ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/337531567

A Neural Network Architecture to Learn Explicit MPC Controllers from Data

Preprint · November 2019

CITATIONS
0

READS
24

4 authors, including:

Caio Guilherme da Silva Moraes
Federal University of Santa Catarina

9 PUBLICATIONS **8** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project

Sistema multiportas para o aproveitamento de energia proveniente da frenagem em locomotivas diesel-elétrica View project

All content following this page was uploaded by Caio Guilherme da Silva Moraes on 02 January 2020.

The user has requested enhancement of the downloaded file.

arXiv:1911.10789v1 [eess.SY] 25 Nov 2019

A Neural Network Architecture to Learn Explicit MPC Controllers from Data*

E. T. Maddalena, C. G. da S. Moraes, G. Waltrich and C. N. Jones

Abstract

We present a methodology to learn explicit Model Predictive Control (eMPC) laws from sample data points with tunable complexity. The learning process is cast in a special Neural Network setting where the coefficients of two linear layers and a parametric quadratic program (pQP) implicit layer are optimized to fit the training data. Thanks to this formulation, powerful tools from the machine learning community can be exploited to speed up the off-line computations through high parallelization. The final controller can be deployed via low-complexity eMPC and the resulting closed-loop system can be certified for stability using existing tools available in the literature. A numerical example on the voltage-current regulation of a multicell DC-DC converter is provided, where the storage and on-line computational demands of the initial controller are drastically reduced with negligible performance impact.

Keywords: Explicit model predictive control, machine learning, data-driven control, neural networks, power electronics.

1 Introduction

Model Predictive Control (MPC) is a valuable control methodology to be considered whenever performance needs to be maximized, possibly operating the system close to its physical constraints. For certain classes of problems – e.g. when the dynamics and constraints are affine and the objective is quadratic – a closed-form solution exists, mapping the current state to the control inputs through a continuous piecewise-affine (PWA) function. This approach is known as explicit Model Predictive Control (eMPC) and is particularly employed when the hardware where such a control law is to be implemented cannot solve the MPC mathematical program on-line (either for time limitations or even safety regulations regarding the final application). Unfortunately, eMPC controllers might also suffer from having high complexity due to the exponential growth of the number of regions in the worst case [1], thus leading to not only high memory requirements but also high computational times.

Scaling down the memory footprint and computational burden of eMPC is possible through a variety of techniques. Given a measurement of the current state, deciding in which region of the polyhedral partition the system is can be efficiently accomplished using search trees, hash tables, among other data structures [2, 3, 4]. If some information is known about the initial condition of the plant (even if only with respect to a subset of the states), reachability arguments can be used to drop polytopic cells that are guaranteed not to be visited during operation; this was recently studied in [5] and in [6]. Additional region elimination schemes were presented in [7] and [8], where an interpolation strategy and a backup control law were respectively defined to be used in case the system visited a state whose region had been eliminated. Finally, approximation methods for eMPC were investigated using polynomial functions [9], multi-scale basis [10], as well as piecewise-affine functions [11].

The use of Neural Networks (NN) to learn nonlinear MPC controllers was studied in [12], where one hidden layer and sigmoid activation functions were employed. The authors

*This work has received support from the Swiss National Science Foundation under the RISK project (Risk Aware Data-Driven Demand Response, grant number 200021 175627.

E. T. Maddalena and C. N. Jones are with École Polytechnique Fédérale de Lausanne (EPFL), Switzerland (e-mails: emilio.maddalena@epfl.ch, colin.jones@epfl.ch). C. G. da S. Moraes and G. Waltrich are with Universidade Federal de Santa Catarina (UFSC), Brazil (e-mails: caio.moraes@posgrad.ufsc.com, gierri.waltrich@ufsc.br).

assumed a specific degree of regularity of the obtained approximator in terms of a bound of an integral weighted by its Fourier decomposition coefficients. It was then shown that the final NN approximation error can be bounded. More recently, [13] proposed training a NN with rectified linear units by means of reinforcement learning instead of fully supervised learning. The context was that of quadratic MPC for linear systems. Their approach was shown to alleviate the training phase computations, especially since the system model can be exploited at this stage. Nevertheless, no guarantees of closed-loop stability are given or even tools to analyze it a posteriori – a rather common drawback of machine learning approximators.

Contribution: Herein we propose a method to fit datapoints sampled from an MPC controller – either in implicit or explicit form – with a particular type of Neural Netowrk. The architecture is defined by two linear layers and an implicit parametric quadratic program (pQP) layer, which together are able to capture any linear MPC problem with quadratic cost. In contrast with the majority of previous approaches such as [7, 8, 14], the final complexity is incremental and can be flexibly adjusted. The resulting approximator can be deployed essentially as a new simpler eMPC controller, and *closed-loop stability can be certified a posteriori* with tools already available in the literature. A case study in the power electronics domain, where several successful implementations of eMPC have been reported (see e.g. [15]), is presented to illustrate the potential of the proposed methodology. Although the proposed simplification technique gives up the original controller optimality, our numerical investigations show that significant complexity reduction is possible with little to negligible performance degradation[1].

Notation: $\mathbb{R}^n$ is the n-dimensional Euclidean space equipped with its usual norm. Given a matrix A, its transpose is denoted by A', and $A \succ (\succeq)\ 0$ means it is positive-definite (semidefinite). Furthermore, $||v||_A^2$ is defined as $||v'Av||^2$, I denotes an identity matrix of appropriate size, and $\mathbf{0}$ denotes a zero matrix of appropriate size. $\text{diag}(a_1, \ldots, a_n)$ represents a diagonal matrix with entries $a_1, \ldots, a_n$.

2 Learning Predictive Controllers

Consider the following standard MPC formulation for linear dynamical systems

$$\mathbb{P}1 : \min_{X,U} \quad \sum_{k=0}^{H-1} \left(x_k' Q x_k + u_k' R u_k\right) + x_H' P x_H \tag{1a}$$

$$\text{s.t.} \quad \forall k = 0, \ldots, H-1 \tag{1b}$$

$$x_{k+1} = A x_k + B u_k \tag{1c}$$

$$x_k \in \mathbb{X} \tag{1d}$$

$$u_k \in \mathbb{U} \tag{1e}$$

$$x_H \in \mathbb{X}_H \tag{1f}$$

$$x_0 = x(0) \tag{1g}$$

where $X := \{x_1, \ldots, x_H\}$, $U := \{u_0, \ldots, u_{H-1}\}$, $Q \succeq 0$, $P \succeq 0$, $R \succ 0$, and the constraints are all described by affine equalities and inequalities. Denote by $\boldsymbol{\pi} : \mathcal{X} \to \mathcal{U}$ the optimal solution of (1) in its parametric form with respect to the initial conditions $x(0)$, where $\mathcal{X} \subseteq \mathbb{R}^n$ is the feasible state space of $\mathbb{P}1$ and $\mathcal{U} \subseteq \mathbb{R}^m$ is the control space. We assume a set of N samples can be acquired from the original control law[2]

$$D = \{\mathrm{x}_i, \mathrm{u}_i\}_{i=1}^N \tag{2a}$$

$$\mathrm{u}_i = \boldsymbol{\pi}(\mathrm{x}_i),\ i = 1, \ldots, N \tag{2b}$$

The process of acquiring the training dataset does not have to follow a particular distribution, nor do the samples have to be independent.

[1] All MATLAB scripts and `python` code can be downloaded from `www.c4science.ch/source/QPFit`.

[2] The dataset can also be directly obtained from the implicit controller; depending on the size of the problem at hand, computing the explicit solution might be intractable.

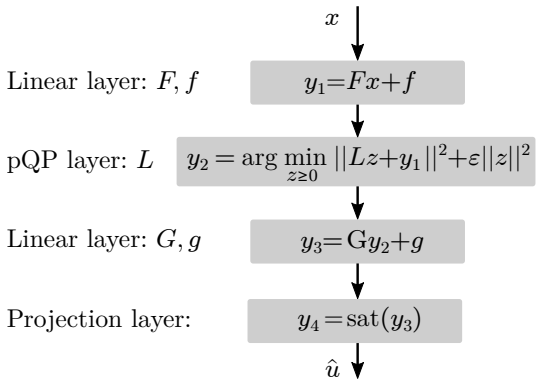

Figure 1: An illustration of the NN eMPC controller approximator. The parameters to be optimized are shown on the left-hand side.

2.1 The proposed architecture

The key idea behind the proposed approximator is the use of a parametric quadratic program layer as part of the Neural Network, and optimizing over its parameters in order to fit the available dataset. This layer is implicitly described by the quadratic program

$$z^\star = \arg \min_{z \geq 0} \; ||Lz + y_1(x)||^2 + \epsilon ||z||^2 \tag{3}$$

which is *always feasible* and *bounded from below*. The size of this mathematical program, i.e. the dimension of z, can be tuned to attain approximations with different complexity. Moreover, as notation suggests, the parameter y_1 depends on a previous affine layer that maps the system states into the z space, $y_1 := Fx + f$. Let $y_2 := z^\star$, then another affine layer maps the optimal solution to the input space $y_3 := Gy_2 + g$, and a projection onto the feasible input set produces the final control action $\hat{u} := \text{Proj}_{\mathbb{U}}(y_2)$. This last step is necessary to guarantee feasibility of the control moves (see e.g. [13]). An illustration of the proposed architecture is presented in Fig. 1, where the projection layer was particularized to a familiar element-wise saturation operation $\text{sat}(\cdot)$, valid for box input constraints.

Let the chosen number of decision variables in the pQP be $z \in \mathbb{R}^{n_z}$. We choose $L \in \mathbb{R}^{n_z \times n_z}$ to be square, and therefore $F \in \mathbb{R}^{n_z \times n}$, $f \in \mathbb{R}^{n_z}$. Moreover, $G \in \mathbb{R}^{m \times n_z}$ and $g \in \mathbb{R}^m$. If $n_z \geq n$, the first layer lifts the input data into a higher dimensional space before it is passed through the optimization layer. A last affine function then projects it onto the control space. These facts will be later employed to analyze the representative power of the network.

The parameters to be trained are therefore F, f, L, G and g. This process can be carried out via a stochastic gradient descent algorithm applied to an appropriate loss function. Differentiability of all layers is trivial with the exception of the pQP one [16, 17]. Regarding the later, note that the objective in (3) can be rewritten as

$$V(z) := z'(\epsilon I + L'L)z + (2L'y_1(x))'\, z + y_1(x)'y_1(x) \tag{4}$$

whose Lagrangian is simply

$$\mathcal{L}(z, \lambda) = V(z) - \lambda' z \tag{5}$$

The Karush-Kuhn-Tucker (KKT) conditions for primal and dual feasibility, complementary slackness, and stationarity then read

$$z^\star \geq 0 \tag{6a}$$
$$\lambda^\star \geq 0 \tag{6b}$$
$$\lambda_i^\star z_i^\star = 0, \; \forall i = 1, \ldots, n_z \tag{6c}$$
$$2(\epsilon I + L'L)z^\star + (2L'y_1(x)) - \lambda^\star = 0 \tag{6d}$$

where λ_i and z_i denote the components of the Lagrange multipliers and decision variables vectors. The following proposition presents the differentiability properties of the pQP layer, and holds since (3) is a particular instance of the `OptNet` layer [17] with strictly convex objective function.

Proposition 1: Let $\theta := (L, y_1)$. The parametric solution $z^\star(\theta)$ of (3) is subdifferentiable everywhere in its domain, i.e., $\partial z^\star(\theta) \neq \{\,\}$, and $\partial z^\star(\theta)$ has a unique element (the jacobian) everywhere but in a set of measure zero.

As shown in [17], the relevant gradients with respect to the parameters to be trained can be obtained from the KKT set of equations (6). Hence, backward passes are possible and backpropagation can be performed to optimize all of the NN parameters.

2.2 Properties of the approximator

The authors of [18] showed that any continuous PWA function can be obtained as the solution of a particular parametric linear program (pLP) transformed by a linear map. Even though this view could be adopted herein, we instead prove a different result that is enough in the context of linear eMPC.

Theorem 1: (The proposed NN architecture can learn any linear quadratic MPC controller) Let $\hat{\boldsymbol{\pi}} : \mathcal{X} \to \mathcal{U}$ be the map defined by the composition of all four layers, i.e., $\hat{\boldsymbol{\pi}}(x) := y_4 \circ y_3 \circ y_2 \circ y_1(x)$. Set $\epsilon = 0$, then $\exists F$, f, L, G and g with appropriate dimensions such that $\forall x \in \mathcal{X},\ \hat{\boldsymbol{\pi}}(x) = \boldsymbol{\pi}(x)$.

Proof: Define the following parametric problem

$$\mathbb{P}2 : \min_{U} \quad U'\,\Lambda\,U + x(0)'\,\Gamma\,U \tag{7a}$$

$$\text{s.t.} \quad \Phi\,U \leq \Omega\,x(0) + \omega \tag{7b}$$

obtained from $\mathbb{P}1$ by using the equality constraints to eliminate all state decision variables (see [19] for the construction of each matrix and vector). We also have that $\Lambda \succ 0$. Problems $\mathbb{P}2$ and $\mathbb{P}1$ are then equivalent in the sense that the solution $U^\star$ of $\mathbb{P}2$ and $\{X^\star, U^\star\}$ of $\mathbb{P}1$ have the same $U^\star$ component. Next, the dual problem of $\mathbb{P}2$ can be shown to be

$$\begin{aligned}\mathbb{D}2 : \min_{\lambda \geq 0} \ \frac{1}{4}\big[&\lambda'\Phi\,\Lambda^{-1}\Phi'\lambda + \ (4x(0)'\Omega' + 2x(0)'\Gamma\Lambda^{-1}\,\Phi' \\ &\cdots + 4\omega')\lambda + \ x(0)'\Gamma\Lambda^{-1}\Gamma' x(0)\big]\end{aligned} \tag{8}$$

From the stationarity optimality condition of $\mathbb{P}2$, it is possible to recover the primal optimal solution $U^\star$ from the dual optimal solution $\lambda^\star$

$$U^\star = -0.5\,\Lambda^{-1}\Phi'\lambda^\star - 0.5\,\Lambda^{-1}\Gamma' x(0) \tag{9}$$

It would be possible to enforce the first two layers to match the dual problem $\mathbb{D}2$, and the third to implement (9) and recover the primal solution. Nevertheless, this would require the third layer to have a so called skip connection directly from the NN input, i.e., access to $x(0)$. We instead define slightly different linear and pQP layers that not only learn $\mathbb{D}2$, but also let $x(0)$ pass through them and arrive at the second linear layer.

Let the auxiliary variable $\tilde{L}$ and function $\tilde{y}_1(x) := \tilde{F}x + \tilde{f}$ be the solution to (compare (4) and (8))

$$\tilde{L}'\tilde{L} + \epsilon I = 0.25\,\Phi\Lambda^{-1}\Phi' \tag{10a}$$

$$2\tilde{L}'\tilde{y}_1(x) = \Omega\,x(0) + 0.5\,\Phi\Lambda^{-1}x(0) + \omega \tag{10b}$$

which leads to $\epsilon = 0$, $\tilde{L} = 0.5\,(\Phi\tilde{\Lambda})'$, where $\tilde{\Lambda}$ is the unique square root of Λ^{-1}, guaranteed to exist as $\Lambda^{-1} \succ 0$. Then, $\tilde{y}_1(x) = (\Phi\tilde{\Lambda})^{-1}(\Omega\,x(0) + 0.5\,\Phi\Lambda^{-1}x(0) + \omega) \implies \tilde{F} = (\Phi\tilde{\Lambda})^{-1}(\Omega + 0.5\,\Phi\Lambda^{-1})$, $\tilde{f} = (\Phi\tilde{\Lambda})^{-1}\omega$.

Set the first layer weights to $F = [-I\ I\ \tilde{F}]'$ and $f = [\mathbf{0}\ \mathbf{0}\ \tilde{f}]'$ so that $y_1(x) = [-x;\ x;\ \tilde{F}x + \tilde{f}]$. Set the weights of the pQP layer (3) to $\epsilon = 0$ and $L = [I\ \mathbf{0}\ \mathbf{0};\ \mathbf{0}\ I\ \mathbf{0};\ \mathbf{0}\ \mathbf{0}\ \tilde{L}]$. If we partition the decision variable as $z = [\tilde{z}\ x^p\ x^n]'$, this results in

$$\min_{\tilde{z}, x^p, x^n \geq 0} \ ||x^p - x(0)||^2 + ||x^n + x(0)||^2 + ||\tilde{L}\tilde{z} + \tilde{y}_1(x)||^2 \tag{11}$$

which is a separable objective in $\tilde{z}$, $\tilde{x}^p$ and $\tilde{x}^n$. Due to the choice of $\tilde{L}$ and $\tilde{y}_1(x)$ in (10), the last term of the pQP matches the dual $\mathbb{D}2$ with the exception of its constant term – not

relevant for determining the optimal solution. Therefore, we have that $\tilde{z}^\star$ in (11) matches $\lambda^\star$ in $\mathbb{D}2$. Regarding $x^{p\star}$, the n optimizer components will satisfy $\forall i = 1, \ldots, n$, $x_i^{p\star} = x_i(0)$ if $x_i(0) \geq 0$, else $x_i^{p\star} = 0$. Similarly, $x_i^{n\star} = -x_i(0)$ if $x_i(0) \leq 0$, else $x_i^{n\star} = 0$. Therefore, $x^{p\star} - x^{n\star} = x(0)$.

Next set the weights of the second linear layer y_3 to $G = [-0.5\Lambda^{-1}\Phi' \; -0.5\Lambda^{-1}\Gamma' \; 0.5\Lambda^{-1}\Gamma']$ and $g = \mathbf{0}$. Therefore, $y_3 = G\,[\tilde{z}^\star \; x^{p\star} \; x^{n\star}]' = G\,[\lambda^\star \; x^{p\star} \; x^{n\star}]' = U^\star$, where equality (9) was used in the last step. Finally, note that the last layer $y_4 = \mathrm{Proj}_{\mathbb{U}}(y_3)$ will simply evaluate to y_3 since y_3 is the optimal primal solution $U^\star$, which satisfies the constraints (7b) and necessarily belongs to $\mathbb{U}$. The theorem then follows from the fact that $x(0)$ in the above calculations can be taken to be any point x in $\mathcal{X}$. $\square$

Exactly matching the original MPC controller would require L to have the same size of $\Phi\tilde{\Lambda}$ and $\epsilon = 0$ as shown. Nevertheless, we are interested precisely in reducing the complexity of the resulting controller through employing less parameters. In this process, choosing a regularizer $\epsilon > 0$ is beneficial since it ensures that the QP is bounded during the training phase for all possible parameters.

2.3 Stability certification

After the NN has been trained and its weights have been defined, closed-loop stability in the sense of Lyapunov can be verified through sum-of-squares (SoS) programming. First note that the NN is a composition of linear maps and optimization problems. The central idea is that the forward pass involves calculating the optimal solution to these mathematical programs, which necessarily satisfy their respective KKT conditions, e.g. (6). Furthermore, these conditions are sets of polynomial inequalities in the primal-dual lifted space; hence, the control moves are defined by polynomial and linear functions. As the system being controlled is linear, the certification technique presented in [20] can be readily applied.

2.4 Controller deployment

The final approximated control law $u = \hat{\boldsymbol{\pi}}(x)$ is evaluated by a forward pass in the NN. If the first and second linear layers are incorporated respectively into the pQP and projection ones, this can be done by solving the two (simpler) optimization problems on-line. Alternatively, two explicit PWA solutions can be calculated off-line with a bound on the number of regions dependent on the chosen size n_z. If the constraints $u \in \mathbb{U}$ are simply a box – as in many practical applications – an element-wise saturation is the closed-form solution to the projection layer $u = y_3 = \mathrm{Proj}_{\mathbb{U}}(y_2)$, further simplifying the final controller. We illustrate the use of the proposed technique in a power electronics context in the next section.

Any discrepancy between $\hat{\boldsymbol{\pi}}(x)$ and $\boldsymbol{\pi}(x)$ in the region that contains the origin will result in steady-state errors. Several techniques can be used to overcome this issue. As discussed in [12], the LQR solution associated with (1) can be stored and used whenever the system is inside the associated invariant set, where no constraints are active. Alternatively, an external disturbance estimator could be adopted to account for the optimal-approximate controller mismatch.

3 Control of a step-down converter

In order to enhance readability and stay consistent with the standard circuits convention, notation will be overloaded.

3.1 Analysis and controller design

Parallelism is a key concept to increase the efficiency and power levels of electronic converters. Still, this design choice has to be followed by proper current/voltage balancing techniques to ensure that no stage is subjected to a higher electrical stress when compared to the others.

A schematic representation of a multicell step-down converter is shown in Fig. 2, and its parameters can be found in Table 1. The topology features three arms that are connected to a coupled inductor, and an L-C output filter. All self inductances are assumed equal

Table 1: DC-DC converter parameters

V_{in}	L_s	L_m	R	L_f	C_o	R_o
350 V	4 mH	-2 mH	10 mΩ	270 μH	20 μF	6.25 Ω

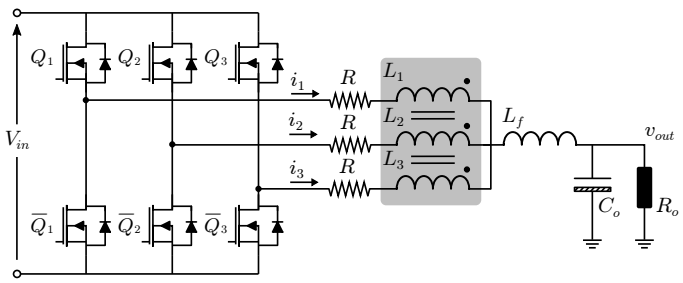

Figure 2: Circuit diagram of the multicell step-down DC-DC converter.

$L_1 = L_2 = L_3 = L_s$, and all mutual inductances have value L_m. The switches of each arm operate in a complementary fashion at a fixed frequency $f = 15$ kHz, and with variable but constrained duty cycle $0 \leq d_i \leq 0.9$, $i = 1, 2, 3$. Let the average voltage applied by the arms over one switching period be denoted by $v_i := d_i V_{in}, i = 1, 2, 3$. In order to ease the analysis, we apply the Lunze transform Ψ to all variables, decomposing the phase voltages and currents into differential and common mode components

$$\begin{bmatrix} i_{dm1} & i_{dm2} & i_{cm} \end{bmatrix}' := \Psi \begin{bmatrix} i_1 & i_2 & i_3 \end{bmatrix}' \tag{12}$$

$$\begin{bmatrix} v_{dm1} & v_{dm2} & v_{cm} \end{bmatrix}' := \Psi \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}' \tag{13}$$

where $\Psi = (1/3)\,[2\ -1\ -1;\ -1\ \ 2\ -1;\ 1\ \ 1\ \ 1]$.

The control input is defined as $u := [v_{dm1}\ v_{dm2}\ v_{cm}]'$ and the continuous-time state vector, by appending the output voltage to the transformed currents $x := [i_{dm1}\ i_{dm2}\ i_{cm}\ v_{out}]'$. By using Kircchoff's circuit laws, a linear model of the form $\dot{x} = A_{ct}x + B_{ct}u$ can be derived with

$$A_{ct} = \begin{bmatrix} \frac{-R}{L_s+L_m} & 0 & 0 & 0 \\ 0 & \frac{-R}{L_s+L_m} & 0 & 0 \\ 0 & 0 & \frac{-R}{L_s+2L_m+3L_f} & \frac{-1}{L_s+2L_m+3L_f} \\ 0 & 0 & \frac{3}{C_o} & \frac{-1}{R_oC_o} \end{bmatrix} \tag{14}$$

$$B_{ct} = \begin{bmatrix} \frac{1}{L_s} & 0 & 0 \\ 0 & \frac{1}{L_s} & 0 \\ 0 & 0 & \frac{1}{L_s+2L_m+3L_f} \\ 0 & 0 & 0 \end{bmatrix} \tag{15}$$

Finally, discretization at frequency f is carried out using the zero-order hold method, yielding $x_{k+1} = Ax_k + Bu_k$.

The control goal is to regulate the output voltage v_{out} to 300 V while maintaining the phase currents balanced at all times, which translates to driving the differential currents to zero. More specifically we have the following fixed reference $x_{eq} = [0\ 0\ 16\ 300]'$ with $u_{eq} = B^{\dagger}(I - A)x_{eq}$, where $B^{\dagger}$ is the pseudo-inverse of B. Moreover, the controller approximation procedure must not incur a steady-state error larger than 200 mA for i_{dm1} and i_{dm2}, and 5% for the common mode component i_{cm} and output voltage v_{out}. The chosen MPC cost function was

$$J = \sum_{k=0}^{H-1} (||x_k - x_{eq}||_Q^2 + ||u_k - u_{eq}||_R^2) \ + \ ||x_N - x_{eq}||_P^2 \tag{16}$$

where $Q = \text{diag}(10,\ 10,\ 0.1,\ 0.1)$, $R = 0.1\,I$, P is the solution to the associated the discrete-time algebraic Riccati equation, and $H = 10$. For all time instants, box state constraints

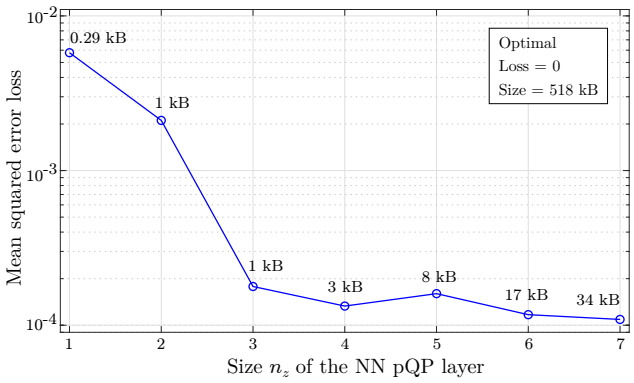

Figure 3: Neural network training loss as a function of the pQP layer size, and storage requirements associated with their PWA representations.

were imposed $\begin{bmatrix}-5 & -5 & -10 & -20\end{bmatrix}' \leq x_k \leq \begin{bmatrix}5 & 5 & 30 & 400\end{bmatrix}'$ and polyhedron constraints on the controls $H_u\, u_k \leq h_u$ that simply mapped the duty cycle saturation to the Lunze domain. Due to the polytopic input constraints, the system cannot be decomposed into three decoupled parts as the structure of matrices A_{ct} and B_{ct} suggest. Furthermore, the standard terminal set constraint was imposed on x_H, defined as the invariant set associated to the unconstrained infinite-time problem formulation.

3.2 Learning the optimal controller

With the aid of the Multi-Parametric Toolbox (MPT) [21], the optimal eMPC solution $\boldsymbol{\pi}(x)$ was calculated and consisted of 2'337 critical regions. By counting the number of parameters necessary to describe each halfspace and control gain, the memory requirement of this PWA function was found to be 518 kB. In the previous calculation, a 4 byte representation was considered for both integers and floating point numbers.

Next, 5'000 samples were randomly acquired from the eMPC controller using a uniform distribution. The first and second components of the sampled control moves had considerably smaller amplitudes compared to the third due to the structure of the Lunze transform Ψ. The dataset labels $\{\mathrm{u}_i\}_{i=1}^{5000}$ had therefore to be scaled to ensure a similar learning of all control components. Moreover, instead of $\mathrm{Proj}_{\mathbb{U}}(y_3)$, the last NN layer was simplified to $y_4 = \Psi\,\mathrm{sat}(y_3)$ with saturation limits 0 and $0.9\,V_{in}$. This clearly guarantees control feasibility without the need of a second quadratic program. NN approximators were trained using `PyTorch` and the `OptNet` framework [17]. A mean squared error loss function was minimized by employing the Adam algorithm, mini-batch stochastic gradient descent with batch size 50, and 150 epochs. The size n_z of the pQP layer was varied from 1 to 7 and a total of 10 models were trained for each size; the lowest obtained losses are shown in Figure 3. On average, training a model required 42 minutes on a 3.1 GHz Intel Core i7 machine without GPU acceleration, and 23 minutes with a single NVIDIA Tesla T4 graphics card. The learned parameters were then exported to MATLAB in order to calculate the PWA solution of the pQP layer.

An increase in the n_z size clearly expands the representation capabilities of the neural network. This however does not always translate to a decrease in the final loss since the training process is affected by the weights initialization among other factors. Although not monotonic, we see a decrease of the overall training loss in Figure 3 as n_z grows. In order to validate the models, the start-up response of the converter was analyzed under all 7 different approximate controllers, and only the two largest ones ($n_z = 6$ and $n_z = 7$) met the target specifications given in Section 3.1. We refer to these two solutions as the *viable learned controllers*. Slices of their control surfaces are shown in Figure 4[3], and a phase portrait of the closed-loop system evolution over 50 steps starting from four initial conditions is depicted in Figure 5. A summary of the two viable learned controller features is presented in Table 2, including the number of polytopic regions, the storage requirements, the worst-

[3]The v_{out} axis was extended to -100 V for visualization purposes.

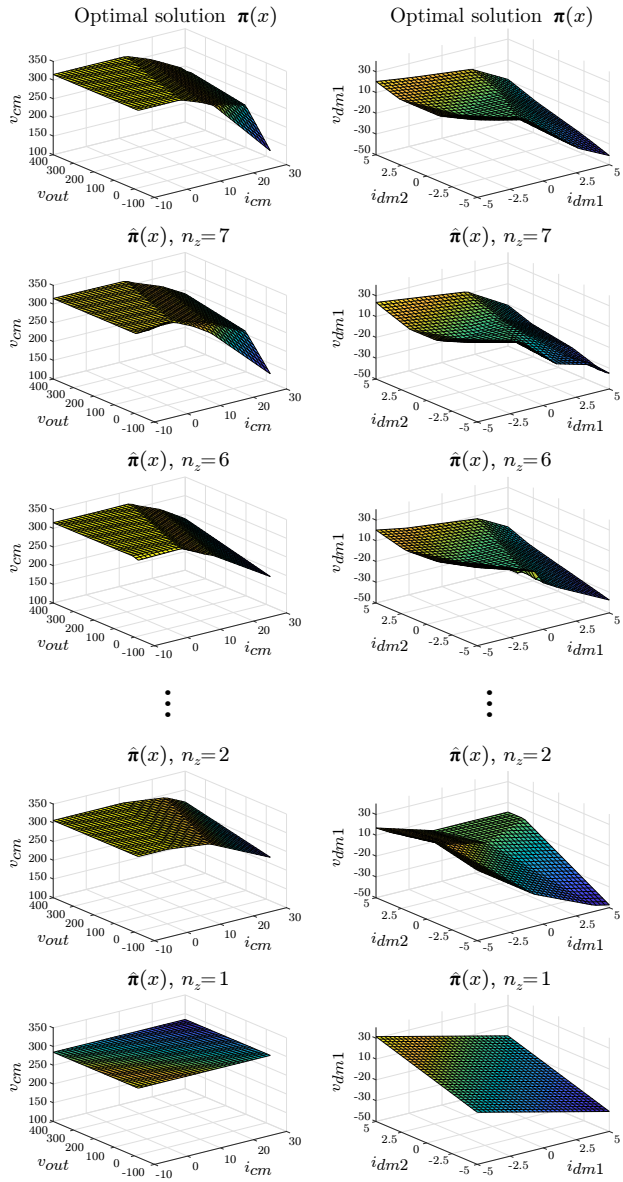

Figure 4: Slices of the optimal eMPC controller $\boldsymbol{\pi}(x)$ and several PWA NN approximations $\hat{\boldsymbol{\pi}}(x)$. The left plots are associated to v_{cm} and the right plots, to v_{dm1}.

case computation time[4] and the steady state (SS) error for i_{cm} and v_{out} – both clearly always equal. Even though four initial states were given, the systems always converged to the same points and, hence, only one SS error is reported. Plus, the storage numbers also take into account all the remaining layers parameters. Analyzing the obtained results we see that the approximations drastically reduced the storage requirements by 93.4% and 96.7% and sped up the worst-case evaluation time by 83.7% and 88.4%, respectively for the $n_z = 7$ and $n_z = 6$ cases. The closed-loop trajectories with the proposed $\hat{\boldsymbol{\pi}}(x)$ remained reasonably close to the scenario with the optimal $\boldsymbol{\pi}(x)$, converging to nearby equilibrium points. In practice, steady-state errors could be removed by using the tools mentioned in Section 2.4.

[4]Before proceeding to implementation, a further speed up is possible through the methods listed in the Introduction.

Table 2: Optimal and viable learned controllers information

	Regions	Storage	Comp. time	SS error
$\boldsymbol{\pi}(x)$	2'337	518 kB	12.9 ms	0%
$\hat{\boldsymbol{\pi}}(x)$, $n_z = 7$	107	34 kB	2.1 ms	0.59%
$\hat{\boldsymbol{\pi}}(x)$, $n_z = 6$	56	17 kB	1.5 ms	1.25%

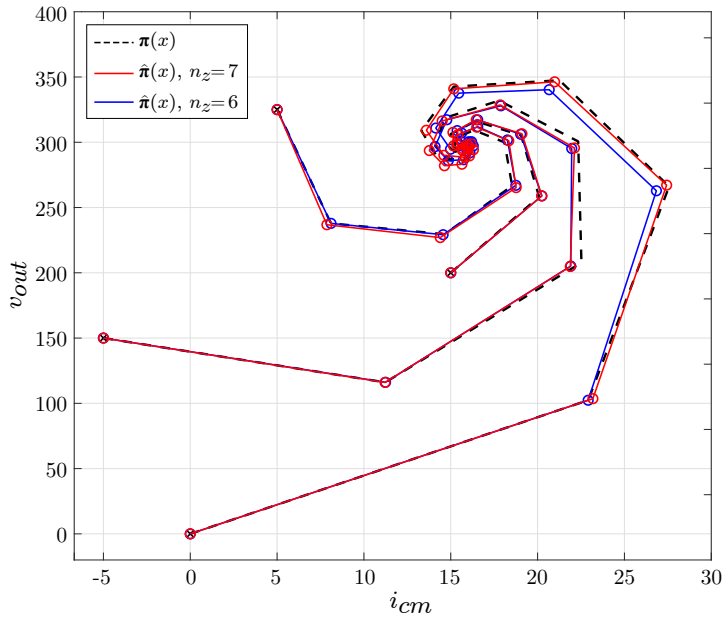

Figure 5: Output voltage and common mode current phase portraits when employing the optimal controller and the two viable approximations.

4 Conclusion

We proposed a novel method to approximate explicit MPC controllers from samples of the optimal control law. The essence of the approach is a Neural Network architecture featuring a pQP layer incorporated to learn the MPC dual problem. It was shown that, with an appropriate pQP size, it is possible to learn any linear quadratic MPC exactly. After its training, an equivalent representation of the NN can be found off-line as a new and simpler controller. Steady-state errors due to approximation inaccuracies can be mitigated by employing standard tools. A numerical example was provided where the storage requirements and computational burden of the approximate PWA controllers were significantly lower than their optimal counterpart with minor performance degradation.

References

[1] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear model predictive control*. Springer, 2009, pp. 345–369.

[2] C. N. Jones, P. Grieder, and S. V. Raković, "A logarithmic-time solution to the point location problem for parametric linear programming," *Automatica*, vol. 42, no. 12, pp. 2215–2218, 2006.

[3] F. Bayat, T. A. Johansen, and A. A. Jalali, "Using hash tables to manage the time-storage complexity in a point location problem: Application to explicit model predictive control," *Automatica*, vol. 47, no. 3, pp. 571–577, 2011.

[4] M. Mönnigmann and M. Kastsian, "Fast explicit MPC with multiway trees," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1356–1361, 2011.

[5] E. T. Maddalena, R. K. H. Galvão, and R. J. M. Afonso, "Robust region elimination for piecewise affine control laws," *Automatica*, vol. 99, pp. 333–337, 2019.

[6] M. Kvasnica, P. Bakaráč, and M. Klaučo, "Complexity reduction in explicit MPC: A reachability approach," *Systems & Control Letters*, vol. 124, pp. 19–26, 2019.

[7] J. A. Rossiter and P. Grieder, "Using interpolation to improve efficiency of multiparametric predictive control," *Automatica*, vol. 41, no. 4, pp. 637–643, 2005.

[8] F. J. Christophersen, M. N. Zeilinger, C. N. Jones, and M. Morari, "Controller complexity reduction for piecewise affine systems through safe region elimination," in *2007 46th IEEE Conference on Decision and Control*, 2007, pp. 4773–4778.

[9] M. Kvasnica, J. Löfberg, and M. Fikar, "Stabilizing polynomial approximation of explicit MPC," *Automatica*, vol. 47, no. 10, pp. 2292–2297, 2011.

[10] S. Summers, C. N. Jones, J. Lygeros, and M. Morari, "A multiresolution approximation method for fast explicit model predictive control," *IEEE Transactions on Automatic Control*, vol. 56, no. 11, pp. 2530–2541, 2011.

[11] C. N. Jones and M. Morari, "Polytopic approximation of explicit model predictive controllers," *IEEE Transactions on Automatic Control*, vol. 55, no. 11, pp. 2542–2553, 2010.

[12] T. Parisini and R. Zoppoli, "A receding-horizon regulator for nonlinear systems and a neural approximation," *Automatica*, vol. 31, no. 10, pp. 1443–1451, 1995.

[13] 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.

[14] M. Kvasnica and M. Fikar, "Clipping-based complexity reduction in explicit MPC," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1878–1883, 2011.

[15] S. Mariéthoz and M. Morari, "Explicit model-predictive control of a PWM inverter with an LCL filter," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 2, pp. 389–399, 2008.

[16] S. Gould, B. Fernando, A. Cherian, P. Anderson, R. S. Cruz, and E. Guo, "On differentiating parameterized argmin and argmax problems with application to bi-level optimization," *arXiv preprint arXiv:1607.05447*, 2016.

[17] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 136–145.

[18] A. B. Hempel, P. J. Goulart, and J. Lygeros, "Every continuous piecewise affine function can be obtained by solving a parametric linear program," in *2013 European Control Conference (ECC)*, 2013, pp. 2657–2662.

[19] S. J. Wright, "Efficient convex optimization for linear MPC," in *Handbook of Model Predictive Control*. Springer, 2019, pp. 287–303.

[20] M. Korda and C. N. Jones, "Stability and performance verification of optimization-based controllers," *Automatica*, vol. 78, pp. 34–45, 2017.

[21] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *2013 European control conference (ECC)*, 2013, pp. 502–510.

View publication stats