^{(i)} - \max_{d\in\mathcal{D}} F_s^{(i)T} d$ [23]. Any policy that maps $\mathcal{S}$ to $\mathcal{T}$ under the nominal dynamics will map $\mathcal{S}$ to itself under the true dynamics, rendering $\mathcal{S}$ robustly invariant. By constraining the nominal state to the target set, robust constraint satisfaction is guaranteed for the first time step. Since $\mathcal{S}$ is RCI, this is sufficient for keeping closed-loop trajectories inside $\mathcal{S}$. Under this formulation, the MPC problem is posed as follows, given initial state x_0:

$$\min_{\mathbf{u}} \sum_{k=0}^{\tau-1} l(x_k, u_k) + l_F(x_\tau) \tag{2a}$$

$$\text{subject to } \forall\ k\text{: } x_{k+1} = Ax_k + Bu_k \tag{2b}$$

$$x_{k+1} \in \mathcal{T} \tag{2c}$$

$$u_k \in \mathcal{U} \tag{2d}$$

where l and l_F are stage and terminal costs that are differentiable but possibly nonlinear or even non-convex. Although (2) differs from the standard tube-based approach, the techniques introduced in this paper can be applied to a variety of MPC formulations.

In this paper, we seek to derive a safe feedback policy $\pi_\theta : \mathbb{R}^n \to \mathbb{R}^m$ that approximates the explicit solution to (2) by first approximating the optimal control sequence with a function $\mu_\theta : \mathbb{R}^n \to \mathbb{R}^{m\tau}$ and then implementing the first action of the sequence in the closed loop. In practice, any MPC policy implemented in closed loop must be stabilizing and recursively feasible. Recursive feasibility is the property that closed-loop trajectories generated by the MPC controller will not lead to states in which the MPC problem is infeasible. This property is guaranteed when $\mathcal{S}$ is RCI [23]. If recursive feasibility is not guaranteed, then a backup controller must be developed or a control sequence that is feasible for the most immediate time steps can be used. There is suggestion in the literature that the latter approach performs quite well in practice [25], but the theoretical aspects remain open. In terms of stability, recursive feasibility guarantees that trajectories will remain within a bounded set. Since this work focuses on constraint satisfaction, we do not consider stricter notions of stability.

III. Phase I: Finding a Feasible Point

The feasible set of (2) is a polytope $\mathcal{F}(x_0) \subseteq \mathbb{R}^{m\tau}$, defined by the following inequalities in $\mathbf{u}$:

$$H_s(M_0 x_0 + M_u \mathbf{u}) \leq \tilde{h}_s, \tag{3a}$$
$$H_u \mathbf{u} \leq h_u \tag{3b}$$

where $H_s, H_u, M_0, M_u, \tilde{h}_s$, and h_u are block matrices and vectors derived from the system dynamics and constraints. In this paper, we assume that $\mathcal{F}(x_0)$ has nonempty interior for all $x_0 \in \mathcal{S}$. Since the state constraints $\mathcal{S}$ form an RCI, $\mathcal{F}(x_0)$ is already guaranteed to be nonempty, and the assumption of nonempty interior is only marginally more restrictive.

The gauge map technique introduced in [17] provides a way to constrain the outputs of a neural network $\mu_\theta : \mathbb{R}^n \to \mathbb{R}^{m\tau}$ to $\mathcal{F}(x_0)$ without a projection or penalty function, but $\mathcal{F}(x_0)$ must contain the origin in its interior. If this is not the case, then we must temporarily "shift" $\mathcal{F}(x_0)$ by subtracting any one of its interior points. In this section, we discuss several ways to reduce the complexity of finding an interior point.

We begin by considering the feasibility problem for the one-step safe action set defined as $\mathcal{V}(x_0) = \{u \in \mathbb{R}^m \mid u \in \mathcal{U}, Ax_0 + Bu \in \mathcal{T}\}$, which is guaranteed to have an interior point by the assumption on $\mathcal{F}(x_0)$. One way to find an interior point of $\mathcal{V}(x_0)$ is to minimize the maximum constraint violation:

$$\min_{u,s} \; s \tag{4a}$$
$$\text{subject to: } F_s(Ax_0 + Bu) \leq \tilde{g}_s + s\mathbf{1} \tag{4b}$$
$$F_u u \leq g_u + s\mathbf{1} \tag{4c}$$

which has an optimal cost $s^* \leq 0$ if $\mathcal{V}(x_0)$ is nonempty, and $s^* < 0$ if $\mathcal{V}(x_0)$ has nonempty interior [16]. To avoid solving a linear program online during closed-loop implementation, the solution to (4) can be stored as a piecewise affine (PWA) function $\pi_0(x_0) : \mathbb{R}^n \to \mathbb{R}^m$ [7]. Although solutions to multiparametric LPs can be demanding on computer memory, we take advantage of the fact that feasibility problems have low accuracy requirements: any suboptimal solution to (4) that achieves a cost $s < 0$ for all $x_0 \in \mathcal{S}$ is acceptable.

Definition 1: A function $\pi_0 : \mathbb{R}^n \to \mathbb{R}^m$ is said to solve (4) if, for all $x_0 \in \mathcal{S}$, the optimal cost of (4) is negative when the decision variable u is fixed at $\pi_0(x_0)$.

Existing techniques for approximate multiparametric linear programming [26], especially those that generate continuous solutions [27], can be used to reduce the memory requirements of offline solutions to (4).

To show just how far one can go with reducing complexity, we will construct an affine (rather than PWA) function that solves (4), for the system studied in Section V. Let $\pi_0(x_0) = Wx_0 + w$. If $W \in \mathbb{R}^{m\times n}$ and $w \in \mathbb{R}^m$ satisfy

$$F_x(Ax_0 + B(Wx_0 + w)) < \tilde{g}_x \tag{5a}$$
$$F_u(Wx_0 + w) < g_u \tag{5b}$$

for all $x_0 \in \mathcal{S}$, then $\pi_0(x_0) = Wx_0 + w$ solves (4). The following optimization problem can be solved to find W and w or certify that none exists. Let $\mathcal{Y}(s) = \{x_0 \in \mathbb{R}^n \mid F_s(Ax_0 + B(Wx_0 + w)) \leq \tilde{g}_s + s\mathbf{1}, F_u(Wx_0 + w) \leq g_u + s\mathbf{1}\}$. If the optimal cost of

$$\min_{W,w,s} \; s \text{ subject to } \mathcal{S} \subseteq \mathcal{Y}(s) \tag{6}$$

is negative, then (5) holds for all $x_0 \in \mathcal{S}$, thus π_0 solves (4). This happens to be the case for the example in Section V, taken from [6]. The constraint in (6) is a polytope containment constraint in halfspace representation, thus (6) can be solved as a linear program [28].

Now consider the feasibility problem for $\mathcal{F}(x_0)$, which is obtained by replacing (4b) and (4c) with (3a) and (3b), and changing the optimization variable from $u \in \mathbb{R}^m$ to $\mathbf{u} \in \mathbb{R}^{m\tau}$. One would naturally expect the complexity of the PWA solution to this feasibility problem to increase rapidly with the time horizon τ, as more decision variables and constraints are added. However, the next proposition shows that the cardinality of the stored partition can be made constant in τ.

Proposition 1 (Phase I solution): If π_0 solves (4), then the vector $\mu_0(x_0) := \left[\pi_0(x_0)^T, \ldots, \pi_0(x_{\tau-1})^T\right]^T$, where $x_{k+1} = Ax_k + B\pi_0(x_k)$, is an interior point of $\mathcal{F}(x_0)$ for any $x_0 \in \mathcal{S}$.

Proof: If π_0 solves (4), then $\pi_0(x) \in \textbf{int } \mathcal{V}(x)$ for all $x \in \mathcal{S}$. Applying the definition of $\mathcal{V}$ in an inductive argument, it is straightforward to show that the state trajectory associated with $\mu_0(x_0)$ is entirely contained in $\mathcal{S}$. Fix any such trajectory $\{x_1, \ldots, x_\tau\} \subset \mathcal{S}$ originating from $x_0 \in \mathcal{S}$ under policy π_0. For any $k \in \{1, \ldots, \tau\}$, $x_k \in \mathcal{S}$ implies $\pi_0(x_k) \in \textbf{int } \mathcal{V}(x_k)$, which implies $\pi_0(x_k) \in \textbf{int } \mathcal{U}$ and $Ax_k + B\pi_0(x_k) \in \textbf{int } \mathcal{T}$. Since this holds for all k, the constraints defining $\mathcal{F}(x_0)$ hold strictly at $\mu_0(x_0)$. ■

In our simulations on the example from [6], (6) was feasible with negative optimal cost, meaning that a polyhedral partition of the state space was not needed (see Section V). This indicates that the minimum number of regions in a polyhedral state space partition associated with a PWA

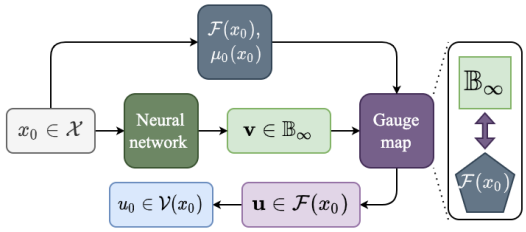

Fig. 2. The proposed control policy uses a neural network combined with the Phase I solution and a *gauge map* to constrain the decision $\mathbf{u}$ to the MPC feasible set $\mathcal{F}(x_0)$. The first action from the sequence $\mathbf{u}$ is extracted and implemented. On the right, the action of the gauge map is illustrated.

solution to (4) is in general very small relative to the number of regions in an explicit solution to (2).

IV. Phase II: Optimizing Performance

In this section, we construct a class of policies from $x_0 \in \mathcal{S}$ to $\mathcal{F}(x_0)$, that can be trained using standard machine learning packages. Although it is difficult to constrain the output of a neural network to an arbitrary polytope such as $\mathcal{F}(x_0)$, it is easy to constrain the output to the hypercube $\mathbb{B}_\infty$ by applying a clamping function elementwise in the output layer. We apply a mapping between polytopes that is closed-form, differentiable, and bijective. This mapping establishes an equivalence between $\mathbb{B}_\infty$ and $\mathcal{F}(x_0)$, allowing one to constrain the outputs of the policy to $\mathcal{F}(x_0)$. The mapping from $\mathbb{B}_\infty$ to $\mathcal{F}(x_0)$ is called the *gauge map*. The concept is illustrated in Figure 2.

We begin constructing the gauge map by introducing some preliminary concepts. A *C-set* is a convex, compact set that contains the origin as an interior point. The *gauge function* with respect to C-set $\mathcal{P} \subset \mathbb{R}^n$, denoted $\gamma_\mathcal{P} : \mathbb{R}^n \to \mathbb{R}_+$, is the function whose sublevel sets are scaled versions of $\mathcal{P}$. Specifically, the gauge of a vector v with respect to $\mathcal{P}$ is given by $\gamma_\mathcal{P}(v) = \inf\{\lambda \geq 0 \mid v \in \lambda\mathcal{P}\}$. If $\mathcal{P}$ is a polytopic C-set given by $\{v \in \mathbb{R}^k \mid Fv \leq g\}$, then $\gamma_\mathcal{P}$ is the pointwise maximum over a finite set of affine functions [17]:

$$\gamma_\mathcal{P}(v) = \max_i \frac{F^{(i)T} v}{g^{(i)}}. \tag{7}$$

Given two C-sets $\mathcal{P}$ and $\mathcal{Q}$, the *gauge map* $G : \mathcal{P} \to \mathcal{Q}$ is

$$G(v \mid \mathcal{P}, \mathcal{Q}) = \frac{\gamma_\mathcal{P}(v)}{\gamma_\mathcal{Q}(v)} \cdot v. \tag{8}$$

This function maps level sets of $\gamma_\mathcal{P}$ to level sets of $\gamma_\mathcal{Q}$.

Proposition 2: Given two polytopic C-sets $\mathcal{P}$ and $\mathcal{Q}$, the gauge map $G : \mathcal{P} \to \mathcal{Q}$ is subdifferentiable and bijective. Further, given a function π_0 from Proposition 1, the set $\tilde{\mathcal{F}}(x) := [\mathcal{F}(x) - \pi_0(x)]$ is a C-set for all $x \in \mathcal{S}$.

Proof: The properties of subdifferentiability and bijectivity come from [17]. For the C-set property, fix $x \in \mathcal{S}$. Since $\mathcal{S}, \mathcal{U}$, and $\mathcal{D}$ are convex and compact, so is $\mathcal{F}(x)$. Since $\mu_0(x)$ is an interior point of $\mathcal{F}(x)$, the set $\tilde{\mathcal{F}}(x)$ contains the origin as an interior point and is therefore a C-set. ■

We now use the gauge map in conjunction with the Phase I solution to construct a neural network whose output is confined to $\mathcal{F}(x_0)$. Let $\psi_\theta : \mathcal{S} \to \mathbb{B}_\infty$ be a neural network parameterized by θ. A safe policy is constructed by composing the gauge map $G : \mathbb{B}_\infty \to \tilde{\mathcal{F}}(x_0)$ with ψ_θ, then adding $\mu_0(x_0)$ to map the solution into $\mathcal{F}(x_0)$:

$$\mu_\theta(x_0) = G(\cdot \mid \mathbb{B}_\infty, \tilde{\mathcal{F}}(x_0)) \circ \psi_\theta(x_0) + \mu_0(x_0). \tag{9}$$

Computing the gauge map online simply requires evaluating $H_s M_0 x_0$ from (3a) as well as the operations in (7).

The function μ_θ has several important properties for approximating the optimal solution to (2). First, it leverages the universal function approximation properties of neural networks [29] along with the bijectivity of the gauge map (Proposition 2) to explore all interior points of $\mathcal{F}(x_0)$. This is an advantage over projection-based methods [11] which may be biased towards the boundary of $\mathcal{F}(x_0)$ when the optimal solution may lie on the interior. Second, μ_θ is evaluated in closed form, and its outputs are constrained to $\mathcal{F}(x_0)$ without the use of an optimization layer [14] that may have high computational overhead. Finally, the subdifferentiability of the gauge map (Proposition 2) enables selection of parameter θ using standard automatic differentiation techniques.

Optimizing the parameter θ

Similar to the approach taken in [10], we optimize θ by sampling $x \in \mathcal{S}$ and applying stochastic gradient descent. At each iteration, a new batch of initial conditions $\{x_0^j\}_{j=1}^M$ is sampled from $\mathcal{S}$ and the loss is computed as

$$J(\theta) = \frac{1}{M} \sum_{j=1}^{M} \sum_{k=0}^{\tau-1} l(x_k^j, u_k^j) + l_F(x_\tau^j) \tag{10}$$

with the control sequences $\mathbf{u}^j$ given by $\mu_\theta(x_0^j)$ and state trajectories $\mathbf{x}^j$ generated according to the nominal dynamics. The parameters θ are updated in the direction of $\nabla_\theta J$, which is easily computed using automatic differentiation [30].

V. Simulations

A. Test systems

We simulate the proposed policy using a modified example from [6] with $n = 3$, $m = 2$, and $\tau = 5$. The system matrices, constraints, costs, and Phase I solution (found using (6)) are given below:

$$A = \begin{bmatrix} -.5 & .3 & -1 \\ .2 & -.5 & .6 \\ 1 & .6 & -.6 \end{bmatrix}, \; B = \begin{bmatrix} -.601 & -.890 \\ .955 & -.715 \\ .246 & -.184 \end{bmatrix}, \tag{11}$$

$$\|x\|_\infty \leq 5, \; \|u\|_\infty \leq 1, \; \|d\|_\infty \leq 0.1, \tag{12}$$

$$l(x,u) = \|x\|_2^2 + c_1\|u\|_2^2, \; l_F(x) = c_2\|x\|_2^2 \tag{13}$$

$$W = \begin{bmatrix} 0.116 & 0.210 & -0.370 \\ -0.320 & -0.104 & -0.122 \end{bmatrix}, w = \begin{bmatrix} -0.157 \\ -0.0533 \end{bmatrix}$$

where c_1 and c_2 are positive constants. Although quadratic costs are used in the simulations, the proposed method can work with any differentiable cost.

We evaluate the performance of a given policy in both open- and closed-loop experiments. In the open-loop experiments, we evaluate the MPC cost (2a) and compare it to the

optimal cost. The fraction suboptimality is

$$\delta = \frac{c_{nn} - c_{mpc}}{c_{mpc}} \tag{14}$$

where c_{nn} is the average cost (2a) incurred by the control sequence μ_θ on a validation set $\{x_0^j\}_{j=1}^{N_{val}} \subset \mathcal{S}$ and c_{mpc} is the optimal cost.

In the closed-loop experiments, we evaluate the performance of a policy $\pi_\theta(x_t) : \mathbb{R}^n \to \mathbb{R}^m, t \geq 0$ which is derived from $\mu_\theta(x_t)$ by taking the first action in the sequence. We simulate (1) for $T \gg \tau$ time steps. The trajectory cost in the closed-loop experiments is computed as $\sum_{t=0}^{T-1} l(x_t, u_t) + l_F(x_T)$ and the disturbance is modeled as an autoregressive sequence [31], $d_{t+1} = \alpha d_t + (1-\alpha)\hat{d}$ where $\alpha \in (0, 1)$ and $\hat{d}$ is drawn uniformly over $\mathcal{D}$.

B. Benchmarks

We compare the proposed method to two of the most common approaches for learning a solution to (2). The first benchmark is a penalty-based approach [32] which enforces the constraints (2c) and (2d) by augmenting the cost (10) with a linear penalty term on constraint violations given by $\beta \cdot \max\{0, F_x x_t - \tilde{g}_x\}$ where the max is evaluated elementwise and $\beta > 0$. Since the penalty-based approach does not encode state constraints in the policy, the policy is constrained to the Cartesian product $\mathcal{U}^\tau = \prod_{k=0}^{\tau-1} \mathcal{U}$ using scaled $\tanh$ functions elementwise.

The second benchmark is a projection-based approach [11] which constrains the policy to the set $\mathcal{F}(x_0)$ by solving a convex quadratic program in the output layer of a neural network [33]. The optimization layer $\mathbf{v} \to \mathbf{u}$ returns

$$\arg\min_{\mathbf{u}} \|\mathbf{v} - \mathbf{u}\|_2^2 \text{ subject to } \mathbf{u} \in \mathcal{F}(x_0).$$

Another class of approaches to learning-based MPC seeks to learn the optimal solution to (2) using regression [12]–[14]. Specifically, data-label pairs (x_0, u_0^*) are generated by sampling x_0 from $\mathcal{S}$, solving (2) for each sample, and extracting u_0^* from the optimal solution $\mathbf{u}^*$. Then, a neural network or other function approximator is trained to learn the relationship between x_0 and u_0^*. Performance and constraint satisfaction are handled e.g. by bounding the approximation error with respect to the MPC oracle. We do not compare against this type of approach since it requires a large number of trained samples, making it difficult to compare with our and the other unsupervised examples.

C. Neural network design

The neural networks were designed with n inputs, $m\tau$ outputs, and two hidden layers with rectified linear unit (ReLU) activation functions. The width of the networks was chosen during hyperparameter tuning. In particular, we performed 30 iterations of random search over the width of the network (number of neurons per hidden layer) $\in \{64, \ldots, 1024\}$, the batch size (number of initial conditions, M) $\in \{100, \ldots, 3000\}$ and the learning rate (LR, the step size for gradient descent) $\in [10^{-5}, 10^{-3}]$. For each set of hyperparameters under consideration, we computed the validation score using (14) with $N_{val} = 100$. The hyperparameters after tuning are reported in Table I.

TABLE I

HYPERPARAMETERS FOR THE THREE NEURAL NETWORKS.

Type	Width	LR	M
Gauge	859	4.7×10^{-4}	1655
Penalty	318	8.7×10^{-4}	133
Projection	956	9.0×10^{-5}	813

TABLE II

OPEN-LOOP TEST RESULTS.

Type	δ (14)	Solve time (sec)
Gauge	0.007	.0015
Projection	0.010	.024

D. Simulation results

Here we compare our proposed approach (Gauge NN), the penalty-based approach (Penalty NN), the projection-based approach (Projection NN) and the "ground truth" obtained by solving (2) online in `cvxpy`. The results of the open-loop experiments are shown in Table II, with performance computed relative to the optimal MPC solution using (14) with $N_{val} = 100$ trials. The proposed Gauge NN achieves lower cost compared to the projection-based method, and has a much lower computational complexity (solve time is only 6% of projection). Table II only compares the NNs with safety guarantees because constraint violations are not accounted for in (14).

Figure 3 shows the training curves for each type of network. The lower training cost achieved by the Gauge NN illustrates that it can be more efficient to explore the interior of the feasible set than the boundary. Since the MPC cost in the simulations is strictly convex, solutions with lower cost are closer to the optimal solution.

Figure 4 compares the policies in terms of computation time and test performance. The box-and-whisker plots indicate the range of performance over 100 test trajectories of length $T = 50$, while the vertical position of each box indicates the average time to compute a control action. Of the policies with safety guarantees (Gauge NN, Projection NN, and online MPC), the Gauge NN achieves Pareto efficiency in terms of average solve time and median trajectory cost.

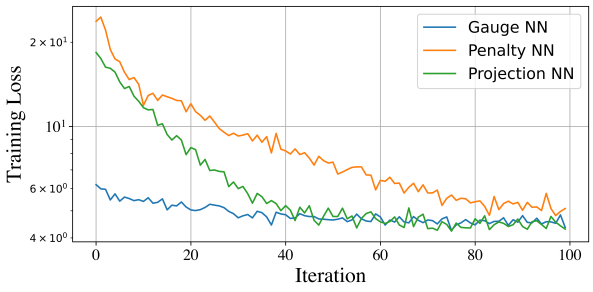

Fig. 3. Training trajectories for the three types of neural netwokrs. Our proposed Gauge-based approach achieves lower cost at a much faster rate.

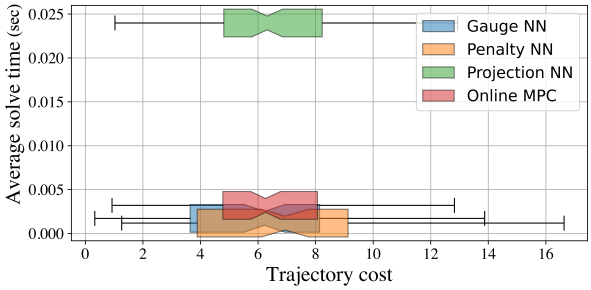

Fig. 4. Solve time vs. trajectory cost for the networks under consideration applied to the 3-state system. The Gauge NN is Pareto-efficient in terms of cost and computation time compared to the other techniques with safety guarantees (Online MPC and Projection NN).

Our intuition behind the high performance of the neural networks is that (2) is a heuristic and the unsupervised learning approach can lead to better closed-loop policies.

VI. Conclusion

In this paper, we provided an efficient way of exploring the interior of the MPC feasible set for learning-based approximate explicit MPC, and demonstrated the performance and computational gains that can be achieved by approaching the problem from the interior. The paradigm relies on a Phase I solution that exploits the structure of the MPC problem and a Phase II solution that features a projection-free feasibility guarantee. The results compare favorably against common approaches that use unsupervised learning, as well as against the oracle itself used in supervised approaches. Future work includes applications to MPC problems with convex but non-polytopic constraint sets, and to distributed settings.

References

[1] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory and Design*. Santa Barbara, CA: Nob Hill, 2 ed., 2019.

[2] Y. Ding, L. Wang, Y. Li, and D. Li, "Model predictive control and its application in agriculture: A review," *Computers and Electronics in Agriculture*, vol. 151, pp. 104–117, 2018.

[3] D. Hrovat, S. Di Cairano, H. E. Tseng, and I. V. Kolmanovsky, "The development of Model Predictive Control in automotive industry: A survey," *Proc. IEEE Int. Conf. on Control Applications*, pp. 295–302, 2012.

[4] A. Ademola-Idowu and B. Zhang, "Frequency Stability Using MPC-Based Inverter Power Control in Low-Inertia Power Systems," *IEEE Trans. Power Syst.*, vol. 36, no. 2, pp. 1628–1637, 2021.

[5] A. Alessio and A. Bemporad, "A Survey on Explicit Model Predictive Control," in *Nonlinear Model Predictive Control* (L. Magni, D. Raimondo, and F. Allgöwer, eds.), Berlin, Heidelberg: Springer, 2009.

[6] M. N. Zeilinger, C. N. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization," *IEEE Trans. Autom. Control*, vol. 56, no. 7, pp. 1524–1534, 2011.

[7] C. N. Jones, M. Barić, and M. Morari, "Multiparametric linear programming with applications to control," *European Journal of Control*, vol. 13, no. 2-3, pp. 152–170, 2007.

[8] T. A. Johansen, "Approximate explicit receding horizon control of constrained nonlinear systems," *Automatica*, vol. 40, no. 2, pp. 293–300, 2004.

[9] A. Grancharova and T. A. Johansen, "Computation, approximation and stability of explicit feedback min-max nonlinear model predictive control," *Automatica*, vol. 45, no. 5, pp. 1134–1143, 2009.

[10] B. M. Åkesson and H. T. Toivonen, "A neural network model predictive controller," *J. Process Control*, vol. 16, no. 9, pp. 937–946, 2006.

[11] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, "Approximating Explicit Model Predictive Control Using Constrained Neural Networks," *Proc. Am. Control Conf.*, vol. 2018-June, pp. 1520–1527, 2018.

[12] T. Parisini and R. Zoppoli, "A Receding-Horizon Regulator for Nonlinear Systems and a Neural Approximation," *Automatica*, vol. 31, no. 10, pp. 1443–1451, 1995.

[13] A. Domahidi, M. N. Zeilinger, M. Morari, and C. N. Jones, "Learning a feasible and stabilizing explicit model predictive control law by robust optimization," in *Proc. IEEE Conf. Decision Control*, pp. 513–519, IEEE, 2011.

[14] E. T. Maddalena, C. G. da Moraes, G. Waltrich, and C. N. Jones, "A neural network architecture to learn explicit MPC controllers from data," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 11362–11367, 2020.

[15] L. Zheng, Y. Shi, L. J. Ratliff, and B. Zhang, "Safe reinforcement learning of control-affine systems with vertex networks," in *Learning for Dynamics and Control*, pp. 336–347, PMLR, 2021.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2009.

[17] D. Tabas and B. Zhang, "Computationally Efficient Safe Reinforcement Learning for Power Systems," *arXiv:2110.10333*, 2021.

[18] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, vol. 15. Philadelphia: Society for Industrial and Applied Mathematics, 1994.

[19] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*, pp. 207–226, London: Springer, 1999.

[20] W. Langson, I. Chryssochoos, S. V. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.

[21] M. Farina, L. Giulioni, and R. Scattolini, "Stochastic linear Model Predictive Control with chance constraints - A review," *J. Process Control*, vol. 44, pp. 53–67, 2016.

[22] M. B. Saltık, L. Özkan, J. H. Ludlage, S. Weiland, and P. M. Van den Hof, "An outlook on robust model predictive control algorithms: Reflections on performance and computational aspects," *J. Process Control*, vol. 61, pp. 77–102, 2018.

[23] F. Blanchini and S. Miani, *Set-theoretic methods in control*. Birkhauser, 2015.

[24] C. Liu and I. M. Jaimoukha, "The computation of full-complexity polytopic robust control invariant sets," in *Proc. 54th IEEE Conf. Decision Control*, pp. 6233–6238, 2015.

[25] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, 2010.

[26] C. Filippi, "An algorithm for approximate multiparametric linear programming," *Journal of Optimization Theory and Applications*, vol. 120, no. 1, pp. 73–95, 2004.

[27] J. Spjøtvold, P. Tøndel, and T. A. Johansen, "A method for obtaining continuous solutions to multiparametric linear programs," *IFAC Proc. Volumes (IFAC-PapersOnline)*, vol. 38, no. 1, pp. 253–258, 2005.

[28] S. Sadraddini and R. Tedrake, "Linear Encodings for Polytope Containment Problems," *Proc. IEEE Conf. Decision Control*, pp. 4367–4372, 2019.

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

[30] A. Baydin, A. A. Radul, B. A. Pearlmutter, and J. M. Siskind, "Automatic Differentiation in Machine Learning: a Survey," *J. Machine Learning Research*, vol. 18, pp. 1–43, 2018.

[31] M. D. Srinath, P. Rajasekaran, and R. Viswanathan, *Introduction to statistical signal processing with applications*. Prentice-Hall, Inc., 1995.

[32] J. Drgona, K. Kis, A. Tuor, D. Vrabie, and M. Klauco, "Differentiable Predictive Control: Deep Learning Alternative to Explicit Model Predictive Control for Unknown Nonlinear Systems," *arXiv:2011.03699*, 2020.

[33] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Zico Kolter, "Differentiable convex optimization layers," *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019.