2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Constrained Neural Networks for Approximate Nonlinear Model Predictive Control

Saket Adhau, Vihangkumar V. Naik, Sigurd Skogestad

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683320

***Abstract*— Solving Non-Linear Model Predictive Control (NMPC) online is often challenging due to the computational complexities involved. This issue can be avoided by approximating the optimization problem using supervised learning methods which comes with a trade-off on the optimality and/or constraint satisfaction. In this paper, a novel supervised learning framework for approximating NMPC is proposed, where we explicitly impart constraint knowledge within the neural networks. This knowledge is inherited by augmenting the loss function of the neural networks during the training phase with insights from KKT conditions. Logarithmic barrier functions are utilized to augment the loss function including conditions of primal and dual feasibility. The proposed framework can be applied to other machine learning based parametric approximators. This approach is easy to implement and its efficacy is demonstrated on a benchmark NMPC problem for continuous stirred tank reactor (CSTR).**

***Index Terms*— Neural Networks, nonlinear model predictive control, constrained optimization problems, constrained control.**

I. INTRODUCTION

Model Predictive Control (MPC) has widespread usage in industrial applications ranging from chemical plants, petroleum refineries, aerospace, and automotive domains [1]. However, the applicability of MPC, especially non-linear MPC (NMPC) is limited due to computational complexity involved in solving the associated nonconvex optimization problem within the stipulated sampling time. Problems of this class are being investigated by control as well as machine learning communities. Especially, by using machine learning tools such as Neural Networks to approximate the optimal control law given by MPC. Such approaches encounter challenges for industrial applications where the availability of necessary data could be limited, and synthesizing the available data could be challenging. Another underlying problem is that such approximated control laws often make a trade-off between performance and constraints satisfaction. Satisfying the constraints while approximating the control law can be of paramount importance and needs to be addressed.

Related Work:

1) Approximate MPC: A number of approaches to approximate optimal control law given by MPC using machine learning methods have been explored in literature. In these approaches, the control law is typically approximated using various parametric approximators often termed as imitation learning or supervised learning [2]–[4].

2) Constraint handling in Neural Networks: The inability to guarantee constraint satisfaction using these methods is one of the major limitations, contributing to active research in this direction. The authors in [5], provide guarantees on closed-loop constraint satisfaction and asymptotic stability. Projection of feasible control input on polytopes derived from maximal control invariant set of the system is presented in [6] and similar methods have also been shown in [7]. Furthermore, two separate networks for primal and dual problems are trained to estimate sub-optimality and probabilistic bounds are provided on the trained linear controller in [8]. In [9], the network is trained while adjusting the architecture until probabilistic bounds are satisfied. Additionally, instead of replacing the traditional MPC with approximate control law, neural networks have also been examined to assist the controller in warm starting, such as [10], [11]. The contribution in [12] present end to end learning of both system dynamics and control policies with constraint satisfaction capabilities.

Research work [13] dating back as early as $90's$, exhibit a class of neural networks for approximating general non-linear programming problems including equality and inequality constraints. The method is based on Lagrangian multipliers in optimization and provide solutions to satisfy the necessary conditions of optimality. The authors in [14] devise Lagrange type neural networks which can handle inequality constraints, whereas stability and convergence is discussed using Liapunov's approximation principle. The optimization aspects of imposing hard inequality constraints on Convolutional neural network (CNN), by using Log barrier functions along-with Lagrangian-dual optimization has been presented in [15]. In addition, [16]–[20] discuss methods from optimization theory especially KKT conditions to impose constraints on neural networks. To the best of the authors' knowledge, none of the ideas mentioned above have been studied in control domain for assisting in approximating optimal control laws.

In this paper, we present a novel constraint handling approach exploiting the domain knowledge from KKT conditions while approximating the NMPC problem. Typically, constraint handling in neural network based approximators is done at a later stage, only after the network is trained. Conversely, in our approach, primal and dual feasibility conditions are inherently passed on to the network during the training phase itself. This is achieved by augmenting the loss

This work was supported by the Research Council of Norway, under the IKTPLUSS program.

Saket Adhau and Sigurd Skogestad are with the Department of Chemical Engineering from Norwegian University of Science and Technology, 7491, Trondheim, Norway. {saket.adhau, sigurd.skogestad}@ntnu.no

Vihangkumar V. Naik is with ODYS S.r.l., Via A. Passaglia 185, Lucca, 55100, Italy. vihang.naik@odys.it

function with penalties on primal and dual feasibility. Hence, the overall idea is to introduce domain specific knowledge whilst training, to better leverage constraint characterization inside the network. We analyze performance of the proposed approach for approximating NMPC applied to a benchmark continuous stirred tank reactor (CSTR) problem. It is interesting to note that the proposed idea is applicable to approximate a wide range of constrained optimization problems.

The paper is organized as follows: Section II introduces background for neural networks. NMPC problem formulation, the conventional approach for NMPC approximation and the proposed approach are presented in Section III. Implementation of the proposed approach and results are discussed in Section IV while the paper concludes in Section V.

II. Background

This section introduces the loss function of neural networks to be utilized as a basis for the proposed framework. Let N be the number of training samples where, input matrix is denoted by $X = \{x_1, x_2, \ldots, x_N\}^\top \in \mathbb{R}^{N\times d}$, and output matrix is denoted by $Y = \{y_1, \ldots, y_N\}^\top \in \mathbb{R}^{N\times m}$ for the training data $\{x_i, y_i\}_{i=1}^N$. Input dimensions are denoted by d, whereas m denotes output dimensions. A fully connected feedforward network consisting of L layers indexed as $0, 1, 2, \ldots, L$ corresponding to input layer is considered. These hidden layers are composed of weight matrices $(W_k)_{k=1}^L \in \mathcal{W} := \mathbb{R}^{d\times n_1} \times \cdots \times \mathbb{R}^{n_{k-1}\times n_k} \times \cdots \times \mathbb{R}^{n_{L-1}\times m}$ where n_k denotes number of neurons on layer k, $n_0 = d$ and $n_L = m$ whereas the bias on the layers $(b_k)_{k=1}^L \in \mathcal{B} := \mathbb{R}^{n_1} \times \ldots \mathbb{R}^{n_L}$. We use nonlinear Rectified Linear Unit (ReLU), a non differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$. Let $\Psi \in \mathbb{R}^{N\times n_k}$ be the matrix containing feature vectors of layer k after application of activation function. Once the network architecture is designed, the Loss function $\Phi : \theta \to \mathbb{R}$ of the network can be defined as,

$$\Phi\left((W_k, b_k)_{k=1}^L\right) = \frac{1}{mN}\sum_{i=1}^{N}\sum_{j=1}^{m}\left((\Psi_{Lj}(x_i)) - y_{ij}\right). \quad (1)$$

In this paper, we consider, Mean Squared Error (MSE) as our loss function which is assumed to be continuously differentiable.

III. Neural network based approximate NMPC

Consider a discrete-time nonlinear system of the form,

$$x(t+1) = f(x(t), u(t)), \quad (2)$$

where the current states $x \in \mathcal{X}$, the control input $u \in \mathcal{U}$ are constrained to lie in compact sets of $\mathcal{X} \subset \mathbb{R}^{n_x}$ and $\mathcal{U} \subset \mathbb{R}^{n_u}$. The nominal model, $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is assumed to be twice differentiable in x and u.

The NMPC problem $\mathcal{P}(x(t))$ can be formulated as,

$$\min_{x(.),u(.)} \int_{t_0}^{t_0+T} \frac{1}{2}\left(||x(t) - x^{\text{ref}}(t)||_Q^2 + ||u(t) - u^{\text{ref}}(t)||_R^2\right) dt + ||x(t_0+T) - x^{\text{ref}}(t_0+T)||_P^2, \quad (3a)$$

subjected to following constraints for all $t \in [t_0, t_0+T]$

$$x(t_0) = \hat{x}_0, \quad (3b)$$
$$x(t+1) = f(x(t), u(t)), \quad (3c)$$
$$\underline{u} \le u(t) \le \overline{u}, \ \forall u \in \mathcal{U}, \quad (3d)$$
$$\underline{x} \le x(t) \le \overline{x}, \ \forall x \in \mathcal{X}. \quad (3e)$$

where, $T > 0$ is the prediction horizon, $x(t) \in \mathbb{R}^{n_x}$ and $u(t) \in \mathbb{R}^{n_u}$. The cost function is weighted by $Q \succeq 0, P \succeq 0$, and $R \succ 0$ and $\hat{x}_0$ is the current state estimate in (3b) at time t_0. The control input is bounded by $\underline{u}$ and $\overline{u}$ and states by $\underline{x}$ and $\overline{x}$, see [21], [22, Chapter 2]. The resulting NMPC control law is given by,

$$u^*(t) = \pi_{MPC}(x(t)). \quad (4)$$

Solving this problem online at each sampling time t is often computationally complex and may become impractical with increase in problem size.

A. Approximate NMPC

The problem in (3) can be approximated with neural networks using the loss function given by,

$$\mathcal{L}(x;\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\pi_{approx}(x_i;\theta) - u_i^*\right)^2. \quad (5)$$

In the above equation, $\pi_{approx}(x_i;\theta)$ is a trained policy such that $\pi_{approx} \approx \pi_{MPC}$ and u_i^* is the optimal control input or labeled dataset. The loss function $\mathcal{L}(x;\theta)$ is minimized w.r.t $\theta = \mathcal{W} \times \mathcal{B}$ which is an unconstrained optimization problem.

However, in general the nonlinear programming (NLP) problems arising in NMPC formulation contains constraints on u and/or x, such as on both in (3d) and (3e). The approximate control law π_{approx} may not mimic the behavior of the optimal policy accurately, often leading to a poor approximation, which results in constraints being not satisfied at all times. As a rule of thumb, the control input/s are saturated/clipped off for input constraint satisfaction [7], which may deteriorate the overall control performance. In order to derive a better approximation, we describe a novel supervised learning framework in the following sections.

B. Penalty methods and Logarithmic Barrier Functions

Penalty methods are often used to convert constrained optimization problem into an unconstrained optimization problem, by augmenting the objective function with a penalty when constraints are violated. However, penalty methods may not always guarantee constraint satisfaction and need precise tuning of the weight for each penalty term in the equation. If in a cost function, there are several constraints, penalty functions will not act as *barriers* at the boundaries of feasible region but rather be null. This may lead to oscillations and make the training unstable [15]. To overcome these drawbacks in penalty terms, we use Logarithmic barrier functions.

Consider the following NLP problem,

$$\min_{w} \ \psi(w), \tag{6a}$$
$$\text{s.t } g_i(w) \leq 0, \ \forall_i \in \{1, \ldots, m\}, \tag{6b}$$

where the strictly feasible region $\mathcal{F}^0$ is defined by,

$$\mathcal{F}^0 \equiv \{w \in \mathbb{R}^n | g_i(w) \leq 0\}, \ \forall_i \in \{1, \ldots, m\}, \tag{7}$$

and $\mathcal{F}^0$ is assumed to be non-empty. The logarithmic barrier function for the constraint, $g_i(w) \leq 0, \ \forall_i \in \{1, \ldots, m\}$, denoted by $\phi(w)$ is,

$$\phi(w) = \sum_{i \in \mathcal{I}} -\log(-g_i(w)), \tag{8}$$

where $\log(.)$ is natural logarithm. The value of the function approaches ∞ as w moves towards the edge of feasible region $\mathcal{F}^0$ implying when the constraints are violated, a high penalty of ∞ would be added to the cost function[1]. In the event of no constraint violation, the barrier term would be 0. The new objective function for the constrained optimization problem (6), when converted into an unconstrained optimization problem, can be written as,

$$\min_{w, \mu^k} \ \psi(w) - \mu^k \sum_{i=1}^{m} \log(-g_i(w)), \tag{9}$$

where, μ^k is a sequence of positive scalar barrier parameters, [23, Chapter 6].

C. Approximation of Constrained Optimization Problems

For approximating the NMPC problem (3), the main idea here is to introduce constraints (3d) and (3e) inside the neural networks during the training phase. This can be done by augmenting the loss function defined in (5), with (3d) and (3e) using logarithmic barrier function as in (9).

We start by introducing the input constraints (3d) in the loss functions (5) given by,

$$\mathcal{L}(x;\theta) = \frac{1}{N} \sum_{i=1}^{N} \Big(\Big\| \pi_{approx}(x_i;\theta) - u_i^* \Big\|_2^2 + \mu_{\underline{u}} \Big\| f_b(\hat{u}_i, \underline{u}_i) \Big\|_2^2 + \ \mu_{\overline{u}} \Big\| f_b(\hat{u}_i, \overline{u}_i) \Big\|_2^2 \Big),$$

where, $\mu_{\underline{u}}, \mu_{\overline{u}} > 0$ are the barrier parameters. The function f_b for lower and upper bound is defined as per (8),

$$f_b(\hat{u}_i, \underline{u}_i) : \ \underline{u}_i - \hat{u}_i \leq 0 : \ -\log(\hat{u}_i - \underline{u}_i), \tag{10a}$$
$$f_b(\hat{u}_i, \overline{u}_i) : \ \hat{u}_i - \overline{u}_i \leq 0 : \ -\log(\overline{u}_i - \hat{u}_i). \tag{10b}$$

Furthermore, to include state constraints (3e), a new policy $\hat{\pi}_x$ is required to be defined which will not only predict the optimal control inputs but also the next states,

$$\begin{bmatrix} \hat{u} & \hat{x}^+ \end{bmatrix}^\top = \hat{\pi}_x(x_i;\theta). \tag{11}$$

[1]In case of neural networks, where the gradient of ∞ cannot be evaluated, a penalty of high value is imposed.

The resulting loss function is formulated as,

$$\mathcal{L}_x(x;\theta) = \frac{1}{N} \sum_{i=1}^{N} \Bigg(\Big\| \hat{\pi}_x(x_i;\theta) - z_{x_i}^* \Big\|_2^2 + \mu_{\underline{u}} \Big\| f_b(\hat{u}_i, \underline{u}_i) \Big\|_2^2 + \ \mu_{\overline{u}} \Big\| f_b(\hat{u}_i, \overline{u}_i) \Big\|_2^2 + \mu_{\underline{x}} \Big\| f_b(\hat{x}_i, \underline{x}_i) \Big\|_2^2 + \ \mu_{\overline{x}} \Big\| f_b(\hat{x}_i, \overline{x}_i) \Big\|_2^2 \Bigg), \tag{12}$$

where, $\mu_{\underline{x}}, \mu_{\overline{x}} > 0$, vector z_x^* consists of $[u^* \ x^+]^\top$ calculated by solving the NMPC problem $\mathcal{P}(x)$.

To better leverage constraint characterization inside the network, we also exploit the insights from KKT conditions, specifically the dual feasibility. In order to do this, for the sake of simplicity of notations, we rewrite inequality constraints in (3d) & (3e) and the dual variables associated with the inequality constraints as, $\mathcal{A} = [-u + \underline{u}, u - \overline{u}, -x + \underline{x}, x - \overline{x}]^\top \leq 0$, $\lambda = [\lambda_{\underline{u}}, \lambda_{\overline{u}}, \lambda_{\underline{x}}, \lambda_{\overline{x}}]^\top \geq 0$ respectively where, $\lambda \in \mathbb{R}^{2n_x + 2n_u}$. For including dual feasibility $\lambda \geq 0$, the function f_λ to be used in loss function is defined similar to (10) as,

$$f_\lambda(\hat{\lambda}_i) : \ -\hat{\lambda}_i \leq 0 : \ -\log(\hat{\lambda}_i).$$

Next, we define a new policy $\hat{\pi}_\lambda(x_i;\theta)$, such that it predicts dual variables $\hat{\lambda}$ along-with predicted inputs $\hat{u}$ and next states $\hat{x}^+$ as in (11). Finally, we write the proposed loss function as,

$$\mathcal{L}_\lambda(x;\theta) = \frac{1}{N} \sum_{i=1}^{N} \Bigg(\Big\| \hat{\pi}_\lambda(x_i;\theta) - z_{\lambda_i}^* \Big\|_2^2 + \mu_{\underline{u}} \Big\| f_b(\hat{u}_i, \underline{u}_i) \Big\|_2^2 + \ \mu_{\overline{u}} \Big\| f_b(\hat{u}_i, \overline{u}_i) \Big\|_2^2 + \mu_{\underline{x}} \Big\| f_b(\hat{x}_i, \underline{x}_i) \Big\|_2^2 + \ \mu_{\overline{x}} \Big\| f_b(\hat{x}_i, \overline{x}_i) \Big\|_2^2 + \mu_\lambda \Big\| f_\lambda(\hat{\lambda}_i) \Big\|_2^2 \Bigg), \tag{13}$$

where, $\mu_\lambda > 0$ and the vector $z_{\lambda_i}^*$ consists $[u^* \ x^+ \ \lambda^*]^\top$. Using dual variables, the idea is to ensure constraint satisfaction when feasible solution exists and help the network optimize its policy-parameters θ, while inheriting constraint knowledge. The proposed general idea for approximating constrained optimization problems is illustrated in Fig. 1.

The barrier terms are only active in the loss function when constraints are violated, reducing the excessive gradient ascent iterations and making the training less computationally expensive [15].

The detailed procedure for the proposed approach is summarized in Algorithm 1. We start with a formulated NMPC problem $\mathcal{P}(x)$, a null dataset $\mathcal{D}$ for collecting the data required and a feasible state space $\mathcal{X}$. For each instance $i = 1, \ldots, N$, the NMPC problem is solved by randomly choosing x_i from the feasible state space $\mathcal{X}$ and the dataset $\mathcal{D}$ is updated with $x_i, u_i^*, x_i^+, \lambda_i^*$. Eventually, once N samples are processed, the network is trained using the cost function described in (13).

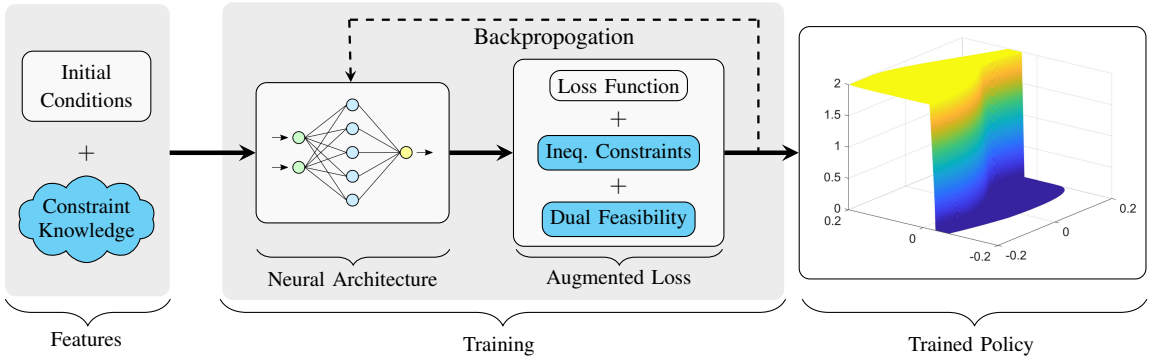

Fig. 1. Illustration of the proposed methodology for injecting constraint knowledge during the training phase for approximating constrained optimization problems.

Algorithm 1 Learning of constrained approximate NMPC.

Input: NMPC problem $\mathcal{P}(x)$, feasible set $\mathcal{X}$, null dataset $\mathcal{D}$

1: **for** $i = 1$ **to** N **do**
2: Sample $x_i \in \mathcal{X}$
3: Collect u_i^* and x_i^+ for every x_i by solving $\mathcal{P}(x_i)$ and λ_i^* for every inequality constraint
4: Update the dataset, $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_i, u_i^*, x_i^+, \lambda_i^*)\}$
5: **end for**
6: $\theta \leftarrow$ (13)

Output: $\hat{\pi}_\lambda(x_i;\theta)$.

IV. Simulation Results

As an illustrative example, we consider the Continuous Stirred Tank Reactor (CSTR) benchmark system [24] given by,

$$\begin{aligned} \dot{x_1} &= \tfrac{1}{\beta}(1 - x_1) - c x_1 e^{-\frac{S}{x_2}}, \\ \dot{x_2} &= \tfrac{1}{\beta}(x_f - x_2) + c x_1 e^{-\frac{S}{x_2}} - \alpha u (x_2 - x_c), \end{aligned} \tag{14}$$

where the states are product concentration x_1 and the reactant coolant temperature x_2. The feasible state space is given by $\mathcal{X} = [0.0632, 0.4632] \times [0.4519, 0.8519]$ and the coolant flow input rate u is characterized as $\mathcal{U} = [-0.7853, 1.2147]$.

The system parameters considered are $\beta = 20$, $c = 300$, $S = 5$, $x_f = 0.3947$, $x_c = 0.3816$ and $\alpha = 0.117$. The control objective is to take the system from a locally stable steady state to a locally unstable steady state point, $x^{\text{ref}} = [0.2632, 0.6519]^\top$ which makes the problem challenging. This control problem is solved using the formulation in (3). The initial conditions for NMPC formulation are $x_0 = [0.9831, 0.3918]^\top$ with a prediction horizon of $T = 60$ and sampling time of $t = 0.5s$.

Step I: Generating data for Neural Networks: We use grid-based sampling approach as seen in [9]. An uniform grid of size $(8 \cdot 10^{-3})$ consisting of $(1.6 \cdot 10^5)$ data points is created out of which around $(1.4 \cdot 10^5)$ data points were identified as feasible. Detailed procedure for generating training data as required by the proposed method is given in Algorithm 1. NMPC problem $\mathcal{P}(x)$ from (3) was solved for all $x_i \in \mathcal{X}$ using CasADi v3.5.5 for Python [25].

Step II: Learning: For comparison, we train 3 different policies namely, (i) supervised learning approach $\pi_{approx}(x_i;\theta)$ in (5); (ii) log barrier penalty function for input and state constraints $\hat{\pi}_x(x_i;\theta)$ as in (12); (iii) the proposed method $\hat{\pi}_\lambda(x_i;\theta)$ in (13).

For training of the above mentioned policies, a feedforward neural network consisting of two hidden layers of 160 and 1560 neurons respectively were used. ReLU and linear activation functions were used with Adam optimizer. The neural network architecture was implemented in Keras v2.3.0 for Python with a learning rate of $1 \cdot 10^{-3}$, batch size of 500 and 512 epochs. The barrier parameters used in the cost function are $\mu_{\underline{u}} = \mu_{\overline{u}} = 0.1$, $\mu_{\underline{x}} = \mu_{\overline{x}} = 0.1$ and $\mu_\lambda = 0.01$.

All the simulations in this paper including training of the policies, have been done on a MacBook Pro with Intel Core i7, 2.6GHz processor and 16GB of memory. The time required for training the policies is reported in Table. I. It is also observed that training time for the proposed approach $\hat{\pi}_\lambda(x_i;\theta)$ is only marginally increased.

TABLE I
Comparison of Training Times for Various Policies.

Policy	$\pi_{approx}(x_i;\theta)$	$\hat{\pi}_x(x_i;\theta)$	$\hat{\pi}_\lambda(x_i;\theta)$
Time [s]	103.54	106.22	108.58

A. Result Analysis

We use the standard NMPC results as primary reference and compare with three approaches described in Step II of Section IV. Figs. 2–4 show the experimental results of state evaluation for x_1, x_2 and control input u for a benchmark CSTR problem.

The results in Fig. 2 show that, the most commonly used method of the control input generated by the policy $\pi_{approx}(x_i;\theta)$ results in violation of constraints imposed on the coolant flow rate input. Secondly, we observed that the results denoted by $\hat{\pi}_x(x_i;\theta)$, demonstrate how the inclusion of input and state constraint penalties as described in the loss function (12), are not sufficient for constraints satisfaction. Though as a convention, adding saturation to the control input for the policies $\pi_{approx}(x_i;\theta)$ and $\hat{\pi}_x(x_i;\theta)$ would satisfy the constraints on coolant flow rate input, the control performance gets degraded as demonstrated in Fig. 3.

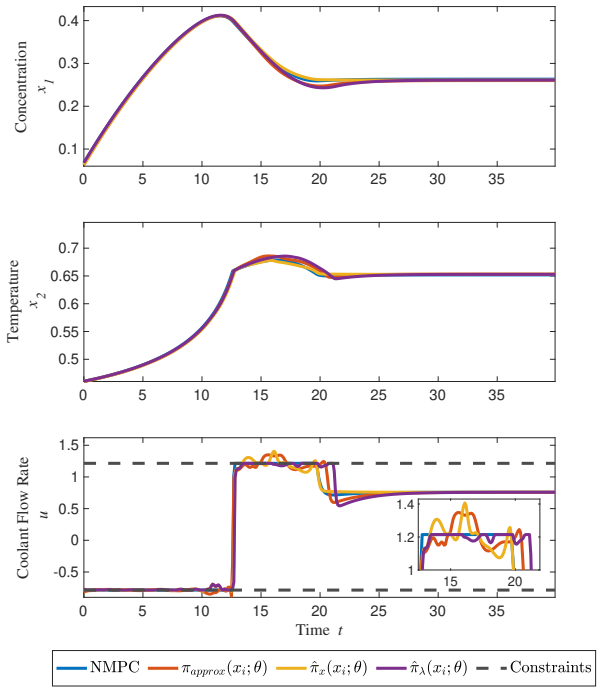

Fig. 2. CSTR benchmark: Performance comparison of states x_1, x_2 and control input u using various approaches *v.i.z* traditional supervised learning approach $\pi_{approx}(x_i;\theta)$; penalty for input and state constraints $\hat{\pi}_x(x_i;\theta)$ and the proposed method $\hat{\pi}_\lambda(x_i;\theta)$.

Finally, the results of $\hat{\pi}_\lambda(x_i;\theta)$ are same in Fig. 2 and 3 for illustration purposes, which denotes performance of the

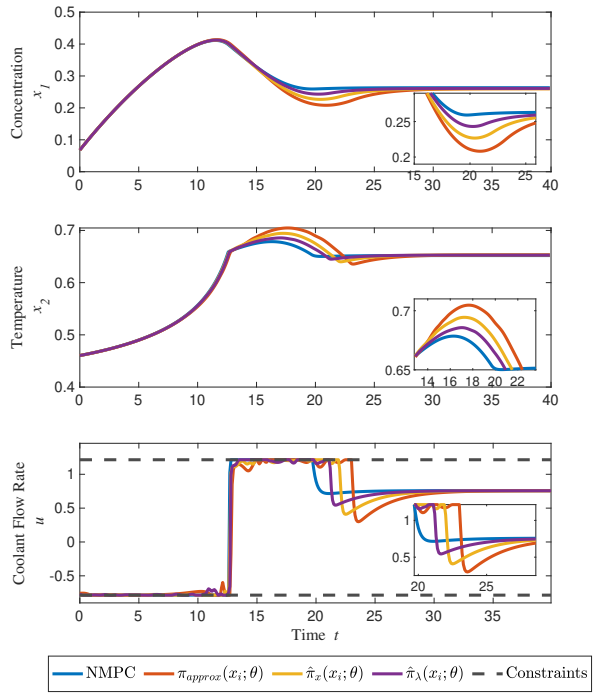

Fig. 3. CSTR benchmark: Performance comparison for states x_1, x_2 and control input u using various approaches *v.i.z* traditional supervised learning approach $\pi_{approx}(x_i;\theta)$ with clipping of control input; penalty for input and state constraints $\hat{\pi}_x(x_i;\theta)$ with input clipped and the proposed method $\hat{\pi}_\lambda(x_i;\theta)$ (without clipping of control input).

proposed method including the dual variables, characterized by (13). From Fig. 3, it can be concluded that using $\hat{\pi}_\lambda(x_i;\theta)$ there are no input constraint violation (without saturating the control input) and it renders the best performance in terms of state evaluation against standard NMPC approach.

As observed in Fig. 4, the policies $\pi_{approx}(x_i;\theta)$ and $\hat{\pi}_x(x_i;\theta)$ violate the constraints for state x_2. Whereas using the proposed approach, state constraints are always ensured, even with a different initial states as compared to the former figures, which is further illustrated in Fig. 5. Fig 5 shows

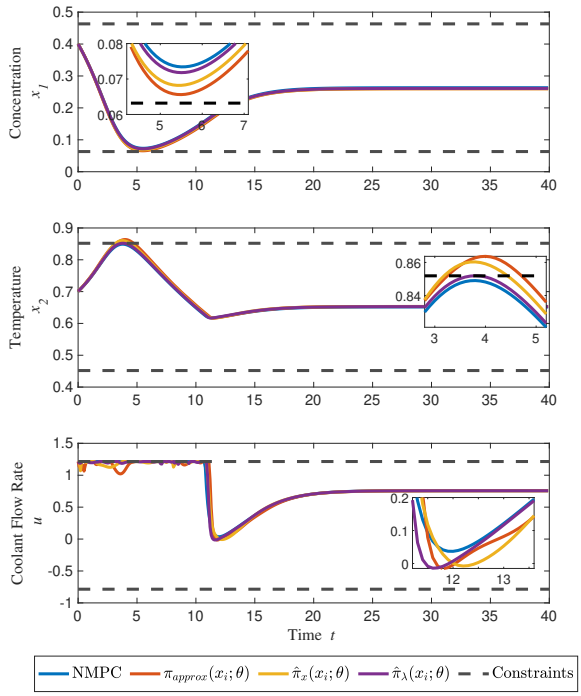

Fig. 4. Demonstration of state constraint satisfaction for CSTR benchmark: traditional supervised learning approach $\pi_{approx}(x_i;\theta)$ with clipping of control input; penalty for input and state constraints $\hat{\pi}_x(x_i;\theta)$ with clipping of control input and the proposed method $\hat{\pi}_\lambda(x_i;\theta)$ (without clipping of control input).

evolution of states x_1 *vs* x_2 inside the feasible region $\mathcal{X}$ considering different initial states for the NMPC as well as the proposed policy $\hat{\pi}_\lambda(x_i;\theta)$. For each of the initial conditions considered, it can be observed that the proposed approach is able to mimic the NMPC trajectories without any constraint violation.

Furthermore, the proposed method is evaluated for 5000 different trajectories. The resulting values of key performance indices (KPIs) compared with standard NMPC and maximum state constraint violations are reported in Table. II. The approximations rendered by the proposed approach $\hat{\pi}_\lambda(x_i;\theta)$, outperforms standard approach $\pi_{approx}(x_i;\theta)$ as well as $\hat{\pi}_x(x_i;\theta)$.

V. CONCLUSION

In this paper, we proposed simple yet effective approach for constraint handling for approximating nonlinear MPC problems using neural networks. The basic idea of including the domain knowledge pertaining to the underlying nonlinear optimization problem while training the neural network has

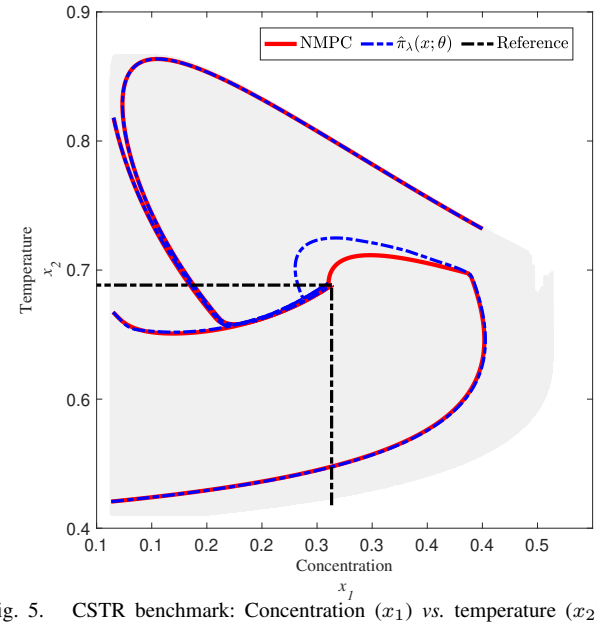

Fig. 5. CSTR benchmark: Concentration (x_1) *vs.* temperature (x_2) for randomly initiated evolution of states using proposed method against NMPC, black dashed line represents reference values.

TABLE II

COMPARISON OF KPIS WITH NMPC FOR OVER 5000 TRAJECTORIES

KPI	$\pi_{approx}(x_i;\theta)$	$\hat{\pi}_x(x_i;\theta)$	$\hat{\pi}_\lambda(x_i;\theta)$
Comparison for states			
ISE	$0.148 \cdot 10^{-6}$	$8.71 \cdot 10^{-6}$	$2.56 \cdot 10^{-6}$
IAE	0.8857	0.5017	0.1528
ITSE	0.3821	0.2914	0.1566
Comparison for control input			
ISCE	$3.02 \cdot 10^{-5}$	$1.35 \cdot 10^{-5}$	$0.142 \cdot 10^{-5}$
Maximum state constraint violation			
	4.4231	3.2783	0.7739

been explored. Specifically, the intuitive insights of KKT conditions (primal and dual feasibility) and log barrier methods are utilized to modify the loss function of the neural networks. Thorough numerical simulations on a benchmark CSTR problem have demonstrated the ability to explicitly handle constraints on both, input as well as states. The proposed framework rendered improved performance in terms of input and state evaluation compared to the conventional neural network based method. This approach is not limited to NMPC but can be used to approximate a wide range of constrained optimization problems.

REFERENCES

[1] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, pp. 733–764, 2003.

[2] S. Lucia and B. Karg, "A deep learning-based approach to robust nonlinear model predictive control," in *6th IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2018, pp. 511–516.

[3] S. Lucia, D. Navarro, B. Karg, H. Sarnago, and O. Lucia, "Deep learning-based model predictive control for resodat power converters," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 409–420, 2020.

[4] 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.

[5] B. Karg and S. Lucia, "Stability and feasibility of neural network-based controllers via output range analysis," in *59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 4947–4954.

[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)*, 2018, pp. 1520–1527.

[7] J. A. Paulson and A. Mesbah, "Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction," *IEEE Control Systems Letters*, vol. 4, pp. 719–724, 2020.

[8] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari, "Large scale model predictive control with neural networks and primal active sets," *arXiv preprint arXiv:1910.10835*, 2019.

[9] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, pp. 543–548, 2018.

[10] M. Klaučo, M. Kalúz, and M. Kvasnica, "Machine learning-based warm starting of active set methods in embedded model predictive control," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 1–8, 2019.

[11] Y. Vaupel, N. C. Hamacher, A. Caspari, A. Mhamdi, I. G. Kevrekidis, and A. Mitsos, "Accelerating nonlinear model predictive control through machine learning," *Journal of Process Control*, vol. 92, pp. 261–270, 2020.

[12] J. Drgona, K. Kis, A. Tuor, D. Vrabie, and M. Klauco, "Differentiable predictive control: An MPC alternative for unknown nonlinear systems using constrained deep learning," *arXiv preprint arXiv:2011.03699*, 2020.

[13] S. Zhang and A. Constantinides, "Lagrange programming neural networks," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, pp. 441–452, 1992.

[14] Y. Huang, "Lagrange-type neural networks for nonlinear programming problems with inequality constraints," in *44th IEEE Conference on Decision and Control (CDC)*, 2005, pp. 4129–4133.

[15] H. Kervadec, J. Dolz, J. Yuan, C. Desrosiers, E. Granger, and I. B. Ayed, "Constrained deep networks: Lagrangian optimization via log-barrier extensions," *arXiv preprint arXiv:1904.04205*, 2019.

[16] P. Márquez-Neila, M. Salzmann, and P. Fua, "Imposing hard constraints on deep networks: Promises and limitations," *arXiv preprint arXiv:1706.02025*, 2017.

[17] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, pp. 25–57, 2006.

[18] Y. Nandwani, A. Pathak, P. Singla *et al.*, "A primal dual formulation for deep learning with constraints," *Advances in Neural Information Processing Systems*, pp. 12 157–12 168, 2019.

[19] S. Shalev-shwartz and S. M. Kakade, "Mind the duality gap: Logarithmic regret algorithms for online optimization," *Advances in Neural Information Processing Systems*, vol. 21, 2009.

[20] Q. Nguyen and M. Hein, "The loss surface of deep and wide neural networks," in *34th International Conference on Machine Learning*, 2017, pp. 2603–2612.

[21] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl, "Auto-generated algorithms for nonlinear model predictive control on long and on short horizons," in *52nd IEEE Conference on Decision and Control (CDC)*, 2013, pp. 5113–5118.

[22] A. Grancharova and T. A. Johansen, *Explicit nonlinear model predictive control: Theory and applications*. Springer Science & Business Media, 2012, vol. 429.

[23] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. Society for Industrial and Applied Mathematics, 2010.

[24] D. Q. Mayne and E. C. Kerrigan, "Tube-based robust nonlinear model predictive control1," in *7th IFAC Symposium on Nonlinear Control Systems*, 2007, pp. 36–41.

[25] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, pp. 1–36, 2019.