IEEE Open Journal of the
Industrial Electronics Society

Received 15 December 2020; revised 25 January 2021; accepted 6 February 2021. Date of publication 10 February 2021; date of current version 8 March 2021. The review of this article was arranged by Associate Editor Z. Zhang.

Digital Object Identifier 10.1109/OJIES.2021.3058411

Embedded PWM Predictive Control of DC-DC Power Converters Via Piecewise-Affine Neural Networks

EMILIO T. MADDALENA [1], MARTIN W. F. SPECQ [1], VIVIANE L. WISNIEWSKI [2], AND COLIN N. JONES [1] (Member, IEEE)

[1]Automatic Control Laboratory, École Polytechnique Fédérale de Lausanne, Route Cantonale, 1015 Lausanne, Switzerland
[2]Power Electronics Laboratory, Berner Fachhochschule, Quellgasse 21, 2501 Biel, Switzerland

CORRESPONDING AUTHOR: EMILIO T. MADDALENA (e-mail: emilio.maddalena@epfl.ch).

This work was supported by the Swiss National Science Foundation under the RISK project (Risk Aware Data-Driven Demand Response), under Grant 200021 17627. *(Emilio T. Maddalena and Martin W. F. Specq contributed equally to this work.)*

ABSTRACT Predictive control is a flexible control methodology that can optimize performance while satisfying current and voltage constraints. Its application in the power electronics domain is however hampered by the high computational demands associated with it. In this paper, piecewise-affine neural networks are explored to greatly simplify these controllers and allow for an inexpensive implementation in commercial hardware. More specifically, we tackle the problem of enhancing the start-up transient response of a step-down dc-dc converter while also satisfying inductor current constraints. We analyze the neural network architecture, and detail its training and validation procedures. The learned controller is then embedded on an inexpensive 80-MHz microcontroller, and experimental results are provided showing that the whole control algorithm can be executed in under 30 microseconds.

INDEX TERMS Model predictive control, embedded deployment, neural networks, dc-dc converters.

I. INTRODUCTION

Well designed compensators are at the heart of high-performance power electronic devices. In this context, linear control techniques are undoubtedly the most popular methodologies used by practitioners [1]–[3], usually exploiting problem-specific architectures and relying on proportional-integral-derivative (PID) blocks [4]–[6]. These are generally reliable and robust solutions capable of accomplishing voltage/current reference tracking with, most of the time, zero steady-state error. Another rather positive aspect of such techniques that cannot be overlooked is their simplicity in terms of implementation, leading to simple firmware that can be easily debugged and tuned by users.

In contrast to linear methodologies, model predictive control (MPC) is a model-based approach that can *systematically handle* state and input constraints. As a result, performance can be pushed to its limits while still respecting physical limitations of the electronic components [7]. Arguably one of the main issues that prevents the wide applicability of MPC is its computational complexity: at each sampling period an optimization problem has to be solved on-line, requiring the execution of intricate numerical routines [8], [9]. In a recent study [10], researchers have shown that at times clever variable transformations can unveil the hidden convexity of certain difficult (non-linear and non-convex) problems, thus alleviating the on-line computational burden. Nevertheless, even embedding convex solvers can be infeasible when high control frequencies are needed or the computing platform has low processing power [11]. Another popular branch of predictive control in power electronics is the so called 'finite control-set' approach [12], where the switching elements are directly considered as controllable without the need of PWM modulating blocks. Although more flexibility is gained, this comes at the price of having to tackle a mixed-integer optimization problem, which often forces users to select extremely short prediction horizons—oftentimes equal to one. See [13] for a detailed study on the matter. From a stability point of view, such limitation is severe as short-sighted

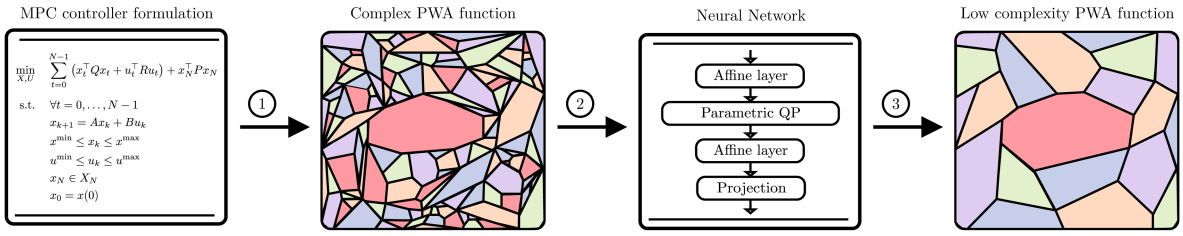

FIGURE 1. **Pre-processing of the control law: The MPC controller is designed, and the problem is then solved off-line (①), yielding a (potentially complex) PWA function. Samples are collected from this function (②), creating a dataset composed of state-control pairs $\{x_n, u_n\}_{n=1}^{N}$. The Neural Network is trained on the examples given in the dataset and the NN parameters are learned. Finally, the pQP layer is converted into a PWA function (③) whose complexity can be adjusted by choosing the size of the NN.**

controllers are known to lead to unstable closed-loop dynamics among other problems [14].

In this context, explicit MPC (EMPC) comes as an alternative implementation scheme for predictive control whereby the whole optimization problem is solved off-line, producing a look-up table of gains to be used during operation [15]. EMPC has been applied in the context of power electronics to systems such as inverters with LCL filters [16], dc-dc boost converters [17] and permanent-magnet synchronous motors [18]. Unfortunately, when the number of states and inputs is large or the prediction horizon is long—which is usually desirable since it promotes stability—EMPC becomes computationally very demanding. In more specific terms, the number of regions over which the piecewise control law is defined can grow exponentially with the number of constraints [15]. This in turn leads to two problems: firstly, the memory footprint required to store the entire control law greatly increases; secondly, the on-line task of identifying the current region becomes highly time-consuming. The latter is known as the point-location problem.

During the past years, researchers in the control field have produced a large body of work on scaling down the complexity of EMPC with many ingenious ideas. These include exploiting information regarding the initial condition of the system to 'delete' certain regions [19], approximating the optimal solution with specific sets of basis functions [20], finding efficient data structures to store and evaluate the piecewise control law [21], etc. For a reference in the power electronics and power systems field, the reader is referred to the thesis [22]. More recently, there has been an increasing interest in importing tools from the domain of machine learning to controls. The hope is that of exploiting data to, for example, refine performance iteratively [23] and build surrogate dynamical models with non-parametric techniques [24], [25]. Neural networks (NN) have also been used to scale down the computational demands of MPC by learning simplified representations of the original controller [26]–[28]. See [29], [30] for recommendations on how to gather the necessary data-points and properly train such models.

In this paper, we build on our previous work [31] where a novel piecewise-affine neural network (PWA-NN) architecture was presented to simplify EMPC controllers. Herein a practical investigation of such technique is carried out in the context of fast dc-dc converters, where we cover controller design, the steps required to train the network off-line, and the ones needed to accomplish embedded implementation on an inexpensive device. As opposed to the NN-based control techniques studied in [26]–[28], [32], [33], the PWA-NN *preserves the MPC structure* after training, yielding a simplified PWA map. A diagram of the proposed approach is shown in Figure 1. Moreover, the final complexity can be adjusted by the user through selecting the network size. In contrast with finite control-set MPC, in this work we assume the use of PWM modulation and the switching signal duty cycle is controlled. The following points summarize this paper:

- The dynamic model of a buck Dc-Dc converter is derived in details considering various parasitic elements;
- An MPC controller is designed to attain a fast and smooth transient response while respecting constraints, and the controller is converted into its explicit form;
- A PWA neural network architecture is employed to learn the predictive controller, resulting in a precise description of it with only a small fraction of the original processor and memory demands;
- The resulting controller is then deployed on an STM32L476 80 MHz microcontroller—a considerably cheaper and simpler target device when compared to the previous works that carried out embedded implementations e.g. [17], [18], [28]. Finally, we provide experimental results in closed-loop and show that the computations are executed in under 30 μs.[1]

II. MATHEMATICAL MODEL

A schematic representation of the buck converter considered in this work is shown in Figure 2 and its parameters are found in Table 1. V_{IN}, V_D, L, and C refer respectively to the input voltage, the diode forward drop, the inductance and the capacitance; whereas R_{ON}, R_L, R_C and R_O refer to the switch on-resistance, the inductor parasitic resistance, the capacitor parasitic resistance, and the output load.

[1] All Python and MATLAB scripts, the embedded firmware and the PCB files are available at github.com/emilioMaddalena/MPCfit

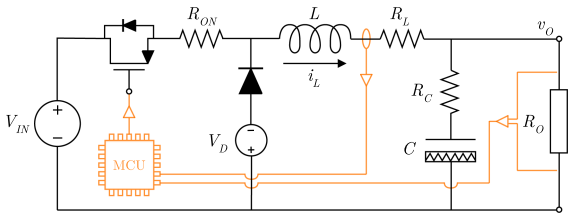

FIGURE 2. A circuit diagram of the buck converter including its parasitic resistances and the diode forward voltage drop. The feedback loop is closed by the MCU, which implements our proposed PWA-NN controller.

TABLE 1. Parameters of the DC-DC Converter

V_{IN}	V_{OUT}	V_D	L	C
15 V	5 V	0.1 V	10 mH	56 μF

R_{ON}	R_L	R_C	R_O	f_{sw}
5 mΩ	2 Ω	330 mΩ	100 Ω	20 kHz

We choose as state variables the inductor current and the output voltage

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} i_L \\ v_O \end{bmatrix} \tag{1}$$

The power switch is operated at a constant frequency f_{sw} and variable duty cycle, which is taken to be the control variable $u = \delta$. Following the classical time-averaging technique, Kircchoff's circuit laws are used to derive differential equations for both when the switch is open, and when it is closed. The expressions can be found in Appendix A. Averaging these equations with δ as a weight yields

$$\dot{x}_1 = -\frac{R_L}{L}x_1 - \frac{1}{L}x_2 + \frac{V_{IN}+V_D}{L}u - \frac{R_{ON}}{L}x_1 u - \frac{V_D}{L} \tag{2a}$$

$$\begin{aligned}\dot{x}_2 = &-\frac{R_C R_O R_L C + R_O L}{(R_C+R_O)LC}x_1 - \frac{R_C R_O C + L}{(R_C+R_O)LC}x_2 \\ &+ \frac{R_C R_O (V_{IN}+V_D)}{(R_C+R_O)L}u - \frac{R_C R_O R_{ON}}{(R_C+R_O)L}x_1 u \\ &- \frac{R_C R_O V_D}{(R_C+R_O)L}\end{aligned} \tag{2b}$$

The expressions above are not linear since the inductor current and the duty cycle multiply each other. As the goal is to design a linear MPC controller, linearization is needed. We first fix the output voltage to the desired value x_{2eq} and solve for the current and duty cycle steady-state values

$$x_{1eq} = \frac{x_{2eq}}{R_O} \tag{3}$$

$$u_{eq} = \frac{R_O V_D + (R_L + R_O)x_{2eq}}{R_O(V_{IN}+V_D) - R_{ON}x_{2eq}} \tag{4}$$

Finally, (2a) and (2b) are expanded around (x_{1eq}, u_{eq}) and the linear terms are kept, leading to the familiar state-space equations $\dot{x} = A_{ct}x + B_{ct}u$ where

$$A_{ct} = \begin{bmatrix} -\frac{R_L + R_{ON}u_{eq}}{L} & -\frac{1}{L} \\ -\frac{R_C R_O (R_L C - R_{ON} C u_{eq}) + R_O L}{(R_C+R_O)LC} & -\frac{R_C R_O C + L}{(R_C+R_O)LC} \end{bmatrix} \tag{5}$$

$$B_{ct} = \begin{bmatrix} \frac{V_{IN}+V_D-R_{ON}x_{1eq}}{L} \\ \frac{R_C R_O (V_{IN}+V_D-R_{ON}x_{1eq})}{(R_C+R_O)L} \end{bmatrix} \tag{6}$$

As a last step, a discrete-time model $x_{t+1} = Ax_t + Bu_t$ is obtained by integrating the continuous-time dynamics using the standard zero-order hold method. The chosen discretization frequency was $f_{samp} = 10$ kHz, which is also the predictive controller frequency.

III. CONTROL GOALS AND CONTROLLER DESIGN

The goal is to attain a fast start-up response with as little overshoot as possible and regulate the output voltage v_O to $v_{eq} = 5$ V. Furthermore, an inductor current constraint of 200 mA and voltage constraint of 7 V must be respected at all times. The prediction horizon has to be long enough to yield a large feasible set [14] and we chose $N = 10$ steps. A standard quadratic objective was employed,[2] penalizing the deviation of the states and control variable from the reference values $x_{eq} = [0.05 \;\; 5]^\top$, $u_{eq} = 0.3379$. The final optimal-control formulation was

$$\min_{X,U} \sum_{t=0}^{N-1} \left(\|x_t - x_{eq}\|_Q^2 + \|u_t - u_{eq}\|_R^2 \right) + \|x_N - x_{eq}\|_P^2 \tag{7a}$$

$$\text{s.t.} \quad \forall t = 0, \ldots, N-1$$

$$x_{t+1} = Ax_t + Bu_t \tag{7b}$$

$$\begin{bmatrix} i_L^{\min} \\ v_O^{\min} \end{bmatrix} \le x_t \le \begin{bmatrix} i_L^{\max} \\ v_O^{\max} \end{bmatrix} \tag{7c}$$

$$u^{\min} \le u_t \le u^{\max} \tag{7d}$$

$$x_N \in \mathcal{X}_N \tag{7e}$$

$$x_0 = x(0) \tag{7f}$$

with state and control constraints

$$x^{\min} = \begin{bmatrix} i_L^{\min} \\ v_O^{\min} \end{bmatrix} = \begin{bmatrix} 0\,\text{mA} \\ 0\,\text{V} \end{bmatrix} \tag{8}$$

$$x^{\max} = \begin{bmatrix} i_L^{\max} \\ v_O^{\max} \end{bmatrix} = \begin{bmatrix} 200\,\text{mA} \\ 7\,\text{V} \end{bmatrix} \tag{9}$$

$$u^{\min} = 0, \quad u^{\max} = 1 \tag{10}$$

The matrix weights were $Q = \text{diag}(90, 1)$, $R = 1$, and P was the solution of the associated discrete-time algebraic Ricatti equation. $\mathcal{X}_N$ was chosen to be the the system's maximal

[2] In the MPC objective function, the squared weighted norms read as in $\|x_t - x_{ref}\|_Q^2 = (x_t - x_{ref})^\top Q\,(x_t - x_{ref})$.

invariant set under the corresponding LQR policy. Both the terminal ingredients (P and $\mathcal{X}_N$) can be easily calculated with the aid of the Multi-Parametric Toolbox (MPT) [34] for MATLAB, and are employed to ensure recursive feasibility and closed-loop stability [14].

As a final step, the MPC controller (7) was solved off-line using MPT, which yielded a piecewise-affine (PWA) function $\pi(x)$ that maps states directly to optimal control inputs. As well known in the area of explicit model predictive control [15], this function partitions the space of feasible states $\mathcal{X}$ into regions described by sets of linear inequalities. Then, applying the predictive controller on-line boils down to implementing the look-up table of feedback gains

$$u = \pi(x) = \begin{cases} F_1 x + g_1, & \text{if } x \in \text{region } 1 \\ \dots & \dots \\ F_M x + g_M, & \text{if } x \in \text{region } M \end{cases} \tag{11}$$

As provided by MPT, the computed control policy $\pi(x)$ had $M = 70$ regions, a number too large to be embedded into the target MCU due to the large storage and computational demands (more details are given in Section IV). These implementation issues motivate the use of our PWA-NN complexity reduction scheme.

IV. LEARNING A FAITHFUL STILL SIMPLER REPRESENTATION OF THE CONTROLLER

Explicit MPC controllers are the exact parametric solution of their optimization counterparts. The geometric landscape depicted by the PWA function $\pi(x)$ is composed of numerous linear pieces patched together. At times, neighboring regions share the same control law and, depending on their arrangement, they could be merged into an equivalent single one. Moreover, the overall surface usually presents two scales of complexity: a general shape and, inspecting it more closely, intricate small details. Based on these observations, it is reasonable to try to reproduce the rough shape of $\pi(x)$ without necessarily replicating its small wiggles.

A. THE GENERAL ARCHITECTURE

The architecture of the piecewise-affine neural network used to learn $\pi(x)$ is shown in Figure 3. It has two affine layers (L1 and L3), an optimization problem as the activation layer [35] (L2) and one projection layer (L4) that in this specific case is simply a saturation function. The latter is needed to ensure that the final control values produced by the NN are within the control bounds $0 \leq u \leq 1$. As discussed in [31], the motivation behind the structure is that of learning the dual MPC problem: L1 maps the state x to the dual space, where L2 represents the dual optimization problem that is solved, L3 then maps the solution back to the primal space, and finally L4 guarantees it respect the control constraints.

As opposed to other approaches to learning MPC controllers with NN [27], [28], the one explored here can be translated to a *closed-form* piecewise-affine function. More

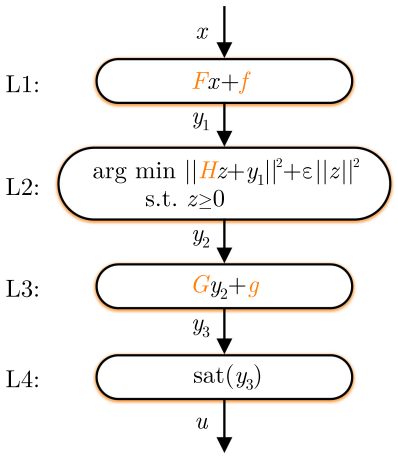

FIGURE 3. The PWA-NN architecture. A dataset composed of states x and control actions u is collected from the EMPC controller off-line, and used to train the PWA-NN parameters (highlighted in orange). After the training and validation phases, the L2 layer is then converted into its piecewise-affine form to ease implementation.

specifically, the parametric quadratic program in layer L2 can be solved off-line after training (e.g. by using MPT), yielding a PWA map of the same form as (11). The complexity of such function in terms of the number of regions can be adjusted by choosing the size of matrix $H \in \mathbb{R}^{n_z \times n_z}$ inside L2. Fixing n_z also defines the sizes of all remaining trainable parameters highlighted in orange in Figure 3. The result presented next assures the designer that this PWA-NN structure is suitable for any possible predictive controller.

Theorem 1 [31] (Given an appropriate size, the piecewise-affine Neural Network can learn any MPC controller of the form (7)). Let $\hat{\pi} : \mathcal{X} \to \mathcal{U}$ be the map defined by the composition of all four layers, i.e., $\hat{\pi}(x) := y_4 \circ y_3 \circ y_2 \circ y_1(x)$. Set $\epsilon = 0$, then $\exists F$, f, L, G and g with appropriate dimensions such that $\hat{\pi}(x) = \pi(x)$, $\forall x \in \mathcal{X}$.

The optimization problem associated with training this NN—in fact, almost any NN architecture—is non-convex. As a consequence, even though there might exist a combination of parameters and weights capable of exactly representing the desired function, reaching them is not an easy task. Since local minima exist, the training process has to be performed multiple times with different initializations. Nevertheless, it is reasonably accepted in the machine learning community that these loss functions possess many high quality local minima, and pursuing a global optimum is irrelevant in this context (see for instance the influential work [36]).

Theorem 1 establishes that the size of the NN could be chosen to exactly replicate $\pi(x)$, but that would defeat its purpose since the goal is to learn a faithful but *simpler* version of the MPC controller. For this reason, we gradually increased the size n_z during the training process until a desirable approximation quality was attained. From a machine learning perspective, the problem could be interpreted as an approximation one, where the ground-truth is known.

B. THE TRAINING PHASE AND OBTAINED RESULTS

The explicit control law $\pi(x)$ was sampled in order to collect a set of state-control pairs

$$\{(x_d, u_d) \,|\, d = 1, \ldots, D\} \tag{12}$$

where x_d can be regarded as features and u_d as labels. A total of $D = 5000$ points were gathered randomly using a uniform distribution over the set of feasible states. We highlight that the samples could have been acquired directly from (7) as well. Next, the data-points were used to train the internal parameters of the layers shown in Figure 3.

A standard backpropagation approach can be used to iteratively update the NN parameters since, as shown in [35], optimization layers of this type are differentiable (except on sets of measure zero, where subgradients can be used). The `PyTorch` and `OptNet` packages for Python were employed to code the NN and mini-batch stochastic gradient descent was used to train it. The batch size was chosen to be 50, and the whole dataset was presented to the algorithm a total of 150 times, i.e., 150 epochs. In order to achieve a balanced learning throughout the domain, the currents and voltages values that formed the input locations x_d were normalized to a range of [0,1]. Furthermore, all trainable weights were initialized randomly. The code was run on a 3.1 GHz Intel Core i7 laptop with 16 GB 2133 MHz of memory. As previously explained, we gradually increased the size n_z of the PWA-NN. Training the network once took approximately 35 mins without any GPU acceleration. With $n_z = 3$, after only 5 initializations, the network presented a very low mean squared error training loss: 1.66×10^{-7}. As for the testing phase, we calculated the true outputs $u = \pi(x)$ and the predicted values $\hat{u} = \hat{\pi}(x)$ on a grid of points; the latter were capable of closely reproducing the original controller as shown in the top plots of Figure 4.

In order to assess the complexity of the learned controller, its L2 layer was converted into a PWA function using the MPT toolbox. As can be seen from lower plots in Figure 4, the number of region was greatly reduced: from 70 in the original partition to 6 in the simplified one, a reduction of 91%. The total memory required to store the control law parameters was reduced from 9.25 kB to 528 B. The latter quantities were calculated by counting the total number of constants needed to describe all the inequalities that compose the polytopes and the remaining NN layers, and assuming that each of them occupies 1 *word* of space.

A final validation phase was carried out through simulating the system under the original MPC controller $\pi(x)$ and the learned controller $\hat{\pi}(x)$, starting from different initial conditions. The results are shown in Figure 5. As can be seen from the top and bottom plots, the closed-loop evolution was nearly identical with both controllers. This indicates that the learning procedure was able not only to learn the given dataset, but also generalize the function to other areas of the space. A small mismatch can be seen is the area close to $i_L = 0$, $v_O = 6$. This is however not a major concern since that region of the space is not visited during start-up, and the discrepancy causes the system to stay inside the feasible region rather than violating the constraints. An additional numerical investigation is reported in Appendix B.

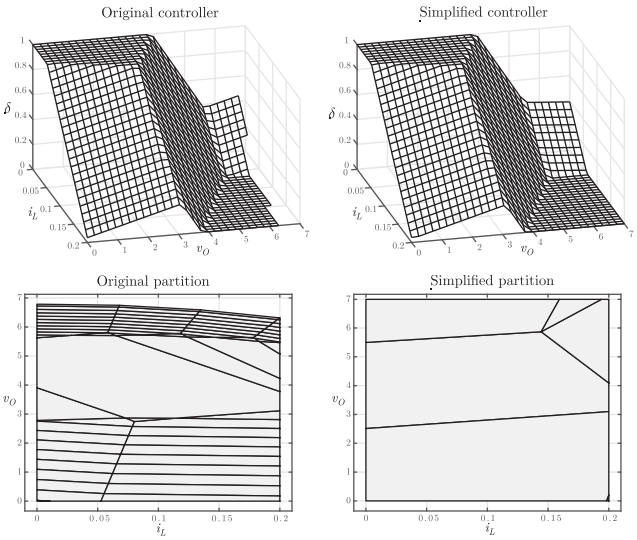

FIGURE 4. Original PWA domain partition and MPC control surface (left). PWA partition of the NN second layer and simplified control surface (right). With just a small fraction of the original number of regions, the final simplified controller can closely reproduce the original control surface.

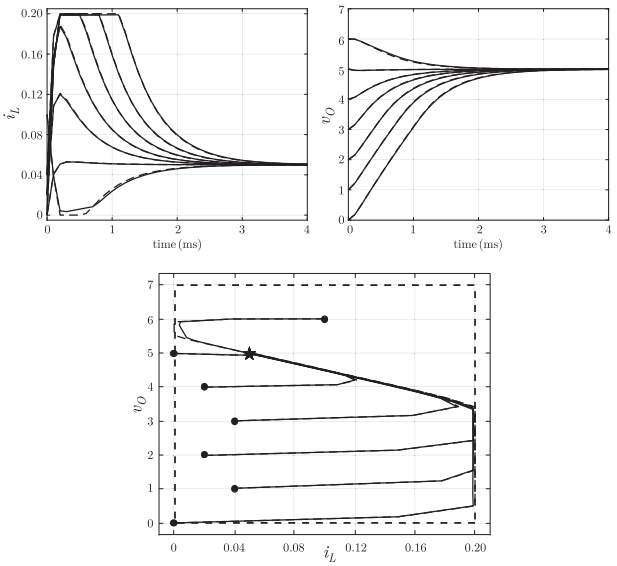

FIGURE 5. Simulation results: Current and output voltage time evolution (top plots), and phase portrait (bottom plot). The dashed lines represent the system evolution with the original EMPC controller, and the solid lines, with the simplified EMPC controller. In the bottom plot, the solid circles are the initial conditions, and the star depicts the target steady-state.

V. EXPERIMENTAL VALIDATION

A prototype of the buck converter is presented in Figure 6. The target embedded hardware was an STM32L476 platform: the MCU runs at 80 MHz, it has a 32-bit RISC architecture and two independent 12-bit ADC channels. The controller

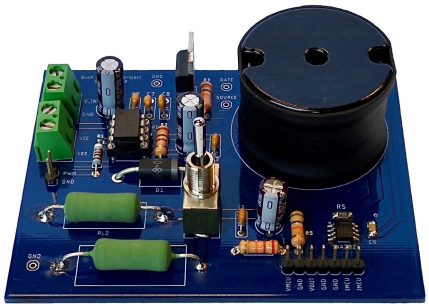

FIGURE 6. **A picture of the switched-mode electronic converter prototype.**

firmware was written in an interrupt-driven, bare metal, fashion and was triggered at 10 kHz. To avoid using raw measurements and also to filter noise, the ADCs were operated at a faster pace and 5 consecutive readings were averaged before being sent to the EMPC controller. Only integers were used to encode all control law parameters (polytopes and feedback gains) and the direct memory access peripheral was employed to link the ADC blocks directly to memory. All these steps were needed to ensure that all instructions could be executed within the 100 μs time window.

Even though both v_O and i_L were being measured, a state observer was used to estimate the two variables. The produced estimates were only used during the first two sampling instants after the control task was initiated, and the sensors readings were employed afterwards. This approach was crucial to counteract the high non-linearity caused by the diode barrier during the first time instants, which would lead otherwise to current constraint violations. Later in time, this effect caused by the diode was neglected because i_L was already high enough to quickly overcome the barrier.

For comparison purposes, the open-loop and closed-loop start-up responses of the power converter are shown in Figure 7. Whereas the open-loop behavior shows high output voltage and current overshoot, the closed-loop EMPC response was faster and satisfied the imposed constraints. More specifically, the settling time was reduced from approximately 6.73 ms to 2.33 ms; the voltage peak from 7.97 V to 5.16 V; and the current peak of the average signal from 333 mA to 202 mA. In Figure 8 (top) we see the same closed-loop curves, but without the use of the state observer. Due to the current 'delay' caused by the diode, the controller causes the inductor to accumulate enough energy to violate the constraints. Finally, a flag was risen to indicate the time needed to execute the EMPC interrupt service routine (ISR) that was responsible for: averaging the ADC samples, estimating the states when needed, solving the point location problem, calculating control input, and updating the PWM peripheral duty cycle. The execution period varied according to the current EMPC region, but lasted between 22.0 μs and 27.5 μs. As shown in Figure 8 (bottom), the MCU processor core was not always busy, but had time available to execute other tasks if requested by a particular application.

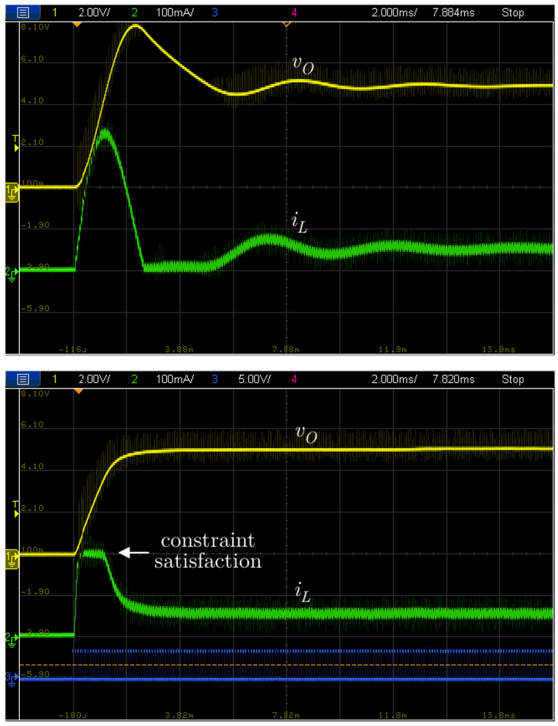

FIGURE 7. **Open-loop start-up response (top) and closed-loop EMPC start-up response (bottom). The output voltage settling-time was reduced from 6.73 ms to 2.33 ms, and the inductor current respected the imposed constraints.**

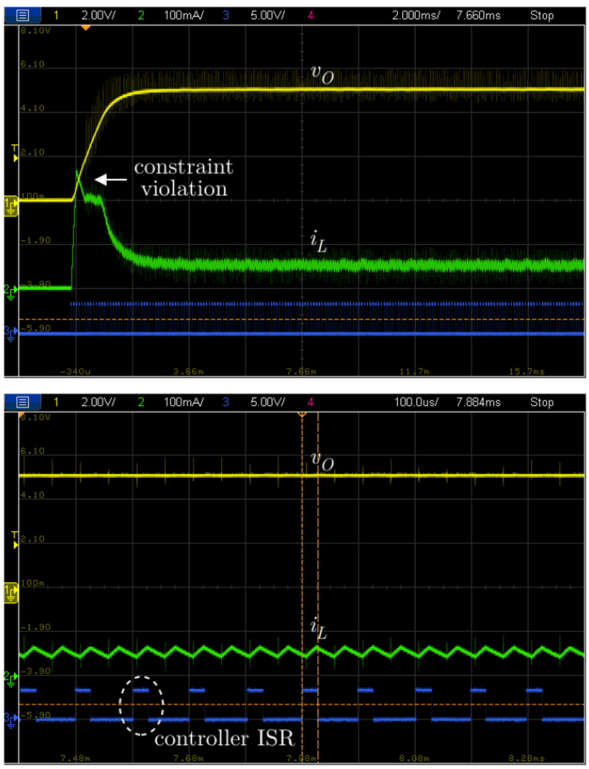

FIGURE 8. **Closed-loop start-up response with no state-observer (top) and the controller interrupt service routine taking approximately 27 μs (bottom).**

VI. CONCLUSION

The practical investigation reported here shows the PWA-NN potential in reducing the computational demand of MPC. This allows for predictive control to be embedded in inexpensive microcontrollers and enhance the dynamics of electronic power converters. Our experiments show that the current constraints were satisfied by the closed-loop system even after the controller simplification phase, which indicates that high-quality approximation results were attained. Future works could explore alternative neural network architectures that incorporate memory to learn MPC controllers, as well as applying these techniques to more complex converter topologies at higher current-voltage scales.

APPENDIX A: EQUATIONS FOR THE ON AND OFF SWITCH STATES

Note that the algebraic complexity of our model stems mainly from including many parasitic elements, which are normally neglected. When the switch is open, the following equations describe the buck converter dynamics

$$\dot{x}_1 = -\frac{R_L}{L}x_1 - \frac{1}{L}x_2 - \frac{V_D}{L} \tag{13}$$

$$\dot{x}_2 = -\frac{R_C R_O R_L C + R_O L}{(R_C + R_O)LC}x_1 - \frac{R_C R_O C + L}{(R_C + R_O)LC}x_2 - \frac{R_P V_D}{L} \tag{14}$$

where $R_P = (R_C R_O)/(R_C + R_O)$ is the parallel equivalent of the capacitor and load resistances. When the switch is closed one has

$$\dot{x}_1 = -\frac{R_{ON} + R_L}{L}x_1 - \frac{1}{L}x_2 + \frac{V_{IN}}{L} \tag{15}$$

$$\dot{x}_2 = -\frac{R_C R_O (R_{ON} + R_L)C + R_O L}{(R_C + R_O)LC}x_1 - \frac{R_C R_O C + L}{(R_C + R_O)LC}x_2 + \frac{R_P V_D}{L} \tag{16}$$

making the dynamics affine (linear with a constant bias).

APPENDIX B: ADDITIONAL SIMPLIFICATION RESULTS

Here we present additional results that illustrate the potential of the proposed technique. Firstly, the MPC controller parameters described in Section III were modified to $Q = \mathrm{diag}(1, 1)$, $R = 100$ and a horizon of $N = 100$. The predictive controller explicit form $\pi(x)$ had $M = 189$ regions. Next, *all* simplification techniques available on MPT were applied to $\pi(x)$—these can be accessed through the `simplify()` command. The results are shown in Table 2, where the validation error indicates the mean squared error calculated on a grid of 6561 points. As can be seen, most of the MPT techniques failed in simplifying $\pi(x)$ and essentially preserved the original number of regions. The 'fitting' approach however drastically reduced the number of regions with a reasonably low validation error.

TABLE 2. Additional Complexity Reduction Results for an Optimal Control Law $\pi(x)$ With 189 Regions

MPT method	Num. regions	Validation error
'orm' [37]	182	$< 10^{-30}$
'clipping' [38]	187	$6.31\,10^{-8}$
'greedy'	183	$< 10^{-30}$
'separation' [39]	187	$< 10^{-30}$
'fitting' [40]	5	$3.71\,10^{-4}$

PWA-NN size	Num. regions	Validation error
$n_z = 1$	2	$1.30\,10^{-3}$
$n_z = 2$	2	$2.90\,10^{-4}$
$n_z = 3$	5	$1.75\,10^{-4}$

We then collected $D = 5000$ samples from the MPC controller and trained several PWA-NNs with increasing size. For every size, the training process was reinitialized only 10 times and the best results are reported in Table 2. As n_z was increased, the resulting number of regions also increased. With $n_z = 3$, this extra flexibility was already enough to reduce the validation error to less than half of the 'fitting' method error with exactly the same number of regions.

REFERENCES

[1] V. Wisniewski, E. Maddalena, and R. Godoy, "Discrete-time regional pole-placement using convex approximations: Theory and application to a boost converter,"*Control Eng. Pract.*, vol. 91, 2019, Art. no. 104102.

[2] Y. Guan, T. Yao, Y. Wang, W. Wang, and D. Xu, "Analysis and design of a class E type high frequency DC/DC converter based on resonant driving circuit," *IEEE Open J. Ind. Electron. Soc.*, to be published, doi: 10.1109/OJIES.2020.3007437.

[3] Y. Liao, X. Wang, and F. Blaabjerg, "Passivity-based analysis and design of linear voltage controllers for voltage-source converters," *IEEE Open J. Ind. Electron. Soc.*, to be published, doi: 10.1109/OJIES.2020.3001406.

[4] E. T. Maddalena, C. G. da Silva Moraes, G. Bragança, L. G. Junior, R. B. Godoy, and J. O. P. Pinto, "A battery-less photovoltaic water-pumping system with low decoupling capacitance," *IEEE Trans. Ind. Appl.*, vol. 55, no. 3, pp. 2263–2271, May/Jun. 2019.

[5] A. Khan, M. B. Shadmand, S. Bayhan, and H. Abu-Rub, "A power ripple compensator for DC nanogrids via a solid-state converter," *IEEE Open J. Ind. Electron. Soc.*, to be published, doi: 10.1109/OJIES.2020.3035073.

[6] W. Chen, T. Liang, and V. Dinavahi, "Comprehensive real-time hardware-in-the-loop transient emulation of MVDC power distribution system on nuclear submarine," *IEEE Open J. Ind. Electron. Soc.*, to be published, doi: 10.1109/OJIES.2020.3036731.

[7] S. Vazquez, J. Rodriguez, M. Rivera, L. G. Franquelo, and M. Norambuena, "Model predictive control for power converters and drives: Advances and trends," *IEEE Trans. Ind. Electron.*, vol. 64, no. 2, pp. 935–947, Feb. 2017.

[8] S. Almér, S. Mariéthoz, and M. Morari, "Dynamic phasor model predictive control of switched mode power converters," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 1, pp. 349–356, Jan. 2015.

[9] S. Richter, S. Mariethoz, and M. Morari, "High-speed online MPC based on a fast gradient method applied to power converter control," in *Proc. Amer. Control Conf.*, 2010, pp. 4737–4743.

[10] S. Almér, D. Frick, G. Torrisi, and S. Mariéthoz, "Predictive converter control: Hidden convexity and real-time quadratically constrained optimization," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 1, pp. 403–410, Jan. 2021.

[11] P. Zometa, M. Kögel, T. Faulwasser, and R. Findeisen, "Implementation aspects of model predictive control for embedded systems," in *Proc. Amer. Control Conf.*, 2012, pp. 1205–1210.

[12] C. Xia, T. Liu, T. Shi, and Z. Song, "A simplified finite-control-set model-predictive control for power converters," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 991–1002, May 2014.
[13] T. Geyer and D. E. Quevedo, "Multistep finite control set model predictive control for power electronics," *IEEE Trans. Power Electron.*, vol. 29, no. 12, pp. 6836–6846, Dec. 2014.
[14] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Madison, WI, USA: Nob Hill Publishing, vol. 2, 2017.
[15] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear Model Predictive Control*, pp. 345–369, 2009.
[16] S. Mariéthoz and M. Morari, "Explicit model-predictive control of a PWM inverter with an LCL filter," *IEEE Trans. Ind. Electron.*, vol. 56, no. 2, pp. 389–399, Feb. 2009.
[17] A. G. Beccuti, S. Mariéthoz, S. Cliquennois, S. Wang, and M. Morari, "Explicit model predictive control of DC-DC switched-mode power supplies with extended Kalman filtering," *IEEE Trans. Ind. Electron.*, vol. 56, no. 6, pp. 1864–1874, Jun. 2009.
[18] S. Mariéthoz, A. Domahidi, and M. Morari, "Sensorless explicit model predictive control of permanent magnet synchronous motors," in *Proc. IEEE Int. Electric Machines Drives Conf.*, pp. 1250–1257, 2009.
[19] 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.
[20] C. N. Jones and M. Morari, "Polytopic approximation of explicit model predictive controllers," *IEEE Trans. Auto. Control*, vol. 55, no. 11, pp. 2542–2553, Nov. 2010.
[21] F. Bayat, T. A. Johansen, and A. A. Jalali, "Flexible piecewise function evaluation methods based on truncated binary search trees and lattice representation in explicit MPC," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 3, pp. 632–640, May 2012.
[22] T. Geyer, *Low Complexity Model Predictive Control in Power and Power*. Gottingen, Germany: Cuvillier Verlag, 2005.
[23] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. a data-driven control framework," *IEEE Trans. Robot. Autom.*, vol. 63, no. 7, pp. 1883–1896, Jul. 2017.
[24] E. T. Maddalena, P. Scharnhorst, Y. Jiang, and C. N. Jones, "KPC: Learning-based model predictive control with deterministic guarantees," 2020, *arXiv:2011.11303*.
[25] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annu. Revi. Control, Robot., Auto. Syst.*, vol. 3, pp. 269–296, 2020.
[26] S. Chen *et al.*, "Approximating explicit model predictive control using constrained neural networks," in *Proc. Annu. Amer. Control Conf.*, 2018, pp. 1520–1527.
[27] X. Zhang, M. Bujarbaruah, and F. Borrelli, "Near-optimal rapid MPC using neural networks: A primal-dual policy learning framework," *IEEE Trans. Control Syst. Technol.*, to be published, doi: 10.1109/TCST.2020.3024571.
[28] S. Lucia, D. Navarro, B. Karg, H. Sarnago, and O. Lucia, "Deep learning-based model predictive control for resonant power converters," *IEEE Trans. Ind. Informat.*, vol. 17, no. 1, pp. 409-420, Jan. 2021.
[29] R. Winqvist, A. Venkitaraman, and B. Wahlberg, "On training and evaluation of neural network approaches for model predictive control," 2020, *arXiv:2005.04112*.
[30] C. Cervellera, D. Macciò, and T. Parisini, "Learning robustly stabilizing explicit model predictive controllers: A non-regular sampling approach," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 737–742, Jul. 2020.
[31] E. T. Maddalena, C. d. S. Moraes, G. Waltrich, and C. N. Jones, "A neural network architecture to learn explicit MPC controllers from data," 2019, *arXiv:1911.10789*.
[32] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking control on a robot manipulator: Robust MPC and neural network control," *IEEE Trans. Robot. Autom.*, vol. 5, no. 2, pp. 3050–3057, 2020.
[33] J. A. Paulson and A. Mesbah, "Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 719–724, Jul. 2020.
[34] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *Proc. Eur. Control Conf.*, 2013, pp. 502–510.
[35] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 136–145.
[36] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2015, pp. 192–204.
[37] T. Geyer, F. D. Torrisi, and M. Morari, "Optimal complexity reduction of polyhedral piecewise affine systems," *Automatica*, vol. 44, no. 7, pp. 1728–1740, 2008.
[38] M. Kvasnica and M. Fikar, "Clipping-based complexity reduction in explicit MPC," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1878–1883, Jul. 2012.
[39] M. Kvasnica, J. Hledík, I. Rauová, and M. Fikar, "Complexity reduction of explicit model predictive control via separation," *Automatica*, vol. 49, no. 6, pp. 1776–1781, 2013.
[40] B. Takács, J. Holaza, M. Kvasnica, and S. Di Cairano, "Nearly-optimal simple explicit MPC regulators with recursive feasibility guarantees," in *Proc. 52nd Conf. Decis. Control*, 2013, pp. 7089–7094.

EMILIO T. MADDALENA received the B.S. degree in electrical engineering from Universidade Federal de Mato Grosso do Sul, Campo Grande, Brazil and the M.S. degree in systems and control from Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil, in 2016 and 2018, respectively. He was also a Visiting Student with Politecnico di Torino, Torino, Italy. Since 2018, he has been a Doctoral Assistant with the École Polytechnique fédérale de Lausanne, Lausanne, Switzerland. His research interests include safe learning and model predictive control techniques, and their applications to electronic energy conversion and building air conditioning systems.

MARTIN W. F. SPECQ received the B.S. degree in microengineering from École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 2019. From 2018 to 2019, as part of an exchange program, he studied one academic year with Technische Universität Berlin, Berlin, Germany. He is currently working toward the master's degree in robotics with EPFL. He is a Teaching Assistant for electronics and mathematical analysis courses with EPFL. His research interest include control systems, electronics, embedded systems, and mobile robotics.

VIVIANE L. WISNIEWSKI received the B.S. degree in electrical engineering from Universidade Federal de Mato Grosso do Sul, Campo Grande, Brazil, in 2018. She is currently working toward the master's degree in the engineering program, energy and environment with Berner Fachhochschule, Bern, Switzerland. She was a Research Intern with the Artificial Intelligence, Power Electronics and Digital Systems Laboratory, BATLAB, Campo Grande, Brazil. Her research interest include high-efficiency power converters, control systems, sustainability, and renewable energies.

COLIN N. JONES (Member, IEEE) received the bachelor's and master's degrees in electrical engineering and mathematics from the University of British Columbia, Vancouver, Canada and the Ph.D. degree in 2005 from the University of Cambridge, Cambridge, U.K, for his work on polyhedral computational methods for constrained control. Since 2017, he has been an Associate Professor with Automatic Control Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland and since 2011, he has been an Assistant Professor. He was a Senior Researcher with the Automatic Control Lab, ETH Zurich, Zürich, Switzerland, until 2010. He has authored or coauthored more than 200 publications and was awarded an ERC starting grant to study the optimal control of building networks. His current research interests include high-speed predictive control and optimization, and the control of green energy generation, distribution and management.