Constrained Neural Networks for Approximate Nonlinear Model Predictive Control

Saket Adhau, Vihangkumar V. Naik, Sigurd Skogestad

***Abstract*— Solving Non-Linear Model Predictive Control (NMPC) online is often challenging due to the computational complexities involved. This problem 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 cost function of the neural networks during the training phase with insights from KKT conditions. Logarithmic barrier functions are utilized to augment the cost 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, log barrier functions.**

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 the control as well as machine learning communities, by using machine learning tools especially 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 NN: The inability to guarantee constraint satisfaction using these methods is one of the major limitation, 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 by augmenting its loss function during the training phase. Such feasibility conditions in terms of inequality constraints are penalized in the objective function using well-known log

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, 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

Manuscript 1135 submitted to 2021 60th IEEE Conference on Decision and Control (CDC). Received March 25, 2021.

barrier functions [21]. Hence, the overall idea is to introduce domain specific knowledge whilst training with an aim 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.

The paper is organized as follows: Section II introduces the theory for neural networks with its cost function. NMPC problem formulation, the conventional approach for approximation of NMPC and the proposed approach is presented in Section III. Implementation of the proposed approach and the 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 $X = \{x_1, x_2, \ldots, x_N\}^T \in \mathbb{R}^{N\times d}$, and output matrix is denoted by $Y = \{y_1, \ldots, y_N\}^T \in \mathbb{R}^{N\times m}$ 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 $(b_k)_{k=1}^L \in \mathcal{B} := \mathbb{R}^{n_1} \times \ldots \mathbb{R}^{n_L}$.

Now we have $\theta = \mathcal{W}\times\mathcal{B}$ which is set of all the parameters in the network. In this paper, we use nonlinear Rectified Linear Unit (ReLU), a non differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$. Let $f_k, g_k : \mathbb{R}^d \to \mathbb{R}^{n_k}$ map the inputs states to the feature space at layer k as,

$$f_0(x) = x, f_k(x) = \sigma(max(0, g_k(x))), \tag{1}$$

$$g_k(x) = W_k^T f_{k-1}(x) + b_k, \tag{2}$$

for every $x \in \mathbb{R}^d$ and $k \in [L]$. The activation function ReLU is calculated as element wise maximum between 0 and the function $g(x)$. Let $G_k = [g_k(x_1), g_k(x_2), \ldots, g_k(X_N)]^T \in \mathbb{R}^{N\times n_k}$ be the matrix containing feature vectors of layer k before applying activation function and $F_k = [f_k(x_1), f_k(x_2), \ldots, f_k(X_N)]^T \in \mathbb{R}^{N\times n_k}$ after application of activation function,

$$F_k = \begin{cases} 1 & \text{if } \; g_k(x) \geq 0, \\ 0 & \text{else.} \end{cases} \tag{3}$$

Once the network architecture is designed, the final objective $\Phi : \theta \to \mathbb{R}$ of the network is,

$$\Phi\Big((W_k, b_k)_{k=1}^L\Big) = \frac{1}{mN}\sum_{i=1}^{N}\sum_{j=1}^{m}\Big((f_{Lj}(x_i)) - y_{ij}\Big) \tag{4}$$

In this paper, we consider, Mean Squared Error (MSE) as our loss function which is assumed to be continuously differentiable. The network parameters θ, are optimized using an optimizer to minimize the unconstrained loss function [20].

III. Neural network based approximate NMPC

Consider a discrete-time nonlinear system of the form,

$$x(t+1) = f(x(t), u(t)), \tag{5}$$

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 a constrained finite horizon optimal control problem (CFHOCP) given by,

$$\min_{x(.),u(.)} \int_{t_0}^{t_0+T} \frac{1}{2}\left(||x(t) - x^{\text{ref}}(t)||_Q^2 + ||u(t) - u^{\text{ref}}(t)||_R^2\right) dt + ||x(t_0+T) - x^{\text{ref}}(t_0+T)||_P^2, \tag{6a}$$

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

$$x(t_0) = \hat{x}_0, \tag{6b}$$

$$\dot{x}(t+1) = f(x(t), u(t)), \tag{6c}$$

$$\underline{u} \leq u(t) \leq \overline{u}, \; \forall u \in \mathcal{U}, \tag{6d}$$

$$\underline{x} \leq x(t) \leq \overline{x}, \; \forall x \in \mathcal{X}. \tag{6e}$$

The term (6a), is the cost function while $T > 0$ is the prediction horizon. The cost function is weighted by $Q \succeq 0, P \succeq 0$, and $R \succ 0$. The function f in (6c) describes the nonlinear dynamics of the system in ordinary differential equations (ODE) form which depends on the input signal $u(t) \in \mathcal{U}$. The $\hat{x}_0$ is the current state estimate in (6b) at time t_0. The control input is bounded by $\underline{u}$ and $\overline{u}$ and states by by $\underline{x}$ and $\overline{x}$. The time varying states and control references are x^{ref} and u^{ref} respectively, see [22], [23, Chapter 2].

The CFHOCP in (6a) is solved for minimizing the stage cost at each sampling time t using the initial condition as $x(t)$. The first optimal input is fed to the plant from the sequence of optimal control inputs, $(U_N^* = u_0^*, \ldots, u_N - 1^*)$ over the prediction horizon N, giving the NMPC control law as,

$$u^*(t) = \pi_{MPC}(x(t)). \tag{7}$$

Solving this problem online is often computationally complex and may become impractical with increase in problem size.

A. Approximate NMPC

The problem in (6) can be approximated with neural networks using the loss function given by,

$$\mathcal{L}(x;\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\pi_{approx}(x_i;\theta) - u_i^*\right)^2 \tag{8}$$

In the above equation, $\hat{u}_i = \pi_{approx}(x_i;\theta)$ 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 $\mathcal{L}(x;\theta)$, would be minimized w.r.t θ which is an unconstrained optimization problem.

However, in general the NLP problem arising in NMPC formulation contains constraints on x and/or u, such as on

Manuscript 1135 submitted to 2021 60th IEEE Conference on Decision and Control (CDC). Received March 25, 2021.

both in (6d) and (6e). The approximate control law π_{approx} does not have any theoretical/practical guarantees regarding constraint satisfaction. Generally as a rule of thumb, the control input $\hat{u}$ is saturated/clipped off for input constraint satisfaction [7], which deteriorates the control performance. In order to address this issue, we describe a supervised learning framework to inherit knowledge of input as well as state constraints inside the NN 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,

$$\min_{w} \; f(w), \tag{9a}$$

$$\text{s.t } g_i(w) \leq 0, \; \forall_i \in \{1, \ldots, m\}, \tag{9b}$$

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

$$\mathcal{F}^0 \equiv \{w \in \mathbb{R}^n | g_i(w) \leq 0, \; \forall_i \in \{1, \ldots, m\}, \tag{10}$$

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,

$$\phi(w) = \sum_{i \in \mathcal{I}} -\log(-g_i(w)), \tag{11}$$

where $\log(.)$ is natural logarithm and the barrier function have the following properties [24],

1. The barrier function is smooth inside $\mathcal{F}^0$,
2. infinite everywhere except in $\mathcal{F}^0$,
3. the value of the function approaches ∞ as w moves towards the edge of feasible region, $\mathcal{F}^0$

and,

$$\phi(w) = \begin{cases} 0 & \text{if } \; g_i(w) \leq 0, \\ \infty & \text{if } \; g_i(w) \geq 0. \end{cases} \tag{12}$$

which means, when the constraints are violated, a high penalty of ∞ would be added to the cost function. In the event of no constraint violation, the barrier term would be 0 and the problem would be solved as an unconstrained optimization problem. The new objective function for the constrained optimization problem (9), when converted into an unconstrained optimization problem, can be written as,

$$\min_{w, \mu^k} \; f(w) - \mu^k \sum_{i=1}^{m} \log(-g_i(w)), \tag{13}$$

where, μ^k is a sequence of positive scalar barrier parameters, [21, Chapter 6].

C. Constrained Approximate NMPC

For approximating the NMPC problem with constraints, the main idea here is to introduce (6d) and (6e) inside the neural networks during the training phase. We propose to augment the loss function (8) with logarithmic barrier terms as in (13). Since the predicted output from the policy $\pi_{approx}(x_i; \theta)$ can only predict $\hat{u}$, the constraint in (6e) cannot be considered in (8) in a straight forward manner. Hence, we start by introducing the constraints on input in the loss functions (8) given by,

$$\mathcal{L}(x; \theta) = \frac{1}{N} \sum_{i=1}^{N} \Big(\Big\| \pi_{approx}(x_i; \theta) - u_i^* \Big\|_2^2 + \mu_{\underline{u}} \Big\| f_b(\hat{u}_i, \underline{u}_i) \Big\|_2^2 + \; \mu_{\overline{u}} \Big\| f_b(\hat{u}_i, \overline{u}_i) \Big\|_2^2 \Big), \tag{14}$$

where, $\mu_{\underline{u}}, \mu_{\overline{u}} > 0$ are the barrier parameter. The function f_b for lower and upper bound is defined as per (11),

$$f_b(\hat{u}_i, \underline{u}_i) : \; \underline{u}_i - \hat{u}_i \leq 0 : \; \; -\log(\hat{u}_i - \underline{u}_i), \tag{15a}$$

$$f_b(\hat{u}_i, \overline{u}_i) : \; \hat{u}_i - \overline{u}_i \leq 0 : \; \; -\log(\overline{u}_i - \hat{u}_i). \tag{15b}$$

Furthermore, to include state constraints from (6e), 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,

$$\begin{bmatrix} \hat{u} & \hat{x}^+ \end{bmatrix}^\top = \hat{\pi}_x(x_i; \theta) \tag{16}$$

The resulting loss function is formulated as,

$$\mathcal{L}_x(x; \theta) = \frac{1}{N} \sum_{i=1}^{N} \Bigg(\Big\| \hat{\pi}_x(x; \theta) - z_{x_i}^* \Big\|_2^2 + \mu_{\underline{u}} \Big\| f_b(\hat{u}_i, \underline{u}_i) \Big\|_2^2 + \; \mu_{\overline{u}} \Big\| f_b(\hat{u}_i, \overline{u}_i) \Big\|_2^2 + \mu_{\underline{x}} \Big\| f_b(\hat{x}_i, \underline{x}_i) \Big\|_2^2 + \; \mu_{\overline{x}} \Big\| f_b(\hat{x}_i, \overline{x}_i) \Big\|_2^2 \Bigg), \tag{17}$$

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

In the loss functions (14) and (17), the constraints are handled in a "soft" manner by augmenting the loss function with a high penalty when the constraints are being violated. Since the barrier terms are only active in the loss function when constraints are violated, excessive gradient ascent iterations are reduced, making the training less computationally expensive [15].

Augmenting the loss function with state and input constraints alone, is not sufficient to satisfy constraints. To ensure constraint satisfaction, we also exploit the insights from KKT conditions, specifically the dual feasibility. For the sake of simplicity of notations, we rewrite inequality constraints in (6d) & (6e) and the dual variables associated with the inequality constraints as,

$$\mathcal{A} = \begin{bmatrix} -u + \underline{u} \\ u - \overline{u} \\ -x + \underline{x} \\ x - \overline{x} \end{bmatrix} \leq 0, \; \lambda = \begin{bmatrix} \lambda_{\underline{u}} \\ \lambda_{\overline{u}} \\ \lambda_{\underline{x}} \\ \lambda_{\overline{x}} \end{bmatrix} \; \text{and } \lambda \geq 0, \tag{18}$$

Manuscript 1135 submitted to 2021 60th IEEE Conference on Decision and Control (CDC). Received March 25, 2021.

where, $\lambda \in \mathbb{R}^{2n_x+2n_u}$. From (17), the primal feasibility is enforced using log barrier functions. Similarly, to enforce dual feasibility $\lambda_i \geq 0$, we can add this additional constraint by augmenting loss function as,

$$\mathcal{L}_\lambda(x;\theta) = \frac{1}{N}\sum_{i=1}^{N}\Bigg(\Big\|\hat{\pi}_\lambda(x;\theta) - z^*_{\lambda_i}\Big\|_2^2 + \mu_{\underline{u}}\Big\|f_b(\hat{u}_i,\underline{u}_i)\Big\|_2^2 + \mu_{\overline{u}}\Big\|f_b(\hat{u}_i,\overline{u}_i)\Big\|_2^2 + \mu_{\underline{x}}\Big\|f_b(\hat{x}_i,\underline{x}_i)\Big\|_2^2 + \mu_{\overline{x}}\Big\|f_b(\hat{x}_i,\overline{x}_i)\Big\|_2^2 + \mu_\lambda\Big\|f_\lambda(\hat{\lambda}_i)\Big\|_2^2\Bigg), \tag{19}$$

where, $\mu_\lambda > 0$ and the vector $z^*_{\lambda_i}$ consists $[u^*\ x^+\ \lambda^*]^\top$ and the proposed policy $\hat{\pi}_\lambda(x_i;\theta)$ is defined such that it predicts lagrangian multiplier, $\hat{\lambda}$ along-with next states $\hat{x}^+$ as in (16). Furthermore, similar to (15) the function f_λ is defined as,

$$f_\lambda(\hat{\lambda}_i): \ -\hat{\lambda}_i \leq 0: \ -\log(\hat{\lambda}_i). \tag{20}$$

Using dual variables, the idea is to ensure constraint satisfaction when feasible solution exists, act as a barrier when constraints are satisfied and help the network optimize its hyper-parameters θ, while inheriting constraint knowledge.

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 $u_i^*, x_i^+, \lambda_i^*$. Eventually, once N samples are processed, the network is trained using the cost function described in (19).

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: $\hat{\theta} \leftarrow$ (19)

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

IV. Simulation Results

As an illustrative example, we consider the Continuous Stirred Tank Reactor (CSTR) benchmark system [25] given by,

$$\begin{aligned}\dot{x_1} &= (1/\theta)(1-x_1) - kx_1e^{-(M/x_2)},\\ \dot{x_2} &= (1/\theta)(x_f - x_2) + kx_1e^{-(M/x_2)} - \alpha u(x_2 - x_c),\end{aligned} \tag{21}$$

where the two states, product concentration x_1, and the reactant coolant temperature x_2, are constrained as,

$$\begin{aligned}0.0632 \leq x_1 \leq 0.4632,\\ 0.4519 \leq x_2 \leq 0.8519,\end{aligned} \tag{22}$$

the coolant flow input rate u is characterized as,

$$-0.7853 \leq u \leq 1.2147. \tag{23}$$

The system parameters considered are $\theta = 20$, $k = 300$, $M = 5$, $x_f = 0.3947$, $x_c = 0.3816$ 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 (6a). 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 $T_s = 0.5s$.

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 (6) was solved for all $x_i \in \mathcal{X}$ using CasADi v3.5.5 for Python [26].

Step II: Learning: For comparison, we train 3 different policies namely, (i) supervised learning approach $\pi_{approx}(x_i;\theta)$ in (8); (ii) log barrier penalty function for input and state constraints $\hat{\pi}_x(x_i;\theta)$ as in (17); (iii) the proposed method $\hat{\pi}_\lambda(x_i;\theta)$ in (19).

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. To speed up the training process, parallelism was implemented using the command `use_multiprocessing` available in Keras. The time required for training the policies is reported in Table. I, where it is observed that the proposed approach $\hat{\pi}_\lambda(x_i;\theta)$ is marginally increased.

A. Result Analysis

We use the standard NMPC results as primary reference and compare with various approaches described in Section III. Figs. 1–3 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. 1 show that, the most commonly used method of the control input generated by the policy $\pi_{approx}(x;\theta)$ results in violation of constraints imposed on the coolant flow rate input. Secondly, we observed that the

Manuscript 1135 submitted to 2021 60th IEEE Conference on Decision and Control (CDC). Received March 25, 2021.

TABLE I

COMPARISON OF TRAINING TIMES FOR VARIOUS POLICIES.

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

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 (17), are not sufficient for constraints satisfaction. Though as a convention, adding saturation to the control input for the policies $\pi_{approx}(x;\theta)$ and $\hat{\pi}_x(x;\theta)$ would satisfy the constraints on coolant flow rate input, the control performance gets degraded as demonstrated in Fig. 2. Finally, the results of $\hat{\pi}_\lambda(x_i;\theta)$ are same in Fig. 1 and 2 for illustration purposes, which denotes performance of the proposed method including the dual variables, characterized by (19). From Fig. 2, 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. 3, 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. 4.

Fig 4 shows evolution of states x_1 *vs* x_2 inside the feasible

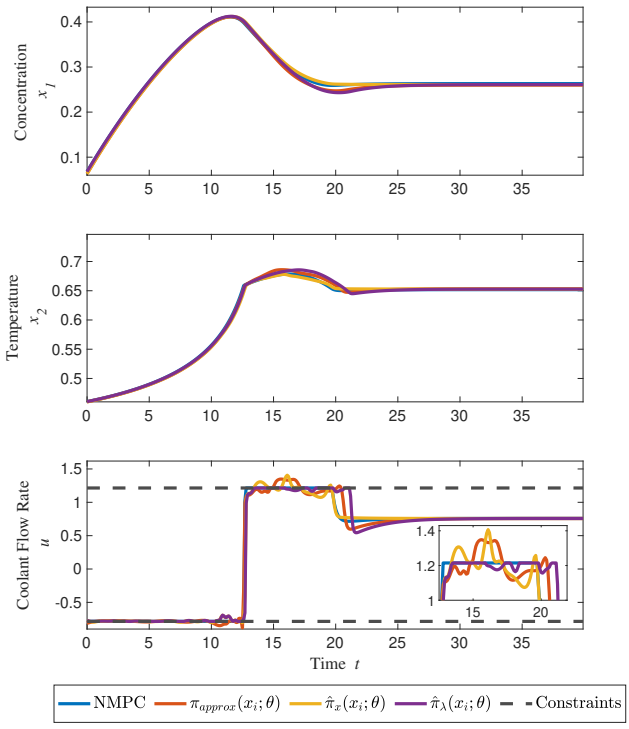

Fig. 1. CSTR benchmark: Performance comparison of state evaluation for x_1, x_2 and control input u with bounds using various approaches *v.i.z* traditional supervised learning approach $\pi_{approx}(x_i;\theta)$; log barrier penalty for input and state constraints $\hat{\pi}_x(x_i;\theta)$ and the proposed method $\hat{\pi}_\lambda(x_i;\theta)$.

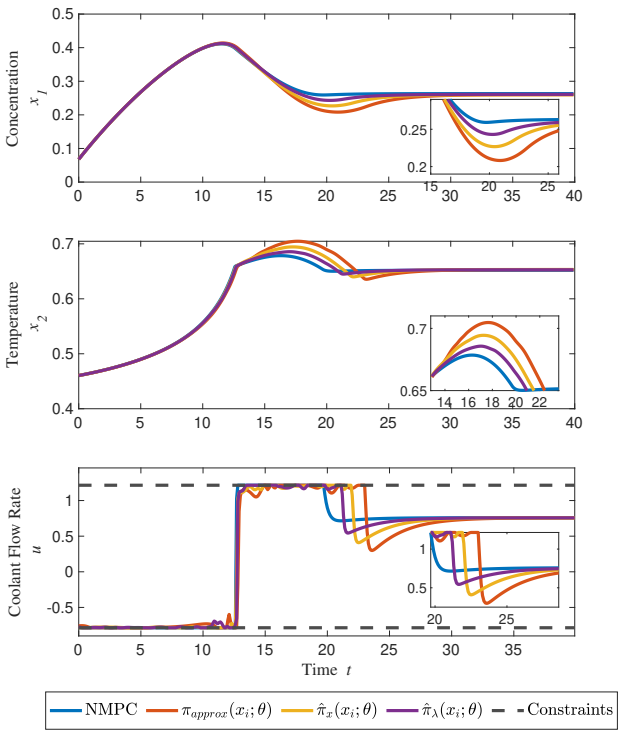

Fig. 2. CSTR benchmark: Performance comparison of control input u with bounds using various approaches *v.i.z* traditional supervised learning approach $\pi_{approx}(x_i;\theta)$ with input clipped; log barrier penalty for input and state constraints $\hat{\pi}_x(x_i;\theta)$ with input clipped and the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

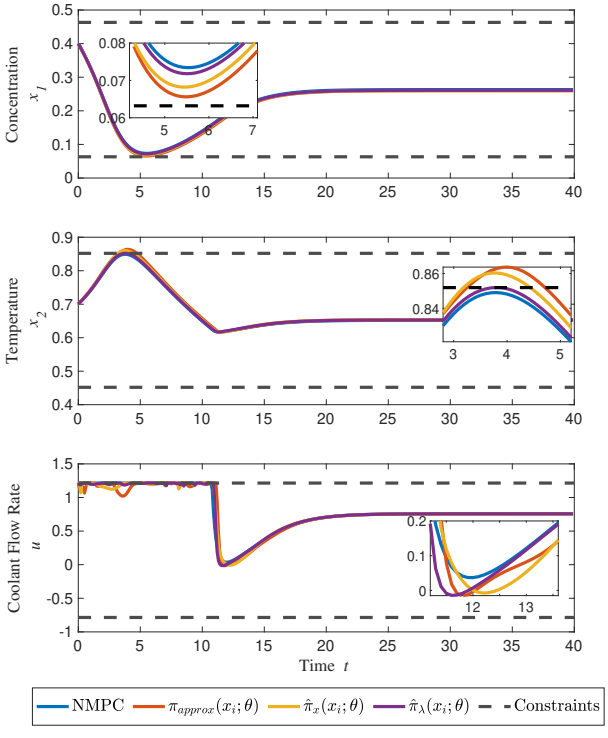

Fig. 3. Demonstration of state constraint satisfaction for CSTR benchmark: traditional supervised learning approach $\pi_{approx}(x_i;\theta)$ with clipping of control input; log barrier penalty for input and state constraints $\hat{\pi}_x(x_i;\theta)$ with clipping of control input and the proposed method $\hat{\pi}_\lambda(x_i;\theta)$ (without clipping of control input).

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.

We present detailed performance comparison in terms of key performance indices including Integral Square Error (ISE), Integral Absolute Error (IAE) and Integral Time Square Error (ITSE) for output states and Integral Squared Controller Efforts (ISCE) for the control input. The proposed method is evaluated for 5000 different trajectories and the resulting mean values of KPI's as well as 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 $\hat{\pi}_x(x_i;\theta)$.

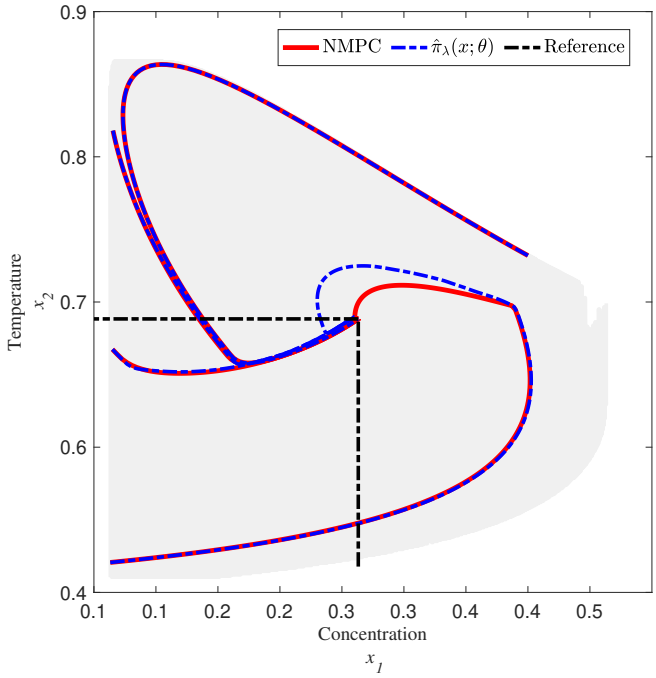

Fig. 4. CSTR benchmark: Concentration (x_1) *vs.* temperature (x_2) for randomly initiated evolution of states using proposed method against NMPC, black dashed line represents reference values.

V. CONCLUSION

In this paper, we propose 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 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 benchmark CSTR problem, 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.

TABLE II
COMPARISON OF KPI'S WITH STANDARD NMPC FOR OVER 5000 TRAJECTORIES

KPI	$\pi_{approx}(x;\theta)$	$\hat{\pi}_x(x_i;\theta)$	$\hat{\pi}_\lambda(x;\theta)$
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

REFERENCES

[1] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[2] S. Lucia and B. Karg, "A deep learning-based approach to robust nonlinear model predictive control," *IFAC-PapersOnLine*, pp. 511–516, 2018.

[3] S. Lucia, D. Navarro, B. Karg, H. Sarnago, and O. Lucia, "Deep learning-based model predictive control for resonant power converters," *IEEE Transactions on Industrial Informatics*, 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 *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 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)*. IEEE, 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*, 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*, 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*, 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, no. 7, pp. 441–452, 1992.

Manuscript 1135 submitted to 2021 60th IEEE Conference on Decision and Control (CDC). Received March 25, 2021.

[14] Y. Huang, "Lagrange-type neural networks for nonlinear programming problems with inequality constraints," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 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," 2019.
[19] S. Kakade and S. Shalev-Shwartz, "Mind the duality gap: Logarithmic regret algorithms for online optimization," *Advances in Neural Information Processing Systems*, vol. 22, 2008.
[20] Q. Nguyen and M. Hein, "The loss surface of deep and wide neural networks," in *International conference on machine learning*. PMLR, 2017, pp. 2603–2612.
[21] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010.
[22] 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*, 2013, pp. 5113–5118.
[23] A. Grancharova and T. A. Johansen, *Explicit nonlinear model predictive control: Theory and applications*, ser. Lecture Notes in Control and Information Sciences 429. Springer-Verlag Berlin Heidelberg, 2012.
[24] C. Zeng and H. Zhang, "A logarithmic barrier method for proximal policy optimization," *arXiv preprint arXiv:1812.06502*, 2018.
[25] 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*, pp. 1341–1353, 2011.
[26] 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.