

# A Sensitivity-based Data Augmentation Framework for Model Predictive Control Policy Approximation

Dinesh Krishnamoorthy *Member, IEEE*

**Abstract**—Approximating model predictive control (MPC) policy using expert-based supervised learning techniques requires labeled training data sets sampled from the MPC policy. This is typically obtained by sampling the feasible state-space and evaluating the control law by solving the numerical optimization problem offline for each sample. Although the resulting approximate policy can be cheaply evaluated online, generating large training samples to learn the MPC policy can be time consuming and prohibitively expensive. This is one of the fundamental bottlenecks that limit the design and implementation of MPC policy approximation. This technical note aims to address this challenge, and proposes a novel sensitivity-based data augmentation scheme for direct policy approximation. The proposed approach is based on exploiting the parametric sensitivities to cheaply generate additional training samples in the neighborhood of the existing samples.

## I. INTRODUCTION

Model predictive control (MPC) is a popular control strategy for constrained multivariable systems that is based on repeatedly solving a receding horizon optimal control problem at each sampling time of the controller. As the range of MPC application extends beyond the traditional process industries, additional challenges such as computational effort and memory footprint need to be addressed. One approach to eliminate the need for solving optimization problems online, is to pre-compute the MPC policy  $u^* = \pi(x)$  as a function of the states  $x$ . This idea was first proposed under the context of *explicit MPC* for constrained linear quadratic systems where the MPC feedback law is expressed as a piecewise-affine function defined on polytopes [1], [2]. However, this can quickly become computationally intractable for large systems, since the number of polytopic regions grow exponentially with the number of decision variables and constraints. The extension to nonlinear systems (with economic objective terms) is also not straightforward.

An alternative approach is to use some parametric function approximator, such as artificial neural networks (ANN) to approximate the MPC policy. Although this idea dates back to the mid 90s [3], the use of parametric functions to approximate the MPC policy remained more or less dormant until very recently. Motivated by the recent developments and promises of deep learning techniques, there has been an unprecedented surge of interest in the past couple of years in approximating the MPC policy using deep neural networks. This interest has resulted in a number of research works from several research groups published just in the past couple of years. See for example [4]–[11] to name a few.

The underlying framework adopted in these works is as follows. The feasible state-space is sampled offline to generate a finite number of  $N_s$  discrete states  $\{x_i\}_{i=1}^{N_s}$ . The NMPC problem is solved offline for each discrete state as the initial condition to obtain the corresponding optimal control law  $u_i^* = \pi_{\text{mpc}}(x_i)$  for all  $i = 1, \dots, N_s$ . The resulting MPC policy  $\pi_{\text{mpc}}(\cdot)$  is approximated using any suitable parametric function approximator with  $\{(x_i, u_i^*)\}_{i=1}^{N_s}$  as the labeled

training data set, such that the trained policy  $\pi_{\text{approx}}(\cdot)$  can be used online to cheaply evaluate the optimal control input. This approach is also studied more broadly under the context of *direct policy approximation*, and is also known as expert-based supervised learning [12], behavioral cloning [13], or imitation learning [14], [15], where in our case the MPC policy is the “expert” that we would like to imitate as faithfully as possible.

One of the main bottlenecks of this approach is that, generating the training data set (given by the “expert”) can be time consuming and prohibitively expensive. The availability of large training data set covering the entire feasible state space is a key stipulation in using deep learning techniques and has a major impact on the accuracy of the approximate policy. This implies that the sample size  $N_s$  must be sufficiently large, covering the entire feasible state space. In the case of MPC policy approximation, one then typically has to solve a large number of nonlinear programming (NLP) problems offline in order to generate adequate training samples. This challenge is only amplified for higher dimensional systems, since the number of samples  $N_s$  required to adequately cover the feasible state-space increases exponentially with the number of states. For example, the authors in [9] reported a computation time of roughly 500 hours on a Quad-Core PC<sup>1</sup> to learn the approximate MPC control law for the case study considered in their work. Other works also report the need for a large training data set to adequately approximate the MPC policy.

In the field of machine learning and deep neural networks, the problem of insufficient training data samples is typically addressed using a process known as “data augmentation”, which is *a strategy to artificially increase the number of training samples using computationally inexpensive transformations of the existing samples* [16]–[18]. This has been extensively studied in the context of deep learning for image classification problems, where geometric transformations (such as translation, rotation, cropping etc.) and photometric transformations (such as color, saturation, contrast, brightness etc.) are often used to augment the existing data set with artificially generated training samples. Unfortunately, such data augmentation techniques are not relevant when the training data set is obtained by sampling some expert policy, and hence are not applicable in the context of control policy approximation.

This technical note aims to address the key issue of generating the training data samples by exploiting the NLP sensitivities to generate multiple training samples using the solution of a single optimization problem solved offline. That is, the MPC problem solved offline can be seen as a parametric optimization problem parameterized with respect to the initial state  $x_i$ . The NLP sensitivity then tells us how the optimal input  $u_i^*$  changes for perturbations  $\Delta x$  in the neighborhood of  $x_i$ . Therefore, using the solution to one parametric optimization problem solved for  $x_i$ , we can cheaply generate multiple training data samples for other state realizations in the neighborhood of  $x_i$  using the NLP sensitivity (also known as tangential predictor). This only requires computing the solution to a system of linear equations, which is much cheaper to evaluate than solving a nonlinear programming problem. To this end, the aim of this technical note is not to

<sup>\*</sup>This work was supported by the Research Council of Norway, under the IKTPLUSS program (Project number 299585)

Dinesh Krishnamoorthy is currently with the Harvard John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, 02138. (email: dkrishnamoorthy@seas.harvard.edu)

<sup>1</sup>without parallelization of the sampling and validation

present a new MPC approximation algorithm, but rather address the pivotal issue of generating training data samples, that would facilitate efficient design and implementation of the approximate MPC framework.

**Main contribution:** The main contribution of this technical note is a sensitivity-based data augmentation framework to efficiently and cheaply generate training data samples that can be used to approximate the MPC policy. More broadly, the proposed scheme is not just restricted to data augmentation for approximating an MPC policy, but can also be used in cases where the training data set is generated by solving an NLP problem e.g. in inverse optimal control (where the goal is to impute the objective from data sampled from an optimal policy [19], [20]), approximate moving horizon estimation [21], steady-state real-time optimization [22] etc. Thus, a wider contribution of this technical note is a novel data augmentation framework for approximate optimal control problems, where the training data comprises of optimal state-action pairs, generated by sampling some policy given by a nonlinear programming problem.

The remainder of the paper is organized as follows. Section II formulates the problem and recalls the MPC policy approximation framework. The sensitivity-based data augmentation technique to efficiently generate the training samples is presented in Section III, where we also provide an upper bound on the approximation error stemming from augmenting the data set with inexact samples. The proposed approach is illustrated using two different examples in Section IV before concluding the paper in Section V.

## II. PRELIMINARIES

### A. Problem Formulation

Consider a discrete-time nonlinear system,

$$x(t+1) = f(x(t), u(t)) \quad (1)$$

where  $x(t) \in \mathbb{R}^{n_x}$  and  $u(t) \in \mathbb{R}^{n_u}$  are the states and control inputs at time  $t$ , respectively. The mapping  $f: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$  denotes the discrete time plant model. The MPC problem  $\mathcal{P}(x(t))$  is formulated as

$$V_N(x(t)) = \min_{x(\cdot|t), u(\cdot|t)} \sum_{k=0}^{N-1} \ell(x(k|t), u(k|t)) + \ell_f(x(N|t)) \quad (2a)$$

$$\text{s.t. } x(k+1|t) = f(x(k|t), u(k|t)) \quad \forall k \in \mathbb{I}_{0:N-1} \quad (2b)$$

$$x(k|t) \in \mathcal{X}, \quad u(k|t) \in \mathcal{U} \quad \forall k \in \mathbb{I}_{0:N-1} \quad (2c)$$

$$x(N|t) \in \mathcal{X}_f \quad (2d)$$

$$x(0|t) = x(t) \quad (2e)$$

where  $\ell: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}$  denotes the stage cost, which may be either a tracking or economic objective,  $\ell_f: \mathbb{R}^{n_x} \rightarrow \mathbb{R}$  denotes the terminal cost,  $N$  is the length of the prediction horizon, (2c) denotes the state and input constraints, (2d) denotes the terminal constraint, and (2e) denotes the initial condition constraint. In the traditional MPC paradigm, the optimization problem  $\mathcal{P}(x(t))$  is solved at each sample time  $t$  using  $x(t)$  as the state feedback, and the optimal input  $u^*(t) = u^*(0|t)$  is injected into the plant in a receding horizon fashion. This implicitly leads to the control policy  $\pi_{\text{mpc}}: \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_u}$ .

$$u^*(t) = \pi_{\text{mpc}}(x(t)) \quad (3)$$

### B. MPC policy approximation

This subsection recalls the underlying idea of the MPC policy approximation framework common to works such as [3], [5], [8] and [9]. To approximate the MPC control law (3), the feasible state space  $\mathcal{X}_{feas}$  is sampled to generate  $N_s$  randomly chosen initial

**Algorithm 1** Generating training samples and approximating the MPC policy.

**Input:**  $\mathcal{P}(x)$ ,  $\mathcal{X}_{feas}$ ,  $\mathcal{D} = \emptyset$

---

```

1: for  $i = 1, \dots, N_s$  do
2:   Sample  $x_i \in \mathcal{X}_{feas}$ 
3:    $u_i^* \leftarrow \text{Solve } \mathcal{P}(x_i)$ 
4:    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_i, u_i^*)\}$ 
5: end for
6:  $\theta_0 \leftarrow \arg \min_{\theta} \frac{1}{N_s} \sum_{i=1}^{N_s} \|\pi_{\text{approx}}(x_i; \theta) - u_i^*\|^2$ 

```

---

**Output:**  $\pi_{\text{approx}}(x; \theta_0)$

---

states  $\{x_i\}_{i=1}^{N_s}$ . For each initial state  $x_i$ , the MPC problem  $\mathcal{P}(x_i)$  is solved to obtain the corresponding optimal input  $u_i^* = \pi_{\text{mpc}}(x_i)$ . Using the data samples  $\mathcal{D} := \{(x_i, u_i^*)\}_{i=1}^{N_s}$ , any desirable parametric function  $\pi_{\text{approx}}(x; \theta)$  parameterized by the parameters  $\theta$  is trained that minimizes the mean squared error

$$\theta_0 = \arg \min_{\theta} \frac{1}{N_s} \sum_{i=1}^{N_s} \|\pi_{\text{approx}}(x_i; \theta) - u_i^*\|^2 \quad (4)$$

This is summarized in Algorithm 1. Once the parametric function is trained, the approximate control policy  $\pi_{\text{approx}}(x; \theta_0)$  can be used online to cheaply evaluate the optimal control input. Although deep neural networks have become a popular choice for MPC policy approximation, several other function approximators have also been used in the literature. As such, the method proposed in the following section will not be limited to any one class of parametric functions.

## III. PROPOSED METHOD

As mentioned in the previous section, generating the training data requires solving  $N_s$  numerical optimization problems offline, which can be time consuming and computationally expensive. This section leverages the NLP sensitivity to cheaply generate training data samples that can be used to approximate the MPC policy. To keep the notation light, we rewrite the MPC problem (2) into a standard parametric NLP problem of the form,

$$V_N(p) = \min_{\mathbf{w}} J(\mathbf{w}, p) \quad (5a)$$

$$\text{s.t. } c(\mathbf{w}, p) = 0 \quad (5b)$$

$$g(\mathbf{w}, p) \leq 0 \quad (5c)$$

where  $p = x(0|t) = x(t)$  is the initial state, the decision variables  $\mathbf{w} := [u(0|t), \dots, u(N-1|t), x(1|t), \dots, x(N|t)]^T$  the cost (2a) is denoted by (5a), the system equations (2b) are denoted by (5b), the state and input constraints (2c), and the terminal constraint (2d) are collectively denoted by (5c). Since the focus is on solving the MPC problem offline, we drop the time dependency of the initial state, and simply denote the initial condition as  $x$  instead of  $x(t)$ .

The Lagrangian of (5) is given by

$$\mathcal{L}(\mathbf{w}, p, \lambda, \mu) := J(\mathbf{w}, p) + \lambda^T c(\mathbf{w}, p) + \mu^T g(\mathbf{w}, p)$$

where  $\lambda$  and  $\mu$  are the Lagrangian multipliers of (5b) and (5c) respectively, and the KKT condition for this problem is given by,

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, p, \lambda, \mu) = 0 \quad (6a)$$

$$c(\mathbf{w}, p) = 0 \quad (6b)$$

$$g(\mathbf{w}, p) \leq 0 \quad (6c)$$

$$\mu_j g_j(\mathbf{w}, p) = 0, \quad \mu_j \geq 0 \quad \forall j \quad (6d)$$

**Algorithm 2** Generating training samples using sensitivity-based data augmentation and approximating the MPC policy.

**Input:**  $\mathcal{P}(x)$ ,  $\mathcal{X}_{feas}$ ,  $\mathcal{D} = \emptyset$

---

```

1: for  $i = 1, \dots, N_s$  do
2:   Sample  $x_i \in \mathcal{X}_{feas}$ 
3:    $\mathbf{s}^*(x_i) \leftarrow \text{Solve } \mathcal{P}(x_i)$ 
4:   Extract  $u_i^*$  from the primal-dual solution vector  $\mathbf{s}^*(x_i)$ 
5:    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_i, u_i^*)\}$ 
6:   for  $j = 1, \dots, N_p$  do
7:     Sample  $\Delta x_j \in \Delta \mathcal{X}_i$  in the neighborhood of  $x_i$ 
8:      $\hat{\mathbf{s}}^*(x_i + \Delta x_j) = \mathbf{s}^*(x_i) - \mathcal{M}^{-1} \mathcal{N} \Delta x_j$ 
9:     Extract  $\hat{u}_j^*$  from the solution vector  $\hat{\mathbf{s}}^*(x_i + \Delta x_j)$ 
10:    if  $g_{\mathbb{A}}(\mathbf{w}^*(x_i)) = g_{\mathbb{A}}(\hat{\mathbf{w}}(x_i + \Delta x_j))$  then
11:       $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_i + \Delta x_j, \hat{u}_j^*)\}$ 
12:    end if
13:  end for
14: end for
15:  $\theta_2 \leftarrow \arg \min_{\theta} \frac{1}{N_s N_p} \sum_{i=1}^{N_s N_p} \|\pi_{\text{approx}}(x_i; \theta) - u_i^*\|^2$ 

```

---

**Output:**  $\pi_{\text{approx}}(x; \theta_2)$

---

Any point  $\mathbf{s}^*(p) := [\mathbf{w}^*, \lambda^*, \mu^*]^\top$  that satisfies the KKT conditions (6) for a given initial condition  $p$  is known as a KKT point for  $p$ . We define the set of active inequality constraints  $g_{\mathbb{A}}(\mathbf{w}, p) \subseteq g(\mathbf{w}, p)$  such that  $g_{\mathbb{A}}(\mathbf{w}, p) = 0$ , and strict complementarity is said to hold if the corresponding Lagrange multipliers  $\mu_{\mathbb{A}} > 0$  (i.e. no weakly active constraint). This set of KKT conditions can be represented compactly as

$$\varphi(\mathbf{s}(p), p) = \begin{bmatrix} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{s}, p) \\ c(\mathbf{w}, p) \\ g_{\mathbb{A}}(\mathbf{w}, p) \end{bmatrix} = 0$$

*Theorem 1* ([23]): Let  $J(\cdot, \cdot)$ ,  $c(\cdot, \cdot)$  and  $g(\cdot, \cdot)$  of the parametric NLP problem  $\mathcal{P}(p)$  be twice continuously differentiable in a neighborhood of the KKT point  $\mathbf{s}^*(p_0)$ . Further, let linear independence constraint qualification (LICQ), second order sufficient conditions (SOSC) and strict complementarity (SC) hold for the solution vector  $\mathbf{s}^*(p_0)$ . Then,

- $\mathbf{s}^*(p_0)$  is a unique local minimizer of  $\mathcal{P}(p_0)$ .
- For parametric perturbations  $\Delta p$  in the neighborhood of  $p_0$ , there exists a unique, continuous, and differentiable vector function  $\mathbf{s}^*(p_0 + \Delta p)$  which is a KKT point satisfying LICQ and SOSC for  $\mathcal{P}(p_0 + \Delta p)$ .
- There exist positive Lipschitz constants  $L_s$  and  $L_V$  such that the solution vector and the optimal cost satisfy

$$\|\mathbf{s}^*(p_0 + \Delta p) - \mathbf{s}^*(p_0)\| \leq L_s \|\Delta p\| \quad (7)$$

$$\|V_N(p_0 + \Delta p) - V_N(p_0)\| \leq L_V \|\Delta p\| \quad (8)$$

*Proof:* See [23] ■

Since the implicit function  $\mathbf{s}^*(p)$  satisfies  $\varphi(\mathbf{s}^*(p), p) = 0$  for any  $p$  in the neighborhood of  $p_0$ , the implicit function theorem gives

$$\left. \frac{\partial}{\partial p} \varphi(\mathbf{s}^*(p), p) \right|_{p=p_0} = \frac{\partial \varphi}{\partial \mathbf{s}} \frac{\partial \mathbf{s}^*}{\partial p} + \frac{\partial \varphi}{\partial p} = \mathcal{M} \frac{\partial \mathbf{s}^*}{\partial p} + \mathcal{N} = 0 \quad (9)$$

where

$$\mathcal{M} := \begin{bmatrix} \nabla_{\mathbf{w}}^2 \mathcal{L}(\mathbf{s}^*(p_0)) & \nabla_{\mathbf{w}} c(\mathbf{w}^*(p_0)) & \nabla_{\mathbf{w}} g_{\mathbb{A}}(\mathbf{w}^*(p_0)) \\ \nabla_{\mathbf{w}} c(\mathbf{w}^*(p_0))^\top & 0 & 0 \\ \nabla_{\mathbf{w}} g_{\mathbb{A}}(\mathbf{w}^*(p_0))^\top & 0 & 0 \end{bmatrix}$$

is the KKT matrix, and

$$\mathcal{N} := \begin{bmatrix} \nabla_{\mathbf{w}}^2 \mathcal{L}(\mathbf{s}^*(p_0)) \\ \nabla_{\mathbf{w}} c(\mathbf{w}^*(p_0))^\top \\ \nabla_{\mathbf{w}} g_{\mathbb{A}}(\mathbf{w}^*(p_0))^\top \end{bmatrix}$$

Linearizing the solution vector  $\mathbf{s}^*(p)$  around  $p_0$  gives

$$\mathbf{s}^*(p_0 + \Delta p) = \mathbf{s}^*(p_0) + \frac{\partial \mathbf{s}^*}{\partial p} \Delta p + \mathcal{O}(\|\Delta p\|^2)$$

Ignoring the higher order terms, the solution of the neighboring problems  $p_0 + \Delta p$  can be approximated using (9) as,

$$\hat{\mathbf{s}}^*(p_0 + \Delta p) = \mathbf{s}^*(p_0) - \mathcal{M}^{-1} \mathcal{N} \Delta p \quad (10)$$

where  $\hat{\mathbf{s}}^*(p_0 + \Delta p)$  is the approximate primal-dual solution of the optimization problem  $\mathcal{P}(p_0 + \Delta p)$ , and  $\Delta \mathbf{s}^* := -\mathcal{M}^{-1} \mathcal{N} \Delta p$  is known as the *tangential predictor* or *linear predictor*. Simply put, we linearize the solution manifold  $\mathbf{s}^*(p)$  around  $p_0$ , and compute the approximate solution at  $p_0 + \Delta p$ . Note that since SOSC holds, the KKT matrix  $\mathcal{M}$  is invertible.

Computing  $\Delta \mathbf{s}^* := -\mathcal{M}^{-1} \mathcal{N} \Delta p$  requires only a linear solve, which is significantly cheaper to compute than solving the full NLP. From this we can see that once the solution to the NLP problem  $\mathcal{P}(x_i)$  is available for a given initial state  $x_i \in \mathcal{X}_{feas}$ , we can exploit the parametric property of the NLP to compute a fast approximate solution for an additional finite set of  $j = 1, \dots, N_p$  optimization problems  $\mathcal{P}(x_i + \Delta x_j)$  with initial states  $x_i + \Delta x_j \in \mathcal{X}_{feas}$  in the neighborhood of  $x_i$ . More precisely,  $\Delta x_j$  is sampled from a subset of arbitrary size  $\Delta \mathcal{X}_i \subset \mathcal{X}_{feas}$  such that  $x_i \in \text{int}(\Delta \mathcal{X}_i)$ . Using the tangential predictor (10), the corresponding optimal solution denoted by  $\hat{u}_j^*$  can then be evaluated, which only requires a linear solve. By exploiting the sensitivities, one can then generate  $M := N_s N_p$  number of training samples using only  $N_s$  NLP problems solved exactly. The pseudo-code for the proposed sensitivity-based data augmentation technique for MPC policy approximation is summarized in Algorithm 2. The idea of exploiting the parametric property of the MPC problem with respect to the initial states  $p = x(t)$  is also used in other parts of MPC literature such as real-time iteration [24], advanced-step MPC [25], [26] and adaptive horizon MPC [27], [28] to name a few.

*Remark 1 (Change in active constraint set):* Changes in the active set induces non-smooth points in the solution manifold, which cannot be captured by the tangential predictor. If the perturbation  $\Delta x_j$  induces a change in the set of active constraints, then one would have to solve a quadratic programming problem, often known as predictor QP, in order to produce a piecewise linear prediction manifold that can capture the non-smooth ‘‘corners’’ in the solution manifold [29]. Depending on the problem size and complexity, this may still be computationally cheaper than solving a full NLP problem. A simpler alternative is to discard any sensitivity updates that induce a change in active constraint set. By doing so, we do not augment the data set  $\mathcal{D}$  with points that induce a change in the active set. This is what is adopted in this paper (cf. line 10 in Algorithm 2).

A natural question that then arises is, how does augmenting the data set with inexact samples affect the policy approximation. To study this, consider an optimal solution manifold of an NLP denoted by  $\mathbf{s}^*(p)$  that we wish to approximate. Assume  $p \in \mathcal{X}_{feas}$  is sampled at  $N_s$  discrete points, and the corresponding optimal solution  $\mathbf{s}^*(p_i)$  for all  $i = 1, \dots, N_s$  is obtained by solving the NLP exactly. Now consider a piecewise linear inexact solution manifold  $\hat{\mathbf{s}}(p)$  that is given by the tangential predictor (10) using the exact solution at  $p_i$  in the  $i^{th}$  linear region for all  $i = 1, \dots, N_s$ . Let  $\Delta p_i$  be defined around each sample point  $p_i$ , such that  $\hat{\mathbf{s}}(p)$  exists for all  $p \in \mathcal{X}_{feas}$ . The true solution manifold  $\mathbf{s}^*(p)$ , and the piecewise linear

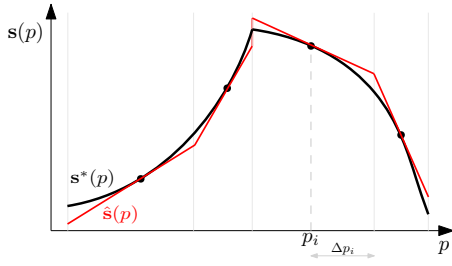


Fig. 1: Graphical illustration of a one dimensional solution manifold  $\mathbf{s}^*(p)$  (black), and a piecewise linear inexact manifold  $\hat{\mathbf{s}}(p)$  (red) approximated around the solutions at  $p_i$  (black dot) using a tangential predictor (shown here for  $N_s = 4$ ).

inexact solution manifold  $\hat{\mathbf{s}}(p)$  generated around the  $N_s$  samples is graphically illustrated for a one-dimensional case in Fig. 1. The maximum deviation between the true solution manifold  $\mathbf{s}^*(p)$  and the inexact manifold  $\hat{\mathbf{s}}(p)$  is quantified in the following Lemma.

*Lemma 1:* Given a solution manifold  $\mathbf{s}^*(p)$ , and a piecewise linear inexact manifold  $\hat{\mathbf{s}}(p)$  approximated around  $\mathbf{s}^*(p_i)$  using the tangential predictor (10) for all  $i = 1, \dots, N_s$ , with  $\|\Delta p_i\|$  chosen around each  $p_i$  such that the active constraint set remains the same and  $\exists \hat{\mathbf{s}}(p) \forall p \in \mathcal{X}_{feas}$ , then the following holds

$$\|\hat{\mathbf{s}}(p) - \mathbf{s}^*(p)\| \leq \max \left( \left\{ L_{p_i} \|\Delta p_i\|^2 \right\}_{i=1}^{N_s} \right) \quad (11)$$

for some positive constants  $L_{p_i}$

*Proof:* From the continuity and differentiability of  $\mathbf{s}^*(p)$  around each sample point  $\mathbf{s}^*(p_i)$  (cf. Theorem 1), there exists some positive Lipschitz constant  $L_{p_i}$  for the  $i^{th}$  piecewise linear region such that,

$$\|\hat{\mathbf{s}}(p_i) - \mathbf{s}^*(p_i)\| \leq L_{p_i} \|\Delta p_i\|^2$$

for all  $i = 1, \dots, N_s$ . Aggregating over all the regions, the maximum distance between the true manifold and the inexact manifold at any point  $p$  is then given by

$$\|\hat{\mathbf{s}}(p) - \mathbf{s}^*(p)\| \leq \max \left( \left\{ L_{p_i} \|\Delta p_i\|^2 \right\}_{i=1}^{N_s} \right)$$

We now want to approximate the solution manifold with any suitable parametric function  $\pi(p; \theta)$  using  $M := N_s N_p$  data samples.

*Definition 1 (Sufficiently rich parametrization):* Given a parametric function  $\pi(x; \theta)$  that is used to approximate a function  $F(x)$  over some domain  $x \in \mathbb{X}$ , the parametric function  $\pi(x; \theta)$  is said to be sufficiently richly parameterized if there exists  $\theta$  in some searchable domain  $\Theta$  such that  $F(x) = \pi(x; \theta)$  for all  $x \in \mathbb{X}$ .

*Assumption 1:* The functional form of  $\pi(p; \theta)$  has sufficiently rich parametrization and  $\exists \theta^*$  such that  $\mathbf{s}^*(p) = \pi(p; \theta^*)$ , and  $\exists \hat{\theta}$  such that  $\hat{\mathbf{s}}(p) = \pi(p; \hat{\theta})$ .

Consider the case where the training data is sampled by solving the NLP exactly at the  $M$  samples, which gives us

$$\theta_1 = \arg \min_{\theta} \sum_{i=1}^M \|\pi(p_i; \theta) - \mathbf{s}^*(p_i)\|^2 \quad (12)$$

Now consider the case where the training data is sampled from the inexact manifold  $\hat{\mathbf{s}}(p)$  at the same  $M$  samples, which gives us

$$\theta_2 = \arg \min_{\theta} \sum_{i=1}^M \|\pi(p_i; \theta) - \hat{\mathbf{s}}(p_i)\|^2 \quad (13)$$

The following result then establishes the error bound between the function approximators  $\pi(p; \theta_1)$  and  $\pi(p; \theta_2)$  stemming solely from augmenting the data set with inexact samples.

*Theorem 2:* Given Assumption 1 for the problem setup as in Lemma 1, if  $\theta_1$  and  $\theta_2$  are consistent estimators of (12) and (13) respectively, then

$$\|\pi(p; \theta_1) - \pi(p; \theta_2)\| \leq D \quad (14)$$

in probability as  $M \rightarrow \infty$ , where  $D := \max \left( \left\{ L_{p_i} \|\Delta p_i\|^2 \right\}_{i=1}^{N_s} \right)$

*Proof:* From Assumption 1 and Lemma 1, we have that

$$\|\hat{\mathbf{s}}(p) - \mathbf{s}^*(p)\| = \|\pi(p; \theta^*) - \pi(p, \hat{\theta})\| \leq D$$

If  $\theta_1$  is a consistent estimator of (12), then

$$\lim_{M \rightarrow \infty} \mathbb{P}(|\theta_1 - \theta^*| > \epsilon) = 0$$

for any  $\epsilon \geq 0$  where  $\mathbb{P}(\cdot)$  denotes the probability. Similarly, we have

$$\lim_{M \rightarrow \infty} \mathbb{P}(|\theta_2 - \hat{\theta}| > \epsilon) = 0$$

if  $\theta_2$  is a consistent estimator of (13). Combining these we get the inequality in (14) in probability as  $M \rightarrow \infty$ . ■

Note that although the above result considers  $M \rightarrow \infty$ , generating the training samples for (13) only requires solving  $N_s$  NLP problems exactly, the remaining  $M - N_s$  training samples are given by the tangential predictor (10). Since  $\mathbf{s}^*(p)$  is the true solution manifold, solving the MPC policy online would also give us  $\mathbf{s}^*(p)$ . Consequently, the case of  $M \rightarrow \infty$  in (12) can be seen as the MPC policy used online, and using the arguments of Theorem 2 we can state the following corollary that establishes the error bound between the MPC policy and the approximate policy trained using the inexact samples.

*Corollary 1:* Given Assumption 1 for the problem setup as in Lemma 1, if  $\theta_2$  is a consistent estimator of (13), then

$$\|\pi_{\text{mpc}}(p) - \pi(p; \theta_2)\| \leq D \quad (15)$$

in probability as  $M \rightarrow \infty$ .

Theorem 2/Corollary 1 thus provides an upper bound on the error solely induced due to augmenting the data set with a very large number of inexact samples based on  $N_s$  exact samples. The following (obvious) result considers the effect of the number of samples  $N_s$  around which the piecewise linear manifold is generated.

*Theorem 3:* Given the problem setup as in Lemma 1, and the estimators (12) and (13),

$$\lim_{N_s \rightarrow \infty} \|\pi(p; \theta_1) - \pi(p; \theta_2)\| = 0 \quad (16)$$

*Proof:* The proof of this result follows trivially from Lemma 1 where, as  $N_s \rightarrow \infty$ , the neighborhood of the tangential predictor gets smaller, i.e.  $\Delta p_i \rightarrow 0 \forall i$ , and we have

$$\lim_{N_s \rightarrow \infty} \|\hat{\mathbf{s}}(p) - \mathbf{s}^*(p)\| = 0$$

This implies that (12) and (13) are identical as  $N_s \rightarrow \infty$ , which proves our result. ■

To summarize, a large size of  $\Delta p_i$  implies that one can cheaply obtain several inexact data samples covering a larger subset of the feasible state-space. However, as the size of  $\Delta p_i$  grows, the approximation error induced by the sensitivity update also increases, as quantified by Theorem 2. As such Theorem 2 and Theorem 3 establish the trade-off between accuracy and computational cost of generating the training samples.

*Remark 2 (Sampling):* The proposed sensitivity-based data augmentation is not dependent on any particular sampling strategy, and can be used with different sampling strategies as one would have used with approximating the MPC policy without data augmentation. For any given sampling strategy, instead of solving all the sampled states exactly, one can instead use the tangential predictor (10) at the



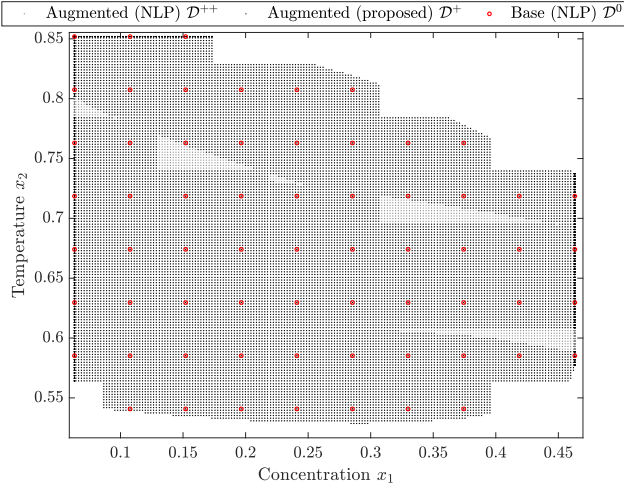


Fig. 2: Example A: Grid-based sampling of the feasible state space  $\mathcal{X}_{feas}$ . Red circles denote the base samples  $\mathcal{D}^0$ , where the corresponding optimal input is generated by solving the full optimization problem, the black dots denotes the augmented samples  $\mathcal{D}^+$  where the corresponding optimal input is generated using the tangential predictor, and the gray dots denote the samples in  $\mathcal{D}^{++}$  where the corresponding optimal input is generated by solving the full NLP.

samples that are within a user-defined neighborhood of an already existing sample.

*Remark 3 (Setpoint and weights):* The proposed sensitivity-based data augmentation is not restricted to parametric NLPs w.r.t. the initial states  $p_i = x_i$ , but can also be utilized by parameterizing the optimization problem with respect to other parameters such as reference trajectories  $x_i^{sp}$ , MPC tuning parameters such as weights in the cost function  $\omega_i$ , or measured disturbances  $d_i$  in addition to the initial states  $x_i$ , i.e.  $p_i = [x_i, x_i^{sp}, \omega_i, d_i]^T$ .

*Remark 4 (Linear MPC):* If  $\ell(\cdot, \cdot)$  is convex quadratic and  $f(\cdot)$  is linear, then  $\hat{s}^*(x + \Delta x) = s^*(x + \Delta x)$ , and consequently  $D = 0$  in Theorem 2.

The proposed approach is also not restricted to the MPC formulation (2), but can also be used with other variants of MPC formulation that typically involves solving nonlinear programming (NLP) problems, such as robust MPC [8], [9], and multistage scenario-based MPC [30] with moving horizon estimation [21] etc. to name a few. It can also be seen from (10) that the tangential predictor provides the primal-dual solution. Therefore, the proposed data augmentation scheme can also be used to augment a training data set consisting of the optimal dual variables. This is useful in cases where one would like to learn the dual policy such as in [11].

## IV. ILLUSTRATIVE EXAMPLES

### A. Benchmark CSTR

We now apply the proposed approach on a benchmark CSTR problem from [31] that was also used in the context of MPC policy approximation in [9]. This problem consists of two states, namely the scaled concentration and reactor temperature (denoted by  $x_1$  and  $x_2$ , respectively). The process is controlled using the coolant flow rate  $u$ . The model is given by

$$\begin{aligned} \dot{x}_1 &= (1/\tau)(1 - x_1) - kx_1e^{-\beta/x_2} \\ \dot{x}_2 &= (1/\tau)(x_f - x_2) + kx_1e^{-\beta/x_2} - \alpha u(x_2 - x_c) \end{aligned}$$

and the model parameters are  $\tau = 20$ ,  $k = 300$ ,  $\beta = 5$ ,  $x_f = 0.3947$ ,  $x_c = 0.3816$ , and  $\alpha = 0.117$ . Furthermore, we have  $\mathcal{X} =$

TABLE I: Example A: CPU time in [s] for generating the training samples.

	Cumulative CPU time [s]	Average CPU time per data point [s]	total no. of samples
Base grid (NLP) $\mathcal{D}^0$	35.766	0.5589	64
Augmented (proposed) $\mathcal{D}^+$	175.77	0.0074	23516
Augmented (NLP) $\mathcal{D}^{++}$	12960.4	0.529	24503

$[0.0632, 0.4632] \times [0.4519, 0.8519]$  and  $\mathcal{U} = [0, 2]$ . The setpoint is given by  $x^{sp} = [0.2632, 0.6519]^T$ . The stage cost is given by

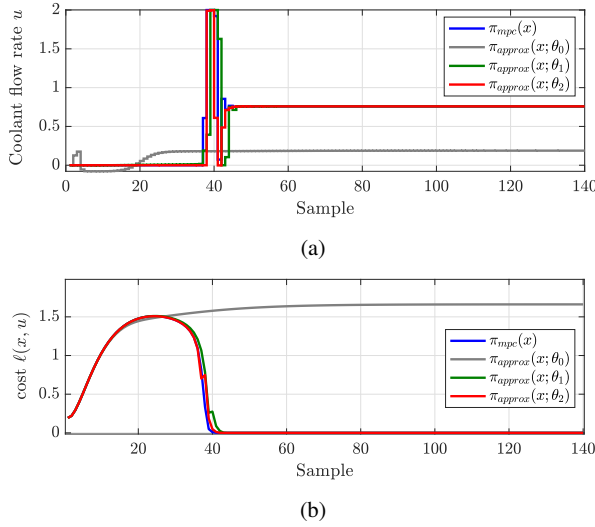
$$\ell(x, u) = \|x - x^{sp}\|^2 + 10^{-4}\|u\|^2$$

The MPC problem is solved with a sampling time of 3s and a prediction horizon of  $N = 140$ .

One approach to generate the learning samples is to use a grid-based sampling approach as done in [9], where the optimal input  $u^* = \pi_{mpc}(x)$  is evaluated at each grid point. In general, a small grid size is preferred since this would improve the MPC policy approximation. However, this would lead to large sample size  $N_s$ . The proposed approach enables us to choose a relatively larger grid size, where the corresponding optimal input  $\pi_{mpc}(x)$  is evaluated by solving the optimization problem. Additional grid points can then be generated with a smaller grid size around each grid point, and the corresponding optimal input can be computed by using the tangential predictor (10).

In this case, we first generated  $N_s = 64$  samples using a sparse grid with an interval of 0.0445 for both  $x_1$  and  $x_2$ . This base data set is denoted by  $\mathcal{D}^0$ , which is shown in red circles in Fig. 2. Since this is too few samples to approximate the MPC policy, we then augment the data set with additional samples, where around each grid point, the state space is further sampled with a smaller grid size of 0.0052 for both  $x_1$  and  $x_2$ . For these additional grid points, we obtained the corresponding inexact optimal input by using the tangential predictor (10) which is shown in black dots in Fig. 2. The additional grid points that induced a change in the active set were simply discarded. By doing so, we were able to cheaply generate and augment 23516 additional data points using only 64 full NLP computations. This data set is denoted by  $\mathcal{D}^+$ . To serve as benchmark, the optimal input at the additional grid points were also generated using Algorithm 1, i.e. by solving the full NLP (shown in gray dots in Fig. 2). Since the changes in the active constraints are not an issue when solving the full NLP, this approach generated an additional 24503 data points, which comes at a very high cost of computation. This data set is denoted by  $\mathcal{D}^{++}$ . The total number of data points, as well as the average and cumulative CPU time for generating the training data sets  $\mathcal{D}^0$ ,  $\mathcal{D}^+$  and  $\mathcal{D}^{++}$  are summarized in Table I. This clearly shows the benefit of the proposed approach in terms of the computational cost of generating the training samples.

Using the generated data sets, we approximate the MPC policy using deep neural networks with 5 hidden layers and 10 neurons in each hidden layer with the hyperbolic tangent sigmoid as the activation function in each neuron. Note that the type of the function approximator, its hyperparameters, and the specifics of the MSE minimization problem are not the focus of this paper, and one may find an alternative/better network architecture than the one used here, for example by using Bayesian optimization [32]. The NLP problems  $\mathcal{P}(x)$  were solved using IPOPT [33] with MUMPS linear solver. The neural network was trained using fitnet in MATLAB v2020b. All computations were performed on a 2.6 GHz processor with 16GB memory. Source codes for the simulation results



**Fig. 3:** Example A: Closed-loop simulation results comparing the (a) control actions and (b) cost  $\ell(x, u)$  given by the MPC policy  $\pi_{\text{mpc}}(x)$  (blue) and the approximate policies  $\pi_{\text{approx}}(x; \theta_0)$  trained using sparsely sampled data (gray),  $\pi_{\text{approx}}(x; \theta_1)$  trained using finely sampled data obtained by solving full NLP (green), and  $\pi_{\text{approx}}(x; \theta_2)$  trained using inexact samples using the proposed data augmentation approach (red).

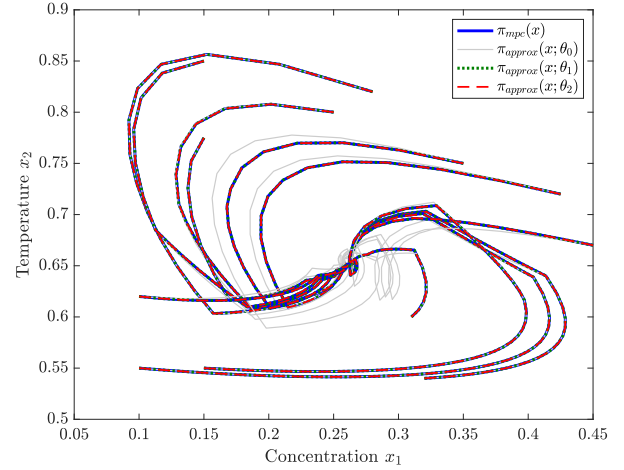
presented in this technical note can be found in the GitHub repository <https://github.com/dinesh-krishnamoorthy/Sensitivity-DataAugmentation>.

First we approximate the MPC policy using only the sparsely sampled data set  $\mathcal{D}^0$ , which gives the control policy  $\pi_{\text{approx}}(x; \theta_0)$ . We then approximate the MPC policy using the computationally expensive full data set  $\mathcal{D}_1 := \mathcal{D}^0 \cup \mathcal{D}^{++}$ , which gives the approximate policy  $\pi_{\text{approx}}(x; \theta_1)$ . We then approximate the MPC policy using the proposed sensitivity-based augmented data set  $\mathcal{D}_2 := \mathcal{D}^0 \cup \mathcal{D}^+$ , which gives the approximate policy  $\pi_{\text{approx}}(x; \theta_2)$ .

Fig. 3a compares the closed-loop control actions provided by the different policies. The control trajectory provided by the MPC policy  $\pi_{\text{mpc}}(x)$  (shown in blue) serves as the ideal benchmark. The closed-loop control trajectory using the approximate policy  $\pi_{\text{approx}}(x; \theta_0)$  trained using the sparsely sampled data set  $\mathcal{D}^0$  is shown in gray, where it can be seen that  $\mathcal{D}^0$  fails to approximate the MPC policy due to insufficient training data samples. The closed-loop control trajectory using the approximate policy  $\pi_{\text{approx}}(x; \theta_1)$  trained using the full NLP data set  $\mathcal{D}_1$  is shown in green, which is able to approximate the MPC policy well as one would expect. Finally, the closed-loop control trajectory using the approximate policy  $\pi_{\text{approx}}(x; \theta_2)$  trained using the proposed augmented data set  $\mathcal{D}_2$  is shown in red, which is also able to approximate the MPC policy closely, although the approximate policy  $\pi_{\text{approx}}(x; \theta_2)$  is trained using inexact samples, that are significantly cheaper to generate than the full NLP (cf. Table I).

The corresponding closed-loop cost  $\ell(x, u)$  is also shown in Fig. 3b. Here it can be clearly seen that the performance using the approximate policy  $\pi_{\text{approx}}(x; \theta_2)$  trained using the proposed data augmentation scheme is almost identical to the performance of the MPC policy, and the approximate policy  $\pi_{\text{approx}}(x; \theta_1)$  trained using the full NLP.

Next, we compare the closed-loop trajectories when starting from different initial conditions. The state trajectories using the MPC policy  $\pi_{\text{mpc}}(x)$  (blue) and the approximate policies  $\pi_{\text{approx}}(x; \theta_0)$  (gray),



**Fig. 4:** Example A: State trajectories starting from different initial conditions given by the MPC policy  $\pi_{\text{mpc}}(x)$  (blue) and the approximate policies  $\pi_{\text{approx}}(x; \theta_0)$  trained using sparsely sampled data (gray),  $\pi_{\text{approx}}(x; \theta_1)$  trained using finely sampled data obtained by solving full NLP (green dotted), and  $\pi_{\text{approx}}(x; \theta_2)$  trained using inexact samples using the proposed data augmentation approach (red dashed).

$\pi_{\text{approx}}(x; \theta_1)$  (green dotted lines), and  $\pi_{\text{approx}}(x; \theta_2)$  (red dashed lines) are shown in Fig. 4. Here it can be clearly seen that the state trajectories obtained using the proposed data augmentation scheme (red dashed lines) is almost identical to the MPC policy (blue) and the approximate policy trained using the full NLP (green dotted lines) for all the different initial conditions, whereas the trajectories obtained by using the approximate policy trained using sparsely sampled data (gray) deviates significantly from the other trajectories.

This clearly demonstrates that by using the proposed data augmentation approach, we can sparsely sample  $\mathcal{X}_{\text{feas}}$ , which reduces the number of optimization problems that needs to be solved offline, and augment the data set with several additional inexact samples that can be obtained cheaply. Consequently, the overall time and computational cost required to generate the training samples is significantly lesser, and at the same time, the approximate policy trained using the augmented samples provides similar performance as the MPC policy.

## B. Building Climate Control

We now illustrate the proposed approach on a building climate control problem, for which there have been several works considering MPC as the control strategy, see [10], [34] and the references therein. In our simulations, we model the heat dynamics of a building based on the modeling framework from [35], as shown below,

$$\begin{aligned} \frac{dT_s}{dt} &= \frac{1}{R_{is}C_s}(T_i - T_s) \\ \frac{dT_i}{dt} &= \frac{1}{R_{is}C_i}(T_s - T_i) + \frac{1}{R_{ih}C_i}(T_h - T_i) + \frac{A_w\Phi}{C_i} \\ &\quad + \frac{1}{R_{ie}C_i}(T_e - T_i) + \frac{1}{R_{ia}C_i}(T_a - T_i) \end{aligned}$$

$$\begin{aligned} \frac{dT_h}{dt} &= \frac{1}{R_{ih}C_h}(T_i - T_h) + \frac{u}{C_h} \\ \frac{dT_e}{dt} &= \frac{1}{R_{ie}C_e}(T_i - T_e) + \frac{1}{R_{ea}C_e}(T_a - T_e) + \frac{A_e\Phi}{C_e} \end{aligned}$$

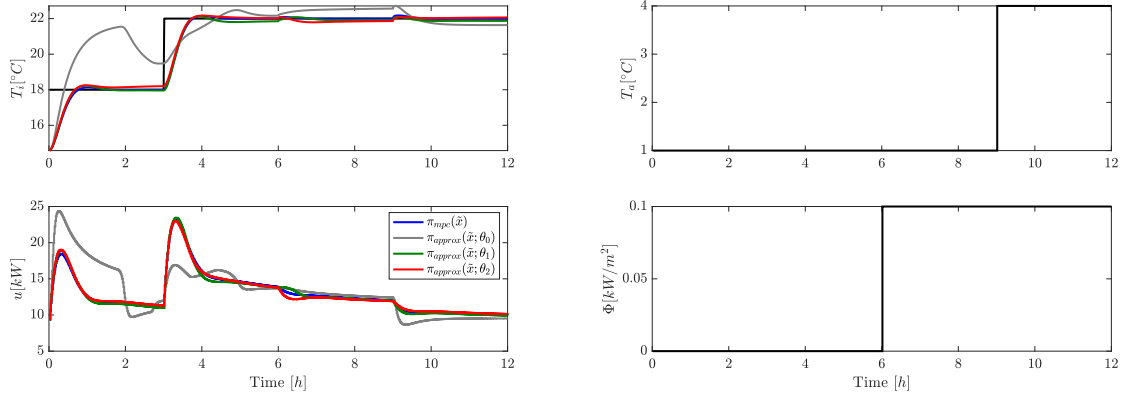


Fig. 5: Example B: Closed-loop simulation results comparing performance of the MPC policy  $\pi_{\text{mpc}}(\tilde{x})$  (blue) and the approximate policies  $\pi_{\text{approx}}(\tilde{x}; \theta_0)$  trained using sparsely sampled data (gray),  $\pi_{\text{approx}}(\tilde{x}; \theta_1)$  trained using finely sampled data obtained by solving full NLP (green), and  $\pi_{\text{approx}}(\tilde{x}; \theta_2)$  trained using inexact samples using the proposed data augmentation approach (red).

where the subscripts  $(\cdot)_s, (\cdot)_i, (\cdot)_h, (\cdot)_e$  and  $(\cdot)_a$  denotes the sensor, building interior, heater, building envelop, and ambient, respectively.  $T$  denotes the temperature,  $R$  denotes the thermal resistance,  $C$  denotes the heat capacity and  $u$  denotes the heat flux. The solar irradiation  $\Phi$  enters the building interior through the effective window area  $A_w$  in addition to heating the building envelop with effective area  $A_e$ . The states are given by  $x = [T_s, T_i, T_h, T_e]^\top$  with  $T_{(\cdot)} \in [12, 40]^\circ\text{C}$ . The ambient temperature  $T_a \in [-5, 20]^\circ\text{C}$  and the solar irradiation  $\Phi \in [0, 0.2] \text{ kW/m}^2$  are measured disturbances. The parameter values used in the model are taken from [35]. The objective is to drive the interior temperature  $T_i$  to a desired setpoint  $T_i^{sp} \in [18, 25]^\circ\text{C}$ , while penalizing the rate of change of the input usage  $u \in [0, 40] \text{ kW}$ . The stage cost is given by

$$\ell(x, u) = (T_i - T_i^{sp})^2 + 0.1(\Delta u)^2$$

The MPC problem is formulated with a sampling time of 1 min and a prediction horizon of  $N = 3$  hours. The goal is to approximate the MPC policy  $\pi_{\text{mpc}}(\tilde{x})$ . In this example  $\tilde{x} = [T_s, T_i, T_h, T_e, T_i^{sp}, T_a, \Phi, u]^\top$ , which requires sampling a 8-dimensional space in order to generate the training samples. This example demonstrates the proposed approach when we have a higher-dimensional augmented state-space (cf. Remark 3).

We randomly generate a total of 6930 samples, out of which, 330 samples were generated by solving the optimization problem (denoted by  $\mathcal{D}^0$ ) and 6600 samples were generated using the proposed sensitivity update as shown in Algorithm 2 (denoted by  $\mathcal{D}^+$ ). As a benchmark, the same 6600 samples were also generated by solving the full NLP as shown in Algorithm 1 (denoted by  $\mathcal{D}^{++}$ ).

Using the generated training samples, we approximate the MPC policy using deep neural networks with 3 hidden layers and 10 neurons in each hidden layer, with rectified linear units (ReLU) as the activation function in each neuron. We first approximate the policy using only  $\mathcal{D}^0$ . This is denoted by  $\pi_{\text{approx}}(\tilde{x}; \theta_0)$ . The policy trained on the full data set  $\mathcal{D}_1 := \mathcal{D}^0 \cup \mathcal{D}^{++}$  is denoted by  $\pi_{\text{approx}}(\tilde{x}; \theta_1)$ . Similarly, the policy trained on the proposed sensitivity-based augmented data set  $\mathcal{D}_2 := \mathcal{D}^0 \cup \mathcal{D}^+$  is denoted by  $\pi_{\text{approx}}(\tilde{x}; \theta_2)$ .

We test the performance of the approximate MPC policy for a total simulation time of 12 hours, with changes in the setpoint (at  $t = 3$  h), solar irradiation (at time  $t = 6$  h), and ambient temperature ( $t = 9$  h). Fig. 5 shows the closed loop simulation results using the traditional MPC control law  $\pi_{\text{mpc}}(\tilde{x})$  (blue) obtained by solving the MPC problem online, and the performance of the

approximate policies  $\pi_{\text{approx}}(\tilde{x}; \theta_0)$  (gray),  $\pi_{\text{approx}}(\tilde{x}; \theta_1)$  (green), and  $\pi_{\text{approx}}(\tilde{x}; \theta_2)$  (red). We see  $\pi_{\text{approx}}(\tilde{x}; \theta_0)$  does not approximate the MPC policy accurately, due to too few training samples. On the other hand,  $\pi_{\text{approx}}(\tilde{x}; \theta_1)$  and  $\pi_{\text{approx}}(\tilde{x}; \theta_2)$  mimics the MPC policy closely. However obtaining the training data set for  $\pi_{\text{approx}}(\tilde{x}; \theta_1)$  is significantly more costly than the proposed sensitivity-based data augmentation scheme used to train  $\pi_{\text{approx}}(\tilde{x}; \theta_2)$ . From this it can be seen that the proposed sensitivity-based data augmentation framework can be used to parameterize the measured disturbances, setpoints, and the control input in addition to the states in order to handle time varying disturbances and setpoints, and approximate the MPC policy closely by augmenting the data set with inexact samples.

## V. CONCLUSIONS

To conclude, this technical note addresses an important implementation aspect of MPC policy approximation, namely the cost of training. Algorithm 2 exploits the parametric sensitivities to augment several training samples using the solution of a single optimization problem. It was shown that by using the proposed approach, one can

- sample the feasible state space sparsely, hence reducing the number of optimization problems that needs to be solved offline,
- and augment the data set with additional samples using a tangential predictor.

The error due to augmenting the data set with inexact samples was also quantified, and it was shown to depend quadratically on the max distance between the augmented sample and the original sample  $\|\Delta x_i\|^2$ . More broadly, the proposed data augmentation scheme can be used in any policy approximation where the training data comprises of optimal state-action pairs that is sampled from a policy given by a nonlinear programming problem. As such, this paper is a first step towards a data augmentation framework for approximate optimal control problems.

## ACKNOWLEDGMENT

Helpful discussions with Prof. Sebastien Gros from the Department of Engineering Cybernetics at the Norwegian University of Science and Technology is gratefully acknowledged.

## REFERENCES

- [1] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, “The explicit linear quadratic regulator for constrained systems,” *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

- [2] P. Tøndel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MPC solutions," *Automatica*, vol. 39, no. 3, pp. 489–497, 2003.
- [3] 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.
- [4] A. Chakrabarty, V. Dinh, M. J. Corless, A. E. Rundell, S. H. Žak, and G. T. Buzzard, "Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 135–148, 2017.
- [5] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Transactions on Cybernetics*, 2020.
- [6] 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," in *2018 Annual American control conference (ACC)*. IEEE, 2018, pp. 1520–1527.
- [7] L. H. Csekő, M. Kvasnica, and B. Lantos, "Explicit MPC-based RBF neural network controller design with discrete-time actual kalman filter for semiactive suspension," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1736–1753, 2015.
- [8] J. A. Paulson and A. Mesbah, "Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction," *IEEE Control Systems Letters*, 2020.
- [9] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, 2018.
- [10] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate model predictive building control via machine learning," *Applied Energy*, vol. 218, pp. 199–216, 2018.
- [11] X. Zhang, M. Bujarbaruah, and F. Borrelli, "Safe and near-optimal policy learning for model predictive control using primal-dual neural networks," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 354–359.
- [12] D. P. Bertsekas, *Reinforcement learning and optimal control*. Athena Scientific Belmont, MA, 2019.
- [13] N. Esmaili, C. Sammut, and G. Shirazi, "Behavioural cloning in control of a dynamic system," in *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, vol. 3. IEEE, 1995, pp. 2904–2909.
- [14] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018. [Online]. Available: <http://dx.doi.org/10.1561/23000000053>
- [15] M. Novak and T. Dragicevic, "Supervised imitation learning of finite-set model predictive control systems for power electronics," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 2, pp. 1717–1723, 2020.
- [16] T. Tran, T. Pham, G. Carneiro, L. Palmer, and I. Reid, "A bayesian data augmentation approach for learning deep models," in *Advances in neural information processing systems*, 2017, pp. 2797–2806.
- [17] L. Taylor and G. Nitschke, "Improving deep learning with generic data augmentation," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 1542–1547.
- [18] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.
- [19] R. E. Kalman, "When Is a Linear Control System Optimal?" *Journal of Basic Engineering*, vol. 86, no. 1, pp. 51–60, 03 1964. [Online]. Available: <https://doi.org/10.1115/1.3653115>
- [20] A. Keshavarz, Y. Wang, and S. Boyd, "Imputing a convex objective function," in *2011 IEEE international symposium on intelligent control*. IEEE, 2011, pp. 613–619.
- [21] B. Karg and S. Lucia, "Approximate moving horizon estimation and robust nonlinear model predictive control via deep learning," *Computers & Chemical Engineering*, p. 107266, 2021.
- [22] D. Krishnamoorthy and S. Skogestad, "Real-time optimization strategies using surrogate optimizers," in *Proceedings of the 2019 Foundations in Process Analytics and Machine Learning*, 2019.
- [23] A. V. Fiacco, "Sensitivity analysis for nonlinear programming using penalty methods," *Mathematical programming*, vol. 10, no. 1, pp. 287–311, 1976.
- [24] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.
- [25] V. M. Zavala and L. T. Biegler, "The advanced-step nmpc controller: Optimality, stability and robustness," *Automatica*, vol. 45, no. 1, pp. 86–93, 2009.
- [26] J. Jäschke, X. Yang, and L. T. Biegler, "Fast economic model predictive control based on nlp-sensitivities," *Journal of Process Control*, vol. 24, no. 8, pp. 1260–1272, 2014.
- [27] D. W. Griffith, L. T. Biegler, and S. C. Patwardhan, "Robustly stable adaptive horizon nonlinear model predictive control," *Journal of Process Control*, vol. 70, pp. 109–122, 2018.
- [28] D. Krishnamoorthy, L. T. Biegler, and J. Jäschke, "Adaptive horizon economic nonlinear model predictive control," *Journal of Process Control*, vol. 92, pp. 108–118, 2020.
- [29] J. F. Bonnans and A. Shapiro, "Optimization problems with perturbations: A guided tour," *SIAM review*, vol. 40, no. 2, pp. 228–264, 1998.
- [30] S. Lucia and B. Karg, "A deep learning-based approach to robust nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 511–516, 2018.
- [31] D. Q. Mayne, E. C. Kerrigan, E. Van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.
- [32] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.
- [33] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.
- [34] S. Prívará, Z. Váňa, J. Cigler, F. Oldewurtel, and J. Komárek, "Role of MPC in building climate control," in *Computer Aided Chemical Engineering*. Elsevier, 2011, vol. 29, pp. 728–732.
- [35] P. Bacher and H. Madsen, "Identifying suitable models for the heat dynamics of buildings," *Energy and Buildings*, vol. 43, no. 7, pp. 1511–1522, 2011.