Sensitivity-based Data Augmentation for Learning an Approximate Model Predictive Controller

Dinesh Krishnamoorthy *Member, IEEE*

***Abstract*— Approximating model predictive control (MPC) policy using expert supervised learning techniques, such as deep neural networks (DNN) requires labeled training data sets. This is typically obtained by sampling the state-space and evaluating the control law by solving the numerical optimization problem offline for each sample. Although the resulting approximate MPC policy can be cheaply evaluated online, generating large training samples to learn the MPC control policy can be time consuming and prohibitively expensive. This is one of the fundamental bottlenecks that limit the design and implementation of deep-learning based approximate MPC policy. This technical note aims to address this issue, and proposes a novel sensitivity-based data augmentation scheme for policy approximation. The proposed approach is based on exploiting the NLP 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 predetermine the optimal control 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 grows exponentially with the number of decision variables and constraints. The extension to nonlinear systems is also not straightforward.

An alternative approach is to use some parametric function approximator, such as artificial neural networks (ANN) to approximate the MPC control law. Although this idea dates back to the mid 90s [3], the use of neural networks to approximate the MPC control law 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 control law $\pi_{\text{mpc}}(\cdot)$ is approximated using any suitable regression technique using $\{(x_i, u_i^*)\}_{i=1}^{N_s}$ as the training data set, such that the trained model $\pi_{\text{approx}}(\cdot)$ can be used online to cheaply evaluate the optimal control input. This approach is also studied more generally in the context of *policy approximation*, where such a framework is known as "expert supervised learning" [12].

However, one of the main bottleneck of this approach is that, generating the training data set 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. 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. For example, the authors in [9] reported a computation time of roughly 500 hours on a Quad-Core PC[1] 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 control law.

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* [13], [14]. This has been extensively studied in the context of deep learning for image classification problems, where geometric transformations (such as rotation, cropping etc.) and photometric transformations (such as color, contrast, brightness etc.) are often used to augment the existing data set with artificially generated training samples. Unfortunately, such data augmentation techniques are not applicable in the context of MPC 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 considered as a parametric optimization problem parameterized with respect to the initial state x_i. The NLP sensitivity then tells us how the optimal solution u_i^* changes for perturbations Δx 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 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. Thus, the main contribution of this technical note is a sensitivity-based data augmentation technique to efficiently and cheaply generate training

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

Dinesh Krishnamoorthy is with Department of Chemical Engineering, Norwegian University of Science and Technology, 7491, Trondheim, Norway dinesh.krishnamoorthy@ntnu.no

[1]without parallelization of the sampling and validation

data samples that can be used to approximate the MPC control law.

The reminder of the paper is organized as follows. Section II formulates the problem and recalls the approximate explicit MPC framework. The sensitivity-based data augmentation technique to efficiently generate the training samples is presented in Section III. 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

$$x(t+1) = f\left(x(t), u(t)\right) \tag{1}$$

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

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

$$\text{s.t. } x(k+1|t) = f(x(k|t), u(k|t)) \tag{2b}$$

$$x(k|t) \in \mathcal{X}, \quad u(k|t) \in \mathcal{U} \tag{2c}$$

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

$$x(0|t) = x(t) \tag{2e}$$

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, N is the length of the prediction horizon, (2c) denotes the path constraints, (2d) denotes the terminal constraint, and (2e) denotes the initial condition constraints. In the traditional MPC paradigm, the optimization problem (2) is solved at each sample time t using $x(t)$ as the state feedback, and the optimal input $u^*(0|t)$ is injected into the plant in a receding horizon fashion. This implicitly leads to the control law

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

B. Approximate Explicit MPC

This subsection recalls the underlying idea of the approximate explicit MPC framework common to works such as [3], [5], [8] and [9]. To approximate the MPC control law (3), the feasible state space $\mathcal{X}$ is sampled to generate N_s randomly chosen initial 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 functional form $\pi_{\text{approx}}(x;\theta)$ parameterized by the parameters θ is trained in order to minimize the mean squared error

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

This is summarized in Algorithm 1.

Deep neural networks have become a popular choice for the functional form for approximating the MPC control law. For a deep neural network with L hidden layers and M neurons in each hidden layer,

$$\pi_{\text{approx}}(x;\theta) = h_{L+1} \circ \alpha_L \circ h_L \circ \cdots \circ \alpha_1 \circ h_1 \tag{5}$$

Each hidden layer is made of an affine function $h_l(\xi_{l-1}) = W_l^{\mathsf{T}} \xi_{l-1} + b_l \quad \forall l = 1, \dots, L$ where $\xi_{l-1} \in \mathbb{R}^M$ is the output of the previous layer and $\xi_0 = x$. $\alpha_l(h_l) : \mathbb{R} \rightarrow \mathbb{R}$ denotes a nonlinear activation function such as sigmoid, rectified linear unit (ReLU) etc. The parameter θ contains all the weights W_l and biases b_l.

Once the network architecture is trained, the approximate control law $\pi_{\text{approx}}(x;\hat{\theta})$ can be used online to cheaply evaluate the optimal control input.

Algorithm 1 Learning an approximate MPC control law

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

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

Output: $\pi_{\text{approx}}(x;\hat{\theta})$

III. PROPOSED METHOD

As mentioned in the previous section, generating the training data requires solving N_s numerical optimization problems, 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 learn the MPC control law. To keep the notation light, we rewrite the MPC problem (2) into a standard parametric NLP problem of the form,

$$V_N(p) = \min_{\mathbf{w}} \quad J(\mathbf{w}, p) \tag{6a}$$

$$\text{s.t.} \quad c(\mathbf{w}, p) = 0 \tag{6b}$$

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

where $p = x(0|t) = x(t)$ is the initial state, the decision variables $\mathbf{w} := [u(0|t), \dots, u(N-1|t), x(1|t), \dots, x(N|t)]^{\mathsf{T}}$ the cost (2a) is denoted by (6a), the system equations (2b) are denoted by (6b), the path constraints (2c) and terminal constraints (2d) are collectively denoted by (6c). 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 (6) is given by

$$\mathcal{L}(\mathbf{w}, p, \lambda, \mu) := J(\mathbf{w}, p) + \lambda^{\mathsf{T}} c(\mathbf{w}, p) + \mu^{\mathsf{T}} g(\mathbf{w}, p) \tag{7}$$

where λ and μ are the Lagrangian multipliers of (6b) and (6c) respectively, and the KKT conditions for this problem is given by,

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, p, \lambda, \mu) = 0 \tag{8a}$$

$$c(\mathbf{w}, p) = 0 \tag{8b}$$

$$g(\mathbf{w}, p) \leq 0 \tag{8c}$$

$$\mu_i g_i(\mathbf{w}, p) = 0, \quad \mu_i \geq 0 \quad \forall i \tag{8d}$$

Any point $\mathbf{s}^*(p) := [\mathbf{w}^{*\mathsf{T}}, \lambda^{*\mathsf{T}}, \mu^{*\mathsf{T}}]^{\mathsf{T}}$ that satisfies the KKT conditions (8) 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 $\mu_{\mathbb{A}} > 0$. This set of KKT conditions can be represented compactly as $\varphi(\mathbf{s}(p), p)$.

Theorem 1 ([15]): Let $J(\cdot,\cdot)$ and $c(\cdot,\cdot)$ 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 hold for the solution vector $\mathbf{s}^*(p)$. Then,

- $\mathbf{s}^*(p_0)$ is a unique local minimizer of $\mathcal{P}(p_0)$.
- For parametric perturbations Δp 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 SSOSC for $\mathcal{P}(p_0 + \Delta p)$.

- There exists positive Lipschitz constants L_s and L_V such that the solution vector and the optimal cost satisfies

$$\|\mathbf{s}^*(p_0+\Delta p)-\mathbf{s}^*(p_0)\| \le L_s\|\Delta p\| \tag{9}$$

$$\|V_N(p_0+\Delta p)-V_N(p_0)\| \le L_V\|\Delta p\| \tag{10}$$

Proof: See [15] ■

Linearizing the KKT conditions $\varphi(\mathbf{s}(p_0),p)$ around $\mathbf{s}^*(p_0)$ gives

$$\begin{aligned}0 &= \nabla_{\mathbf{s}}\varphi(\mathbf{s}^*(p_0+\Delta p))\\ &= \nabla_{\mathbf{s}}\varphi(\mathbf{s}^*(p_0)) + \frac{\partial}{\partial p}\nabla_{\mathbf{s}}\varphi(\mathbf{s}^*(p_0))\Delta p + \mathcal{O}(\|\Delta p\|^2)\end{aligned}$$

Consequently,

$$\frac{\partial}{\partial p}\nabla_{\mathbf{s}}\varphi(\mathbf{s}^*(p_0))\Delta p = \left(\mathcal{M}\frac{\partial \mathbf{s}^*}{\partial p}+\mathcal{N}\right)\Delta p \approx 0 \tag{11}$$

where

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

is the KKT matrix and

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

Therefore, the solution of the neighboring problems $p_0+\Delta p$ can be obtained from

$$\hat{\mathbf{s}}^*(p_0+\Delta p) = \mathbf{s}^*(p) + \frac{\partial \mathbf{s}^*}{\partial p}\Delta p \tag{12}$$

where $\hat{\mathbf{s}}^*(p_0+\Delta p)$ is the approximate primal-dual solution of the optimization problem $\mathcal{P}(p_0+\Delta p)$. Therefore, $\Delta\mathbf{s}^* := \hat{\mathbf{s}}^*(p_0+\Delta p)-\mathbf{s}^*(p)$ can be computed as the solution to the system of linear equations,

$$\mathcal{M}\Delta\mathbf{s}^* = -\mathcal{N}\Delta p \tag{13}$$

From this we can see that once the solution to the NLP problem $\mathcal{P}(x_i)$ is available for a given initial state $x_i \in \mathcal{X}$, 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 $x_i+\Delta x_j \in \mathcal{X}$ in the neighborhood of x_i. More precisely, Δx_j is sampled from a subset of arbitrary size $\Delta\mathcal{X}_i \subset \mathcal{X}$ such that $x_i \in \text{int}(\Delta\mathcal{X}_i)$. Using the solution to a system of linear equations (13) the corresponding optimal solution, denoted by $\hat{u}_j^*$ can then be cheaply evaluated using (13). By exploiting the sensitivities, one can generate N_sN_p number of training samples using only N_s offline optimization problems. The pseudo-code for the proposed sensitivity-based data augmentation technique to learn an approximate MPC control law is summarized in Algorithm 2.

Remark 1 (On the choice of $\Delta\mathcal{X}_i$): The size of the subset $\Delta\mathcal{X}_i$ with x_i in its interior is defined by the user, and can be chosen independently for each x_i. In general, a large size of $\Delta\mathcal{X}_i$ implies that one can obtain data samples covering a larger subset of the state-space using the sensitivity update. However, as the size of $\Delta\mathcal{X}_i$ grows, the approximation error induced by the sensitivity update also increases, as quantified by (9). The choice of $\Delta\mathcal{X}_i$ therefore allows the user to trade-off between accuracy and the computational cost of generating the training samples.

Remark 2 (Linear MPC): In the case of linear MPC where $\ell(\cdot,\cdot)$ is convex quadratic and $f(\cdot)$ is linear, then $\hat{\mathbf{s}}^*(x+\Delta x) = \mathbf{s}^*(x+\Delta x)$.

Remark 3 (Change in active constraint set): If the perturbation Δx_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 obtain the approximate solution $\hat{\mathbf{s}}^*(x+\Delta x)$ [16], which may still be computationally cheaper than solving a full NLP problem. Alternatively, one can simply discard the sensitivity updates that induce a change in active constraint set, and not include it in the dataset $\mathcal{D}$.

Algorithm 2 Sensitivity-based data augmentation to learn an approximate MPC control law

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

```
 1: for $i = 1,\dots,N_s$ do
 2:     Sample $x_i \in \mathcal{X}$
 3:     $\mathbf{s}^*(x_i) \leftarrow$ 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:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_i+\Delta x_j, \hat{u}_j^*)\}$
11:     end for
12: end for
13: $\hat{\theta} \leftarrow \arg\min_\theta \frac{1}{N_sN_p}\sum_{i=1}^{N_sN_p}\|\pi_{\text{approx}}(x_i;\theta)-u_i^*\|^2$
```

Output: $\pi_{\text{approx}}(x;\hat{\theta})$

Note that the idea of exploiting the parametric property of the MPC problem with respect to the initial states is also used in other parts of MPC literature such as the advanced-step MPC [17], [18] and adaptive horizon MPC [19], [20].

The proposed sensitivity-based data augmentation scheme can also be utilized by parameterizing the optimization problem with respect to other time-varying parameters such as exogenous disturbances, time varying setpoints, or tuning parameters in addition to the initial states. 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) problem, such as robust MPC [9], and multistage scenario-based MPC [21] to name a few.

IV. ILLUSTRATIVE EXAMPLES

A. Benchmark CSTR

We now apply the proposed approach on a benchmark CSTR problem from [22] that was also used in the context of approximate MPC in [9]. This problem consists of two states, namely the 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

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

and the model parameters are $\tau = 20$, $k = 300$, $M = 5$, $x_f = 0.3947$, $x_c = 0.3816$, and $\alpha = 0.117$. The feasible state space is given by $\mathcal{X} = [0.0632, 0.4632] \times [0.4519, 0.8519]$ and $\mathcal{U} = [0, 2]$. The setpoint is given by $x^{sp} = [0.2632, 0.6519]^\mathsf{T}$.

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 3 s 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

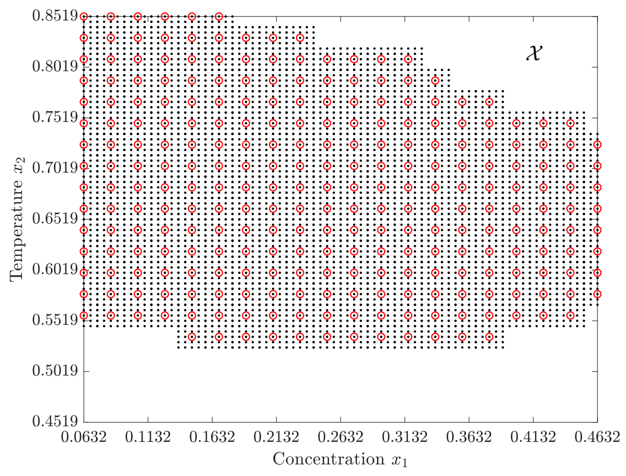

Fig. 1: Example A: Grid-based sampling of the state space $\mathcal{X}$. Red circles denote the samples, where the corresponding optimal input is generated by solving the full optimization problem, and the black dots denotes the samples where the corresponding optimal input is generated using the sensitivity update.

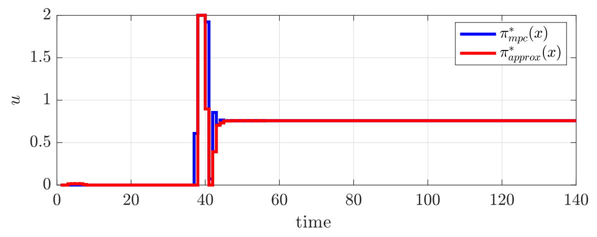

Fig. 2: Example A: Closed-loop simulation results comparing performance of the traditional online MPC $\pi_{\text{mpc}}(x)$ (blue) and the approximate explicit MPC $\pi_{\text{approx}}(x)$ (red).

[9], where the state space $\mathcal{X}$ is divided into finite number of uniform grids. The optimal input $u^* = \pi_{\text{mpc}}(x)$ is then evaluated at each grid point. In general, a small grid size is preferred since this would improve the MPC 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 $\pi_{\text{mpc}}(x)$ 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 sensitivity update (13). This is shown in Fig. 1 where the state space is sampled into grids with a larger grid size (shown in red circle) with an interval of 0.0211 for x_1 and x_2, and around each grid point, the state space is further sampled with a smaller grid size (shown in black dots) with an interval of 0.0052 for x_1 and x_2.

By using this approach, we were able to generate a total of 6968 training samples, out of which 268 training samples (shown in red circles) were generated by solving a numerical optimization problem, and 6700 training samples (shown in black dots) were generated by using the sensitivity update. Note that not all the grid points have a feasible solution. Hence only the feasible points are included in the training data set $\mathcal{D}$ and are shown in Fig. 1.

The optimization problem was solved offline using `IPOPT` [23]. The samples were generated on a 2.6 GHz processor with 16GB memory. The minimum, maximum and average CPU time for generating the training samples using the full optimization problem and the sensitivity-update are summarized in Table I, from which it can be seen that the CPU times differ roughly by a factor of 100.

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

	min	avg	max	# of samples
Full NLP	0.2837	0.5980	2.7697	268
Sensitivity update	0.0038	0.0055	0.0222	6700

Using the generated training samples, we approximate the MPC control law using deep neural networks with $L = 3$ layers and $M = 10$ neurons with rectified linear units (ReLU) as the activation function in each layer[2]. Note that the hyperparameter tuning of the network architecture is 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. From the generated training samples, 4878 training samples were used for training, 1045 training samples were used for validation, and 1045 training samples were used for testing.

Fig. 2 shows the closed loop simulation results using the approximate MPC control law $\pi_{\text{approx}}(x)$ (shown in red) compared with the traditional MPC control law $\pi_{\text{mpc}}(x)$ (shown in blue).

From this it can be seen that by using the proposed approach, we can choose a relatively larger grid size between the samples (sparse sampling), which reduces the number of optimization problems that needs to be solved offline. Consequently, the overall time and computational cost required to generate the training samples is significantly lesser.

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 [10], [24]. In our simulations, we model the heat dynamics of a building based on the modeling framework from [25], as shown below,

$$\begin{aligned}
\frac{\mathrm{d}T_s}{\mathrm{d}t} &= \frac{1}{R_{is}C_s}(T_i - T_s)\\
\frac{\mathrm{d}T_i}{\mathrm{d}t} &= \frac{1}{R_{is}C_i}(T_s - T_i) + \frac{1}{R_{ih}C_i}(T_h - T_i) + \frac{A_w\Phi}{C_i}\\
&\quad + \frac{1}{R_{ie}C_i}(T_e - T_i) + \frac{1}{R_{ia}C_i}(T_a - T_i)\\
\frac{\mathrm{d}T_h}{\mathrm{d}t} &= \frac{1}{R_{ih}C_h}(T_i - T_h) + \frac{u}{C_h}\\
\frac{\mathrm{d}T_e}{\mathrm{d}t} &= \frac{1}{R_{ie}C_e}(T_i - T_e) + \frac{1}{R_{ea}C_e}(T_a - T_e) + \frac{A_e\Phi}{C_e}
\end{aligned}$$

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 Φ 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 $x = [T_s, T_i, T_h, T_e]^\mathsf{T}$ with $T_{(*)} \in [12, 40]$ °C. The ambient temperature $T_a \in [-5, 20]$ °C and the solar irradiation $\Phi \in [0, 0.2]$ kW/m^2 are considered as external disturbances. The parameter values used in the model are shown in Table III.

The objective is to drive the interior temperature T_i to a desired setpoint $T_i^{sp} \in [18, 25]$ °C , while penalizing the rate of change of

[2]Source codes for the simulation results presented in this technical note can be found in this link

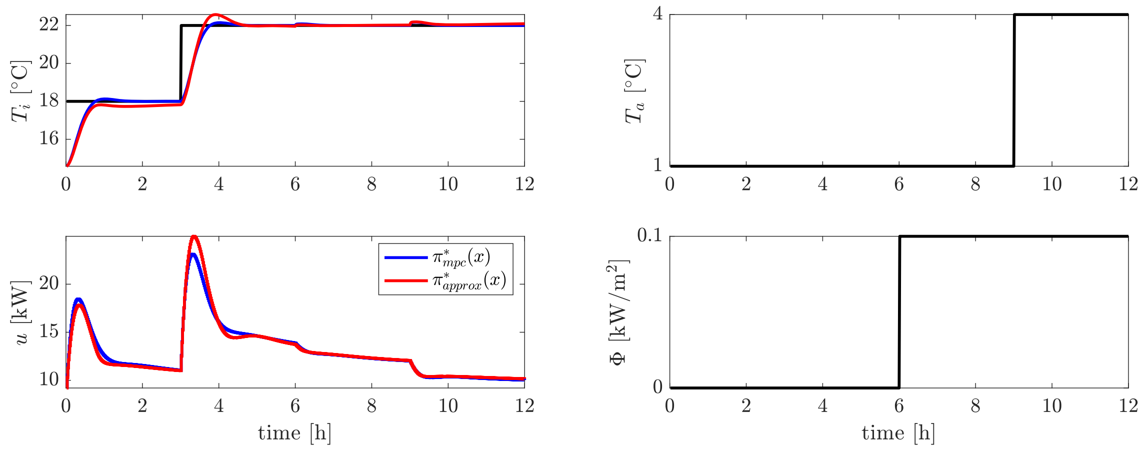

Fig. 3: Example B: Simulation results comparing the closed-loop performance of the traditional online MPC $\pi_{\text{mpc}}(\tilde{x})$, and the approximate explicit MPC $\pi_{\text{approx}}(\tilde{x})$.

TABLE II: Example B: CPU time in [s] for generating the training samples.

	min	avg	max	# of samples
Full NLP	0.3925	0.55	1.0683	330
Sensitivity update	0.029	0.0344	0.1034	6600

the input usage $u \in [0, 40]$ kW. The stage cost is then 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 control law $\pi_{\text{approx}}(\tilde{x})$. In this example $\tilde{x} = [T_s, T_i, T_h, T_e, T_i^{sp}, T_a, \Phi, u]^\mathsf{T}$, which requires sampling a 8-dimensional space in order to generate the training samples.

We randomly generate a total of 6930 samples, out of which, 330 samples were generated by solving the optimization problem and 6600 samples were generated using the sensitivity update. The average, minimum and maximum CPU time for solving the optimization problem and for computing the sensitivity update are shown in Table II.

Using the generated training samples, we approximate the MPC control law using deep neural networks with $L = 3$ layers and $M = 10$ neurons with rectified linear units (ReLU) as the activation function in each neuron. From the generated training samples, 4850 training samples were used for training, 1040 training samples were used for validation, and 1040 training samples were used for testing.

We test the performance of the approximate control law 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. 3 shows the closed loop simulation results using the traditional MPC control law $\pi_{\text{mpc}}(x)$ obtained by solving the MPC problem online, and the performance of the approximate explicit MPC control law $\pi_{\text{approx}}(x)$ approximated using the training samples generated using Algorithm 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.

TABLE III: Example 2: Parameter used in the building climate control problem.

R_{is}	Heat resistance between interior & sensor	1.89	°C/kW
R_{ih}	Heat resistance between interior & heater	0.146	°C/kW
R_{ie}	Heat resistance between interior & envelop	0.897	°C/kW
R_{ia}	Heat resistance between interior & ambient	2.5	°C/kW
R_{ea}	Heat resistance between envelop & ambient	0.146	°C/kW
C_s	Heat capacitance for the sensor	0.0549	kWh/°C
C_i	Heat capacitance for the interior	0.0928	kWh/°C
C_e	Heat capacitance for the envelop	3.32	kWh/°C
C_h	Heat capacitance for the heater	0.889	kWh/°C
A_e	Effective area of solar irradiation	3.87	m^2
A_w	Effective window area	5.75	m^2

V. Conclusions

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

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

The proposed scheme can be used with any MPC formulation that can be cast as a nonlinear programming problem.

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. Żak, 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] 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.

[14] L. Taylor and G. Nitschke, "Improving deep learning using generic data augmentation," *arXiv preprint arXiv:1708.06020*, 2017.

[15] A. V. Fiacco, "Sensitivity analysis for nonlinear programming using penalty methods," *Mathematical programming*, vol. 10, no. 1, pp. 287–311, 1976.

[16] J. F. Bonnans and A. Shapiro, "Optimization problems with perturbations: A guided tour," *SIAM review*, vol. 40, no. 2, pp. 228–264, 1998.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] S. Prívara, 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.

[25] 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.