Real-Time Implementation of Predictive Control in Power Inverters Based on Nearest Neighbor Searching

Joaquin G. Ordonez, Pablo Montero-Robina, *Member, IEEE*, Daniel Limon, and Francisco Gordillo

***Abstract*—Model predictive control is a powerful methodology to control multilevel power inverters because of its ability to deal with multiple variables and control objectives but presents some difficulties, such as high computational burden and the fact that the control input is held constant during each sampling period. Multirate model predictive control was later proposed to alleviate the latter issue but would become even more complex to compute. This work proposes the implementation of multirate model predictive control using a novel adaptation of nearest neighbor searching to learn the control law. This effort made it possible for this control technique to be implemented in real time by taking advantage of hardware-accelerated parallelization. The method is tested on a five-level diode-clamped converter operating in inverter mode. The algorithm was implemented in a field-programmable-gate-array-in-the-loop experiment, and results show a significant reduction in total harmonic distortion, improving current ripples compared to predictive control that does not use the multirate technique.**

***Index Terms*—Hybrid systems, machine learning, modeling and prediction, real-time and embedded systems, real-time control.**

I. Introduction

THE importance of power converters has been growing in recent years as a result of their necessity in the transition to renewable energy sources. In this line, multilevel converters have been a research focus as they present lower current distortion and, thus, are more energy efficient, at the expense of circuit configuration complexity, drawing attention to newer control techniques. Model predictive control (MPC) is very appealing in the field of power electronics because it can easily deal with multiple control objectives that are subject to multivariable models and complex constraints [1], which is the case for these multilevel converters.

Manuscript received 9 February 2023; revised 12 June 2023 and 19 August 2023; accepted 30 September 2023. Date of publication 11 October 2023; date of current version 6 December 2023. This work was supported by MCIN/AEI/10.13039/501100011033 under Grant PID2019-109071RB-I00. Additionally, Grants PID2019-106212RB-C41 by MCIN/AEI/ 10.13039/501100011033 and P20-01116 by Junta de Andalucía-FEDER also supported part of the infrastructure of the experiments. Recommended for publication by Associate Editor M. A Perez. *(Corresponding author: Joaquin G. Ordonez.)*

Joaquin G. Ordonez, Daniel Limon, and Francisco Gordillo are with the Department of System Engineering and Automation, Universidad de Sevilla, 41092 Seville, Spain (e-mail: jgordonez@us.es; dlm@us.es; gordillo@us.es).

Pablo Montero-Robina is with the Autel Iberia, 08500 Barcelona, Spain (e-mail: pablo.montero@autel.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TPEL.2023.3323687.

Digital Object Identifier 10.1109/TPEL.2023.3323687

From the point of view of control system theory, these power converters are switched systems: Currents and voltages are continuous, but control inputs are discrete signals with a finite number of possibilities. Most inverter control design approaches are based on the consideration of control inputs as continuous signals, followed by a discretization stage, called modulation. Two of the main families of modulation techniques are carrier-based pulsewidth modulation (CB-PWM) and space vector modulation (SVM) [2]. Nevertheless, there exist other techniques that do not consider control inputs as continuous signals and take into account, at the outset, the discrete nature of the control input, such as hybrid control [3] or finite-control-set MPC (FCS-MPC) [4], [5]. Although FCS-MPC is well suited for discrete problems, this technique has also been mixed with continuous-time signals in an attempt to emulate modulation techniques, as reviewed in [6]. When multilevel inverters come into play, the methods that include a modulation stage have to pay special attention to an additional objective—the capacitor voltage balance. In the case of MPC-based control, this additional objective can be handled in a natural and direct way by including it in the cost function.

Avoiding the modulation phase can simplify the control algorithm, but it also brings up the main disadvantage of FCS-MPC: The control solution is presented as an input signal to the system that is held constant during the commutation period, usually matching the sampling period. This is not the case for control techniques that use a modulation stage that allows changes in the control action at any instant during the sampling interval. In FCS-MPC, this fact can be overcome at the cost of increasing the commutation frequency, but there are several limitations to doing so: 1) more commutations lead to more electrical losses and thermal stress on the device [7], and 2) sampling and control might not be possible to perform at such high frequencies when they are matched to the commutation frequency, which is a common practice to obtain a better harmonic spectrum [8]. The first issue (thermal limitation) can be avoided by using longer control horizons and commuting fewer times but at better moments, but, in turn, increases computational complexity [9]. The second issue is a direct computational limitation, which motivated many efforts directed to the efficient programming of the algorithm to reduce computational burden [10], including those who seek to calculate the control law over long prediction horizons [11].

TABLE I
MAIN TYPES OF MACHINE LEARNING METHODS COMMONLY USED IN POWER ELECTRONICS

Method	Applications	Memory	Calc. time	Uncert.
Neural Networks	Design, Control, Maintenance	Low	Low	High
Probabilistic	Maintenance	Low	Medium High	Low
Kernel	Control, Maintenance	Medium High	Low Medium	Low

Qualitative comparisons in typical memory usage, calculation time, and uncertainty.

Our contribution to this line began with the application of multirate MPC (MMPC or FCS-MMPC), first presented in [12] as an enhanced technique over the FCS-MPC algorithm to allow several commutations within each sampling period. It was a solution to the second issue when the constraint was the sampling frequency, but it increased the computation time even further. Nevertheless, the conclusion of that research was that FCS-MMPC can achieve great performance and solve the thermal limitation with a proper control horizon. In this case, the only downside would be the huge computational complexity. This is why more research has focused on machine learning techniques to implement the FCS-MMPC solution online [13].

There are many examples of artificial intelligence in power electronics [14]. While most applications are used for maintenance and fault diagnosis, machine learning is widely used to accelerate the computation of a control solution [15], [16]. Recent works commonly use neural networks to learn the control law, displaying promising results while validating the method in simulation [17], [18] or in a hardware-in-the-loop setup [19], [20]. Table I presents a simplified classification of machine learning works in this field. Neural networks and kernel methods are commonly used in control applications. While neural networks are usually easier on the hardware specifications for implementation, kernel methods generally provide better tracing and interpretability in their solution, resulting in more robust control. Another study [21] on the control of cascaded H-bridge inverters found similar conclusions, where the complexity of neural networks was not attractive when kernel methods were simpler and yielded closer performance results to controllers based on FCS-MPC.

In this article, the contribution is given on several fronts. Not only is the first real-time implementation in hardware of the FCS-MMPC solution, but it is also achieved by a new supervised machine learning technique—an adaptation of k-nearest neighbors (k-NN) optimized for power inverters that seeks the best solution within a database in two quick steps. The new kernel method was named 2-step nearest neighbor searching. This is accompanied by field-programmable gate array (FPGA)-in-the-loop experiments and results, which were achieved due to an optimized implementation structure that uses parallel computing to accelerate the searching algorithm.

The main disadvantage of kernel methods is the presence of a training dataset that needs to be accessed in real time. However, this issue is currently becoming less severe due to the broad availability of high-speed memory in embedded control hardware. Our chosen searching algorithm, compared with other kernel techniques, has been proven to have very low and deterministic computation time, fixed and adjustable approximation error, reasonable memory usage that fits in commercial FPGA platforms, and adaptability to different working conditions.

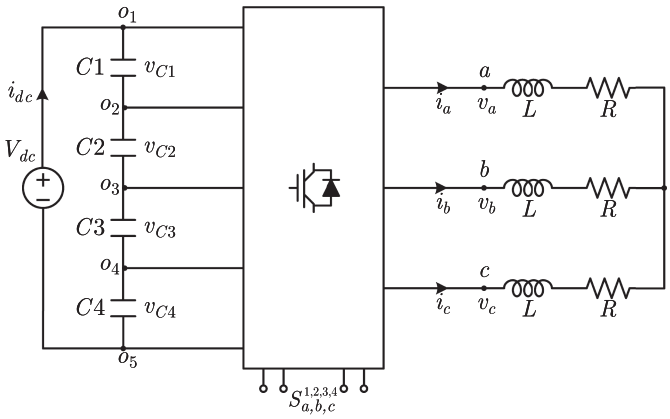

Fig. 1. Diagram of the three-phase, five-level DCC operating in inverter mode, connected to a DC source.

The rest of this article is organized as follows. In Section II, a multilevel inverter is presented. In Section III a new kernel method based on nearest neighbor searching, is used to accelerate the calculation of the solution of FCS-MMPC, allowing its application at high frequencies. In Section IV, the parallel computation scheme that allows the real-time implementation is provided. In Section V, the method is validated in a hardware-in-the-loop setup using an FPGA. The power inverter selected for the control experiment is a five-level diode-clamped converter (DCC), first detailed in Section II, along with its control strategy. Finally, Section VI concludes this article.

II. SYSTEM, MODELING, AND CONTROL

In this section, a multilevel inverter will be presented for the application of this article. The chosen topology is a five-level DCC, which carries high control complexity and the presence of nonsymmetric elements. Some guidelines for modeling will be given, as well as an introduction to FCS-MPC, even if it is out of the scope of this article. These will serve to introduce the functioning and variables of the system a better explanation of the learning method.

A. Five-Level DCC Power Inverter

The inverter considered in this article is a three-phase five-level DCC connected to a weak grid modeled as a three-phase inductive and resistive load, as shown in Fig. 1. The converter is fed from a dc source that could be any element with a dc link at the output. The central block in this figure represents the switching circuit depicted in Fig. 2. The converter parameters are the inductor L that is merged with the grid inductance, the capacitance of the capacitors that are assumed to be identical $C1 = C2 = C3 = C4 = C$, and the symmetric three-phase resistive load R.

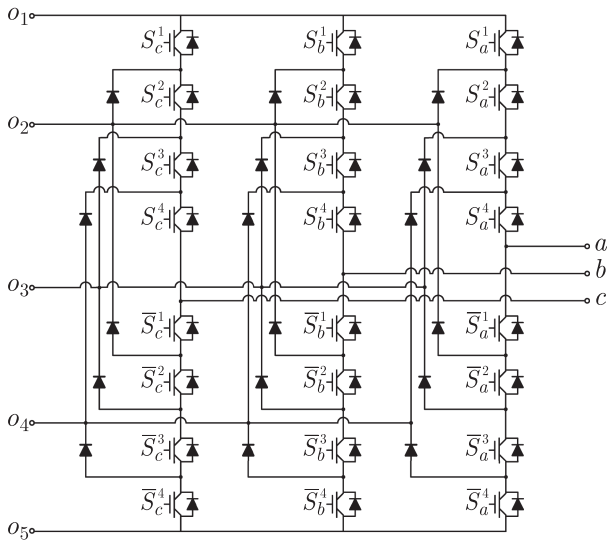

Fig. 2. Three-phase, five-level DCC topology.

TABLE II
FEASIBLE COMBINATIONS FOR THE CONTROL INPUT

f_{pj}	S_p^1	S_p^2	S_p^3	S_p^4	u_p	v_p
$f_{p5}=1$	0	0	0	0	-2	$-v_{C3}-v_{C4}$
$f_{p4}=1$	0	0	0	1	-1	$-v_{C3}$
$f_{p3}=1$	0	0	1	1	0	0
$f_{p2}=1$	0	1	1	1	1	v_{C2}
$f_{p1}=1$	1	1	1	1	2	$v_{C1}+v_{C2}$

The control inputs u_p are the binary position of the switching elements that determine the evolution of the variables of the converter system, that is, the currents i_p, for each phase $p=\{a,b,c\}$. Five switching states f_{pj}, for $j=1,\ldots,5$, are considered for each phase as defined in Table II. At every instant, for each phase, one and only one f_{pj}, for $j=1,\ldots,5$, is equal to 1 while the rest is equal to zero. Variables S_p correspond to the switching devices represented in Fig. 2. The second-to-last column shows the five possible values for the control input u_p. In a five-level three-phase system, the control input has a total of $5^3=125$ possible combinations. The last column in Table II is the corresponding voltage output v_p measured between the points a,b,c, and o_3 in Fig. 2, where v_{C1} to v_{C4} are the voltages measured at each capacitor.

By switching between levels and inducing v_p, currents i_p can be driven. These will affect the currents going from $o_1,\ldots,o_5$ to the switching block, which will charge or discharge the capacitors $C1$ to $C4$, modifying the voltage of the levels of the inverter and affecting its balancing. This relation will be explored further in the following section.

B. Modeling

It is easier to understand the inverter system as two sides, dc and ac, that interact with each other through a box of switching transistors. The system can be modeled in a similar way, where the equations reflect how the variables on each side are related to the level selected in the switching circuit.

The dynamic equations for the ac side are

$$v_p(t) = R\, i_p(t) + L\,\frac{di_p(t)}{dt} \tag{1}$$

for each phase $p=\{a,b,c\}$, where i_p are the ac-side phase currents and v_p is the output voltage given in Table II that depends on the level of operation u_p selected for each phase.

The model described with i_p and u_p can be translated from continuous to discrete time integrating (1) from $t=kT_s$ to $t=(k+1)T_s$, where T_s is the sampling time, as explained in [8]. This obtains a linear time-invariant (LTI) state-space model that can be written in form

$$i(k+1) = A\, i(k) + B\, u(k) \tag{2}$$

where $i\in\mathbb{R}^3$ is equivalent to i_p for $p=\{a,b,c\}$, and $u\in\{-2,-1,0,1,2\}^3$ to u_p, but the subindices p are dropped from now on for clarity when using the discrete form. Matrices $A,B\in\mathbb{R}^{3\times3}$ only depend on system parameters as follows:

$$A = \left(1-\frac{R\,T_s}{L}\right)\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \tag{3}$$

$$B = \frac{V_{\rm dc}\,T_s}{4\,L}\begin{bmatrix}2/3&-1/3&-1/3\\-1/3&2/3&-1/3\\-1/3&-1/3&2/3\end{bmatrix}. \tag{4}$$

The equations for the capacitors on the dc side are

$$C\frac{dv_{C1}}{dt} = i_{\rm dc} - \sum_p f_{p1}\, i_p \tag{5a}$$

$$C\frac{dv_{C2}}{dt} = i_{\rm dc} - \sum_p (f_{p1}+f_{p2})\, i_p \tag{5b}$$

$$C\frac{dv_{C3}}{dt} = i_{\rm dc} + \sum_p (f_{p4}+f_{p5})\, i_p \tag{5c}$$

$$C\frac{dv_{C4}}{dt} = i_{\rm dc} + \sum_p f_{p5}\, i_p\,. \tag{5d}$$

It is common to use the difference in voltage between capacitors as the variable to control, as it allows the reduction of the number of variables by 1 since $v_{C1}+v_{C2}+v_{C3}+v_{C4}=V_{\rm dc}$ at all times. Thus, $v_d\in\mathbb{R}^{n_C-1}$ is

$$v_d = \begin{bmatrix}v_{C1}-v_{C4}\\v_{C2}-v_{C3}\\v_{C3}-v_{C4}\end{bmatrix} \tag{6}$$

and n_C is the number of capacitors that split $V_{\rm dc}$.

Differentiating (6) with respect to time and combining it with (5), the dynamic equation of the capacitor voltage difference is

$$C\frac{dv_d}{dt} = \begin{bmatrix}\sum_p(-f_{p1}-f_{p5})\, i_p\\ \sum_p(-f_{p1}-f_{p2}-f_{p4}-f_{p5})\, i_p\\ \sum_p f_{p4}\, i_p\end{bmatrix}. \tag{7}$$

Since these equations are nonlinear due to the multiplication of f_{pj} and i_p and, thus, u_p and i_p (see Table II), an exact LTI

state-space equivalent cannot be obtained. Instead, the following nonlinear model

$$v_d(k+1) = v_d(k) + G\left(u(k)\right)\, i(k) \tag{8}$$

can be used, where $G : \mathbb{R}^3 \to \mathbb{R}^{(n_C-1)\times 3}$ is a function that depends on the control inputs and returns three column vectors, one for each phase $p = \{a, b, c\}$, where

$$G_p\left(u(k)\right) = \begin{cases} \gamma \begin{bmatrix} -1 & -1 & 0 \end{bmatrix}^\intercal & \text{if } u_p(k) = -2 \\ \gamma \begin{bmatrix} 0 & -1 & 1 \end{bmatrix}^\intercal & \text{if } u_p(k) = -1 \\ \gamma \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\intercal & \text{if } u_p(k) = 0 \\ \gamma \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}^\intercal & \text{if } u_p(k) = 1 \\ \gamma \begin{bmatrix} -1 & -1 & 0 \end{bmatrix}^\intercal & \text{if } u_p(k) = 2 \end{cases} \tag{9}$$

where $\gamma = T_s/C$.

Any controller in a functioning five-level inverter has to consider at least one objective for each of these systems of equations: on the ac side, the three-phase current has to meet some quality conditions, and on the dc side, the voltage differences between capacitors have to be kept within a threshold [22].

C. Finite Control Set MPC

In this section, MPC will be briefly introduced along with the finite-control-set method and the multirate variant, which will be used as an example for the learning process later on in this article.

Following the concepts of MPC in [23], we seek a control action that minimizes a cost function while satisfying several constraints. In power electronics that have a limited number of operations to select for the control action, FCS-MPC is a suitable technique. It eliminates the need for a complex optimization algorithm for the minimization problem, and instead, the cost function is checked a finite number of times, corresponding to every possible operation level, and the control action with the minimum cost is selected. This control process is repeated periodically every sampling period T_s: At each sample time k, the following holds.

1) The variables for the state of the system are measured.
2) The predicted cost function is estimated when applying every combination of the finite set of control actions.
3) The control action that carries the minimum cost is applied.

In order to differentiate between measured and predicted values, we will use variables i, u, v_d for measured state of the system, and $\bar{i}, \bar{u}, \bar{v_d}$ for internal predicted values in the optimization problem, which has the following mathematical expression:

$$\begin{aligned} \min_{\bar{u}} \quad & \sum_{j=0}^{N-1} \lambda_1 f_1\left(\bar{i}(j+1), i_r\right) \\ & + \lambda_2 f_2\left(\bar{v_d}(j+1), \bar{v_d}(j), \bar{u}(j)\right) \\ & + \lambda_3 f_3\left(\bar{u}(j), \bar{u}(j-1), \bar{i}(j)\right) && \text{(10a)} \\ \text{s.t.} \quad & \bar{i}(j+1) = A\,\bar{i}(j) + B\,\bar{u}(j) && \text{(10b)} \\ & \bar{v_d}(j+1) = \bar{v_d}(j) + G\left(\bar{u}(j)\right)\,\bar{i}(j) && \text{(10c)} \\ & \bar{u}(j) \in \{-2,-1,0,1,2\}^3 && \text{(10d)} \\ & \quad j = 0, 1, \ldots, N-1 \\ & \bar{i}(0) = i(k) && \text{(10e)} \\ & \bar{v_d}(0) = v_d(k) && \text{(10f)} \\ & \bar{u}(-1) = u(k-1)\,. && \text{(10g)} \end{aligned}$$

In this formulation, the cost function (10a) reflects the sum of three function terms that correspond to each of the control objectives. In the case of multilevel diode-clamped power inverters, these include at least the terms for current tracking and capacitor balancing. Tracking cost, represented as f_1, is a term that penalizes the deviation between the predicted current $\bar{i} \in \mathbb{R}^3$ and the current reference $i_r \in \mathbb{R}^3$. A larger tracking error contributes to total harmonic distortion (THD). The cost f_2 has the purpose of keeping the capacitors balanced and generally penalizes $\bar{v_d} \in \mathbb{R}^{n_C-1}$ deviation from 0. Furthermore, the decision variable $\bar{u} \in \mathbb{Z}^3$ can appear in the cost function as f_3 to penalize commutations, since these are directly related to electric losses dissipated as heat. Factors λ_1, λ_2, and λ_3 are weights that tune the importance of the three terms in the cost function. The solution to the optimization problem is $u(k) = \bar{u}(0)$. The constraints are the grid (10b) that predicts the grid currents $\bar{i}(j)$, from $j > 0$, the capacitor balancing equations (10c) that predict the future voltage balance $\bar{v_d}(j)$, from $j > 0$, and the control input constraints (10d). This problem receives the initial conditions $i(k) \in \mathbb{R}^3$ and $v_d(k) \in \mathbb{R}^3$ from measurements at every sampling instant, $u(k-1) \in \mathbb{Z}^3$ from the previous optimization problem, and the current reference $i_r \in \mathbb{R}^3$. This problem is solved at a frequency of $1/T_s$.

The control horizon N indicates how many steps are planned for the future to minimize the predicted cost. However, only the first step is applied so that the problem is recalculated in the next step. This is the receding-horizon strategy. In FCS-MPC, the cost function is calculated a finite number of times, corresponding to the possible operation levels available, which are 125^N in the case of three-phase five-level inverters. It is easy to see that computational burden increases dramatically with N. Considering the high sampling frequencies required to control this type of system, the receding-horizon strategy can be just impossible to implement without a heavy optimized method to find the solution.

Furthermore, a quick calculation of the optimization problem can allow higher control input frequencies, which is another important factor in improving reference tracking and lowering THD. In a closed loop, there are more limiting elements to the control input frequency, such as the frequency of the measuring stage or the switching elements. When the limiting factor is the measuring hardware or the calculus time, a multirate strategy can be very useful for reducing harmonic distortion.

D. Multirate MPC

Multirate MPC (MMPC) [12] is a variant of MPC that allows a higher control input frequency while maintaining the sampling

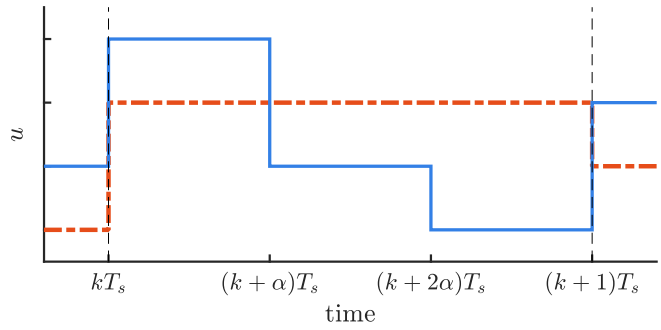

Fig. 3. Example of standard MPC (dashed line) and multirate MPC (continuous line) control action u for one phase, with $N_\alpha = 3$, during a sampling period from kT_s to $(k+1)T_s$.

frequency. This technique aims to improve the performance of the converter when the control time of the FCS-MPC is constrained by hardware limitations such as the sampling rate of the acquisition system and the calculus time, but not by the commutation frequency of the switching devices.

The main change of MMPC is on the variable $u(k) \in \mathbb{R}^{3\times N_\alpha}$, which is now a matrix with 3 rows, one for each phase $\{a, b, c\}$, and N_α columns, because, in this technique, the control action can change N_α times during the same sampling interval. Only the first split is applied in closed-loop control with system readings while the rest $N_\alpha - 1$ splits are estimated based on the measurements performed at the beginning of the sampling interval. For any integer $N_\alpha > 1$, we define $\alpha = 1/N_\alpha$ such that the sampling interval is divided into N_α subintervals with boundaries $k\,T_s, (k+\alpha)\,T_s, (k+2\alpha)\,T_s, \ldots, (k+1)\,T_s$. An example of $N_\alpha = 3$ is illustrated in Fig. 3. The selection of N_α depends on computation time and switching frequency restrictions.

The optimization problem for FCS-MMPC is similar to (10) but updated to account for the subintervals. Its description is

$$\min_{u} \quad \sum_{j=0}^{N\cdot N_\alpha - 1} \lambda_1 f_1\left(\bar{i}(j+1), i_r\right) + \lambda_2 f_2\left(\bar{v_d}(j+1), \bar{v_d}(j), \bar{u}(j)\right) + \lambda_3 f_3\left(\bar{u}(j), \bar{u}(j-1), \bar{i}(j)\right) \tag{11a}$$

$$\text{s.t.} \quad \bar{i}(j+1) = A_\alpha\, \bar{i}(j) + B_\alpha\, \bar{u}(j) \tag{11b}$$

$$\bar{v_d}(j+1) = \bar{v_d}(j) + G_\alpha\left(\bar{u}(j)\right)\, \bar{i}(j) \tag{11c}$$

$$\bar{u}(j) \in \{-2, -1, 0, 1, 2\}^3 \tag{11d}$$

$$j = 0, 1, \ldots, N\cdot N_\alpha - 1$$

$$\bar{i}(0) = i(k) \tag{11e}$$

$$\bar{v_d}(0) = v_d(k) \tag{11f}$$

$$\bar{u}(-1) = u_m\,. \tag{11g}$$

The FCS-MMPC algorithm is solved at each sample time kT_s with the information available at that instant and gives as solution $u(k) = [\bar{u}(0), \bar{u}(1), \bar{u}(2), \ldots, \bar{u}(N_\alpha - 1)]$ that must be applied at times $k\,T_s, (k+\alpha)\,T_s, (k+2\alpha)\,T_s, \ldots, (k+N_\alpha\alpha-\alpha)\,T_s$, respectively. One small difference with respect to the previous formulation (10) is the introduction of the term $u_m \in \mathbb{R}^3$, which is not exactly $u(k-1) \in \mathbb{R}^{3\times N_\alpha}$—only its last column (N_α), corresponding to the control action in the last subinterval. Matrices A_α, B_α, and function G_α depend on the system and must be modified to account for the multirate variant, where the only change is that where T_s appears in the definition of those parameters, it is replaced by $\alpha\, T_s$.

Similarly to the receding-horizon strategy, FCS-MMPC scales the computational burden exponentially with N_α, making this technique almost impossible to implement as is. Even when dodging measurement limitations, the biggest remaining concern is the computational time of predictive control. The next section will address this issue by proposing a machine learning implementation of this control technique.

III. Two-Step Nearest Neighbor Searching Methodology

As stated earlier, the optimization problem of the FCS-MMPC is practically not possible to compute online in real-time implementation. This is the motivation for most of the work in this article, which pursues an explicit description of the resulting control law based on machine learning with the objective of online implementation. In other words, we seek a learned solution that requires low computational effort while being as accurate as possible with respect to the actual control law.

As discussed earlier in the introduction, kernel methods in machine learning provide interpretability and adaptability to achieve satisfactory results for power converter applications. Previous work on these methods included linear regression and nonparametric methods [13] that encompass Lipschitz interpolation, allowing for easier theoretical analysis [24], as well as being simple enough to be run in embedded systems [25]. The technique chosen in this article, nearest neighbor searching, carries these important properties while simplifying computational complexity by skipping the interpolation stage. This is possible due to the discrete nature of the control law, as the solution of the optimization problem can be stored as a single discrete value. Additionally, by removing the interpolation work, more circuit resources can be put into parallelizing the solution searching.

The goal of this section is, thus, to present this adaptation of the nearest neighbor searching methodology to control power inverters by learning the MPC solution. The adaptation makes it easier and faster to locate the closest neighbor by taking advantage of the periodic nature of the output ac current to optimize the searching algorithm. In fact, due to this characteristic, the implementation proposed in this section is applicable to other topologies of power inverters and other control techniques.

First, the content of this section will focus on obtaining a suitable database to learn the FCS-MMPC control strategy presented in this article, including data processing methods. Then, nearest neighbor searching will be presented along with its 2-step adaptation for power inverters.

A. Data Collection

There are two key aspects to consider when building a database to learn the control of a system. The first is to select the appropriate variables that define the state of the system and the control solution associated with that state. The second is the length of the database, the number of different states that will be gathered to cover a whole range of situations that can occur during the control of the system.

Learning the control law of FCS-MMPC starts with examining the system modeling and the optimization problem employed in Section II. The general optimization problem outputs a solution $u(k)$ that minimizes a predicted cost function while receiving the initial variables $i(k)$, $v_d(k)$, the previous solution $u(k-1)$, and the reference $i_r(k)$. Thus, a database $\mathcal{D}$ for learning the control is a static map $y = F(w)$, in which $y \in \mathcal{Y}$ gathers $u(k)$ has $3 \times N_\alpha$ dimensions due to the 3 phases and the number N_α of commutations within the sampling interval. The input $w \in \mathcal{W}$ also depends on N_α, and the space $\mathcal{W}$ has the following form and dimensions in the database:

$$\begin{bmatrix} \overbrace{i(k,1)}^{3} & \overbrace{v_d(k,1)}^{3} & \overbrace{u(k-1,1)}^{3\times N_\alpha} & \overbrace{i_r(k,1)}^{3} \\ \vdots & \vdots & \vdots & \vdots \\ i(k,N_\mathcal{D}) & v_d(k,N_\mathcal{D}) & u(k-1,N_\mathcal{D}) & i_r(k,N_\mathcal{D}) \end{bmatrix}$$

in which the number of columns is indicated above the brackets and the number of rows corresponds to the number of data samples $N_\mathcal{D}$.

With the appropriate variables selected, the second part is to analyze all the different control scenarios that can occur to register a proper range of states. The fact that the inverter is controlled to obtain a three-phase sinusoidal current that is a periodic signal simplifies data collection, since only a few periods of the signal are needed in steady state. However, the transient is more complicated because it depends on the initial conditions.

These initial conditions may vary from experiment to experiment, as the capacitors need to be charged before the inverter starts to operate. This stage is usually not a perfect process and does not guarantee that the capacitors are balanced by the time the control begins. However, the differences among them are expected to be bounded because of the switch voltage limits. To achieve a comprehensive database for any reasonable initial state, it is recommended to simulate every possible combination of signs for the initial value of the voltage difference variables (6) until steady state is reached.

In the application that is the object of the article, in which the conventional control law cannot be implemented online, data collection can be carried out in a simulated environment.

B. Data Processing

Although numerous simulations can be performed to collect data in a large and rich database, the computational cost of the search algorithm increases with the length $N_\mathcal{D}$ of the set. In addition, fast memory for high-frequency control is very limited in embedded hardware. For these reasons, several common procedures in the machine learning field will be proposed to ease computing efforts and simplify the database. These processes are briefly described ahead.

1) *Normalization:* For embedded implementations, it is recommended to subtract the mean value of each column of the database and then scale each column so that the maximum absolute value is 1. Although this is calculated offline for the database, normalization has to be performed online for every new input query, but as this step only uses sums and multiplications, it is quick while also being very useful for working with fixed-point number storing.
2) Basis transformation by principal component analysis (PCA) [26]. This transformation is very useful for reducing the number of variables in a database. It analyzes the cloud of points to find the most-fitting direction, which becomes the principal component and the first axis of a new basis. Then, it searches for the following most-fitting direction, with the caveat that it has to form an orthogonal basis. This is repeated for as many dimensions as the original dataset has. In the end, this technique obtains a square matrix T_P with as many dimensions as the original database, which transforms the basis of the dataset. In the newly transformed dataset, the variables no longer have physical meaning but are ordered by the percentage of the total variance explained by each principal component. These percentages can be obtained, helping to identify which components are important, as well as uncovering unhelpful directions that can be removed from the resulting dataset, such as eliminating the third component of the three-phase current, commonly addressed in this field by the Clarke transformation. This method effectively reduces the number of variables in the dataset.
3) *Filtering:* The goal of this step is to reduce the number of data points, usually by eliminating those that are not meaningful. For our specific application, given the discrete nature of the control action, deleting points that are very close to each other while yielding the same control output has proven to be useful.
4) *Probability-based analysis:* Finally, it is very important to validate that the database can perform well under scenarios that were not included in the collection or training step. A common approach is to perform a good number of control simulations as in the Monte Carlo method while registering how many times each part of the database is visited within a threshold. This can provide insight in two ways: First, these simulations help to identify which part of the database is used more, allowing to increase or adjust the aggressiveness of the filtering depending on the zone. Second, they can discover ill-conditioned zones that can be improved by adding new points. Validation is an iterative process that can be repeated many times until being satisfied with a final database.

C. Nearest Neighbor Searching

Usually known as k-NN, this regression algorithm in its basic form outputs the average of the values associated with the k nearest points in the dataset given an input query point.

The k database points closer to the query point can be found by calculating the distance between two points as the norm ϕ, defined as

$$||w_j - q||_\phi = \left(\sum_{i=1}^{n} |w_{j,i} - q_i|^\phi \right)^{1/\phi} \tag{12}$$

where q is the input query point of dimension n, w_j is the jth point in the database, and ϕ defines the type of norm. If $\phi = \infty$, the infinity norm is defined as

$$||w_j - q||_\infty = \max_i |w_{j,i} - q_i| . \tag{13}$$

Nearest neighbor searching is k-NN when $k = 1$, where only the closest point y_q of the database is used to select the output

$$y_q = \{y_j \in \mathcal{Y} : j = \arg\min_j ||w_j - q||_\phi, w_j \in \mathcal{W}\} . \tag{14}$$

This choice makes the algorithm faster while also being very effective when dealing with finite-set outputs. Notice that once ϕ has been chosen, the 1-NN estimator does not require any additional parameter.

D. 2-Step Nearest Neighbor Searching

The main disadvantage of neighbor searching is that, in its simplest form, the computational complexity grows linearly with the size of the training dataset. For our application, this is likely impossible to carry out in real time on current hardware due to the large training set required given the complexity of the system and the high frequency required for control. To overcome this problem, in this article, we propose an adaptation of nearest neighbor searching consisting of two quick focused searches that are significantly faster than one through the whole database. This method takes advantage of the periodic oscillatory nature of the three-phase alternating current produced by an inverter system to select a smaller subset of the training data to find the nearest neighbor.

The search is conducted by the variable $i_r \in \mathbb{R}^3$. When a three-phase system is balanced, the third phase is dependent on the other two. The Clarke transformation is a common resort in this field for simplifying and removing the data of the third current, and results in two sinusoidal waves that are 90° out of phase, which take the shape of a circumference in the transformed 2-D space, which is referred to as $\mathcal{P}$ in the following. PCA also achieves the same goal when applied to a dataset that contains a balanced three-phase grid because the new basis is orthogonal by definition, and it is the proposed method in this work due to the multivariable characteristic of the database.

This circumference in the $\mathcal{P}$ plane will be exploited to separate the search in two steps. The original database will be fragmented and classified by the angle of its point in this plane. In this way, the first 1-NN search will consist of finding the fragmented sector that the solution is in, and then, the second 1-NN search will find the exact solution. To obtain this classification space $\mathcal{P}$, we take the input space with the PCA transformation already applied $\mathcal{V} := \{v_j \in \mathbb{R}^{n_v} \mid v_j = T_P \cdot w_j, w_j \in \mathcal{W}\}$, with $n_v \leq n_w$. Then, the two main components related to i_r are projected, obtaining the new space $\mathcal{P} \in \mathbb{R}^2$. An example of this space is represented in Fig. 4.

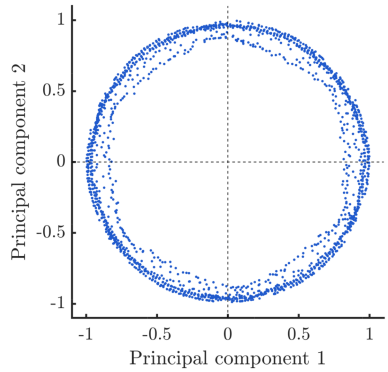

Fig. 4. Three-phase current projection.

A direct method to classify this space is to change these two components to a polar basis and then use the angle to divide the space $\mathcal{P}$ into n_s radial sectors, forming the partition $\mathcal{P} = \cup(\mathcal{P}_l)$, with $l = 1, \ldots, n_s$. Every section $\mathcal{P}_l$ can be later represented back into the original space $\mathcal{V}$ as hypersectors $\mathcal{H}_l \in \mathbb{R}^{n_v}$. Hypersectors $\mathcal{H}_l$ are, thus, portions of space $\mathcal{V}$ that are smaller the higher n_s is.

However, if the final purpose is the online implementation, it is recommended not to use the angle itself as the classification parameter, since the real-time calculus of an arctangent can be more demanding due to the division that is required to perform. Instead, we suggest using the two components of plane $\mathcal{P}$ to perform a 1-NN search, since vector norm calculations are simpler. Thus, for the first step of nearest neighbor searching, what needs to be stored in memory is the coordinates $\sigma \in \mathbb{R}^2$ of the middle point of each sector $\mathcal{P}_l$. This middle point can actually be any point on the bisecting line of $\mathcal{P}_l$, although points farther from the center of the circumference are recommended to improve searching accuracy. With this, any given query point q mapped in space $\mathcal{V}$, with its two main components related to i_r, can be used to perform 1-NN with the dataset of the middle points σ of sectors $\mathcal{P}_l$, whose solution can be transformed back into a hypersector $\mathcal{H}_l$ that contains the solution for query input q.

Then, a second 1-NN search can be performed inside the $\mathcal{H}_l$ obtained in the previous step, locating the final solution—the nearest neighbor of q. This is where the method vastly reduces the computing time of the whole process because it searches for the solution inside $\mathcal{H}_l$ instead of the entire database. However, the full database must be stored in memory anyway, just like the searching in one step. The only disadvantage of the two-step searching is that of storing an additional dataset of points σ, but this second dataset is significantly smaller.

However, a problem arises if the query point q is too close to any of the lateral borders of the sector $\mathcal{P}_l$, because there is a chance that the closest neighbor can belong to the adjoining sector, and, in this case, the inferred output could not be ideal [27]. In order to improve the accuracy of the searching method, the data belonging to neighboring hypersectors will also be added to the search, so a total of three hypersectors will be used in the second 1-NN search. Thus, the dataset $\mathcal{D}_k$ to be employed by

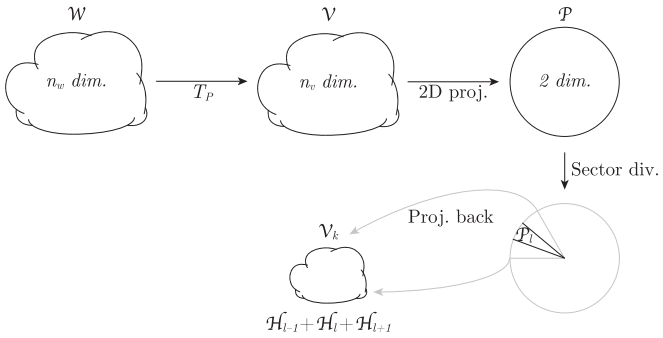

Fig. 5. Transformations of the adapted NNS methodology with dimensions.

the algorithm is

$$\mathcal{D}_k = \{(y_j, v_j) \mid v_j \in \mathcal{V}_k\} \tag{15}$$

with

$$\mathcal{V}_k = \bigcup_{l=k-1}^{l=k+1} \mathcal{H}_l \tag{16}$$

where, with some abuse of notation, $\mathcal{H}_0$ must be understood as $\mathcal{H}_{n_s}$, and $\mathcal{H}_{n_s+1}$ as $\mathcal{H}_1$.

A summary of the methodology is illustrated in Fig. 5. Each new input query point q is transformed by T_P and then projected into space $\mathcal{P}$. The first step of nearest neighbor searching will determine which sector $\mathcal{P}_l$ the input belongs to. With that, hypersector $\mathcal{H}_l$ is identified and the partition $\mathcal{D}_k$ of the database is obtained. The second step of nearest neighbor searching will infer the output y_j by calculating the nearest neighbor only within $\mathcal{V}_k$.

IV. FPGA Implementation of Nearest Neighbor Searching

In addition to all previous work and simplifications to facilitate the implementation of the control algorithm, additional procedures are required to meet the real-time control requirements, considering that the time interval target is usually in the range of 10–100 μs and that this type of kernel methods have to process the training dataset at every step. This is why we have to combine the learning algorithm with a hardware solution inside a reconfigurable platform such as the FPGA. Similarly to what was done in [28] for the Lipschitz interpolation algorithm, in this work we propose the implementation of nearest neighbor searching with a parallel architecture, designed to concurrently process different batches of data. This accelerates the execution of the algorithm, reaching the lower bound of the time target and allowing real-time implementation.

FPGA platforms are electronic devices made up of configurable logic blocks (CLBs) that can be used to implement custom reconfigurable combinational and sequential circuits. For our purpose, we will distinguish three elements that will be essential to explain the parallelization of the nearest neighbor searching algorithm.

1) Blocks of random access memory (BRAM), where all data will be stored for high-speed readings.
2) Arithmetic-logic units (ALU), which are programmed subsystems in charge of computing the algorithm.
3) Finite-state machines (FSM), which synchronizes BRAMs and ALUs with the clock signal.

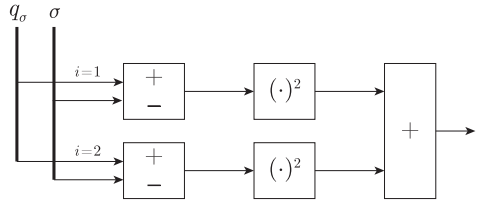

Fig. 6. ALU for Euclidean norm, with the final square-root calculation omitted. Notation q_σ refers to the two components of point q related to i_r, and σ are the two coordinates of a center of a sector $\mathcal{P}_l$.

The input to the FPGA is a point $q \in \mathcal{W}$ that contains measurements and references to the system and has to be compared to the points included in the database, which are saved in space $\mathcal{V}$ to reduce the required memory. For that, at the beginning of the implementation, the first circuit block contains ALUs programmed to transform any point q from space $\mathcal{W}$ into $\mathcal{V}$. This ALU performs only simple operations—fixed-point sums for normalization and multiplications for each dimension of $T_P \in \mathbb{R}^{n_w \times n_v}$. Then, as explained in the previous section, the 2-step nearest neighbor searching is two different 1-NN searches. The 1-NN method is also based on two procedures: 1) the calculation of norms of the difference between the query point and points of the dataset, and 2) the determination of the minimum norm that corresponds to the nearest neighbor.

In the first step of the methodology, there will be a BRAM containing the coordinates σ of the center of sectors $\mathcal{P}_l$. The norm recommended to find the closest sector is the Euclidean distance with $\phi = 2$. The circuit block for calculating this norm is shown in Fig. 6, where its ALUs carry out two subtractions, two multiplications, and one sum. Notice that the square root of the Euclidean norm has been omitted—It is not needed for 1-NN, as will be seen later. This norm is calculated in every sampling time for as many sectors as the database has been divided in, making this process ideal for parallelization, since many points can be read from BRAM to then calculate several norms in multiple ALUs at the same time. The maximum number of data that can be processed in parallel depends on the maximum number of ALUs programmed that can be fitted in this implementation without violating the constraint imposed by the number of resources available in the FPGA. It is also important to leave resources available for the second step of the searching methodology.

Once all the norms have been calculated, the second procedure of 1-NN is performed in the circuit shown in Fig. 7. Here, the closest sector center for the input q is obtained. This block combines a tree of comparators with multiplexers, in which the objective is not the minimum value itself, but the argument of the minimum, that is, the database address of the hypersector $\mathcal{H}_l$. This is why the square root of the Euclidean norm was omitted in Fig. 6, as it is a monotonic function that does not alter the order relation of the results.

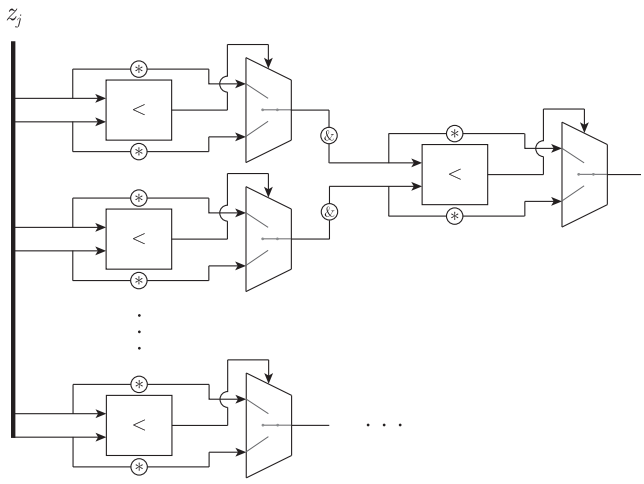

Fig. 7. ALU for argument of the minimum. The operator $*$ obtains the memory address of a value while the operator & retrieves the value from a memory address.

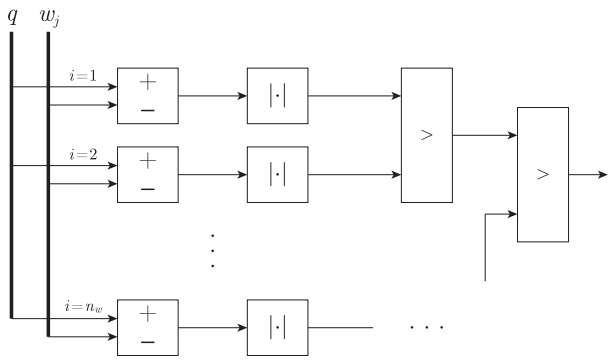

Fig. 8. ALU for infinity norm. Point w_j is located in address j and has n_w components, which are compared against each component of q after the PCA transformation in n_w subtraction blocks.

Similarly, in the second step of the methodology, BRAM blocks contain the input dataset $\mathcal{V}_k$, and several ALUs are in charge of calculating the infinity norm, which is preferred for multidimensional searching. Operations of subtraction and absolute value are required for each dimension of q and w_j. Then, a tree of comparators selects the maximum component, as shown in Fig. 8. This norm is calculated in every sampling time for as many points included in the subset $\mathcal{V}_k$, which can also be paralleled. The maximum number of data that can be processed in parallel can also be tuned by selecting the size of hypersectors $\mathcal{H}_l$. Dividing the database into more hypersectors results in the hypersectors being smaller because they have fewer points. With all infinity norms calculated, the same circuit block shown in Fig. 7 obtains the address of the actual nearest neighbor of query point q. With that address, the final access to memory will retrieve the control solution y_j from dataset $\mathcal{Y}$, which is stored in more BRAM blocks.

The scheme of the entire circuit implemented on the FPGA platform can be put together and summarized in the diagram in Fig. 9, inside the FPGA box. Here, the two steps of the searching methodology are visualized. After the input q is transformed from space $\mathcal{W}$ to space $\mathcal{V}$ in the processing block, the first step (NNS-1) identifies the section $\mathcal{D}_k$ in which the nearest neighbor of q is located, and then, the second step (NNS-2) locates the actual nearest neighbor of q. As demonstrated in [28], the parallelization of these types of algorithms implemented on FPGA hardware is several orders of magnitude faster than the sequential algorithm implemented in software.

TABLE III
WORKFLOW SUMMARY WITH THREE TESTBENCHES

Workspace 1: Regular PC (x86 Intel processor)
Matlab: FCS-MMPC program
Simulink: Testbench 1 with nonlinear simulator
Workspace 2: Regular PC (x86 Intel processor)
Matlab: Data gathering and processing
Matlab: Learned FCS-MMPC program
Simulink: Testbench 2 with nonlinear simulator
Matlab: Analysis and validation
Workspace 3: Speedgoat (CPU+FPGA combo)
FPGA: Learned FCS-MMPC program
CPU: Final testbench with nonlinear simulator

TABLE IV
SYSTEM PARAMETERS

Variable	Description	Value
R	Grid resistive load	30 Ω
L	Filter inductance	2 mH
C	DC-link capacitor	20 mF
V_{dc}	DC-link voltage	700 V
T_s	Sampling period	20 μs
N_α	Number of subintervals	2
$\lambda_1, \lambda_2, \lambda_3$	Cost function weights	$100, 1, 2$

V. CASE STUDY AND EXPERIMENT RESULTS

In previous sections, we presented general guidelines for modeling and controlling power inverters based on predictive techniques at high frequencies and reduced computational cost using machine learning. In this section, we will briefly go through these steps again with the specifics of our case study, assisted with a workflow summary in Table III, and finally, the experiment results that will help to justify the techniques and methods used.

A. Case Study

First, in Table IV, some system and control parameters are provided. The chosen multirate parameter N_α allows the control action u to commute at $N_\alpha\, T_s = 100$ kHz while the sampling frequency T_s is kept at 50 kHz. The cost function has been designed with the following terms:

$$f_1 = |i(j+1) - i_r(j+1)| \tag{17}$$

is the tracking cost, defined as the absolute value of the error between the values of the next predicted current and the reference for each phase

$$f_3 = |u(j) - u(j-1)| \tag{18}$$

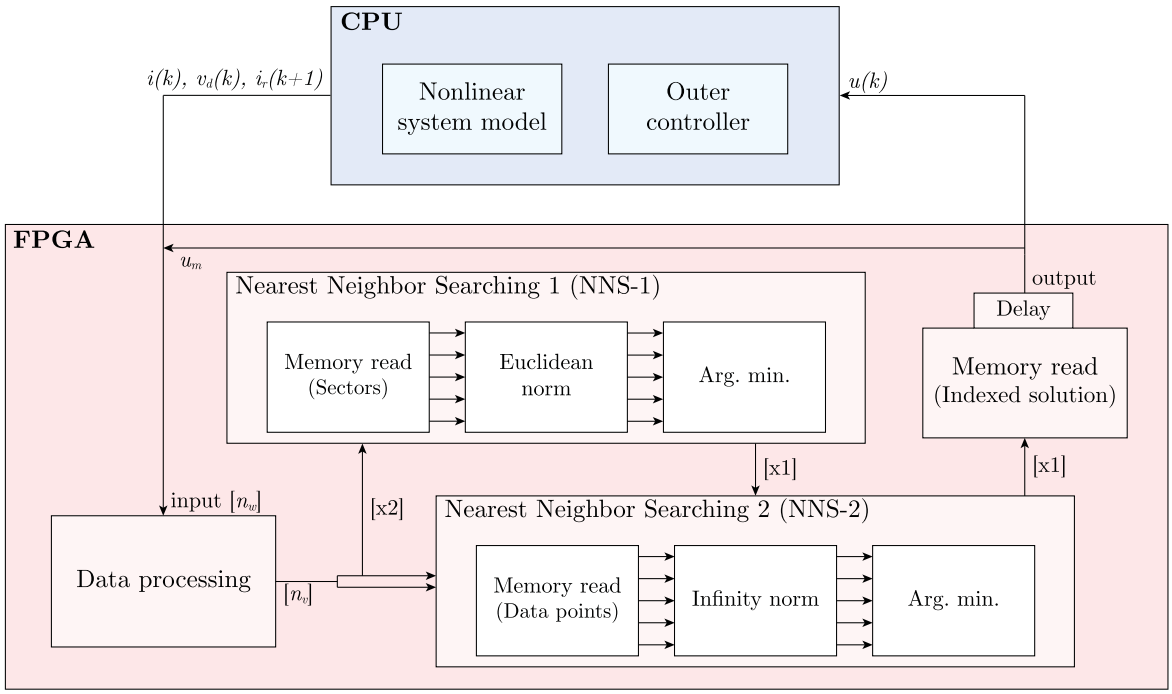

Fig. 9. Embedded implementation. Dimensions of data transfers between blocks are displayed in brackets.

penalizes jumps in the control action that can be associated with electric losses in the switching mechanisms; and finally

$$f_2 = v_d(j)^\intercal \left(v_d(j+1) - v_d(j)\right) \tag{19}$$

which is in charge of the capacitor balancing problem. As the objective is to drive each component of vector v_d to 0, the deviation from it is penalized by the dot product of the current value and the approximate future derivative. In this way, a positive component in v_d encourages a decreasing evolution in the same component for the next instant, and vice versa. Higher absolute values will receive more priority.

The first simulations of FCS-MMPC were performed in MATLAB-Simulink, where the control algorithm was programmed in MATLAB code, and the five-level inverter was a nonlinear system simulated on Simulink using the Simscape toolbox. This is the first testbench of the workflow. Several tests were carried out under the same system parameters, conditions, and control design previously shown, comparing FCS-MPC and FCS-MMPC with $N_\alpha = 2$. Fig. 10 shows an example of the behavior of the current of the first phase i_a in a steady state, including a zoomed view below. The multirate technique considerably improves current ripples even with the minimum N_α, resulting in a reduction in THD from 5.50% to 2.85% thanks to the multirate upgrade. More comparative information between FCS-MPC and its multirate enhancement can be found in [29], where simulations have been carried out for different numbers of N_α and N. Higher N_α significantly improves THD performance while $N = 3$ is usually a good ceiling. Computation time increases exponentially with higher numbers for both parameters. These increments correspond to the use of traditional solvers.

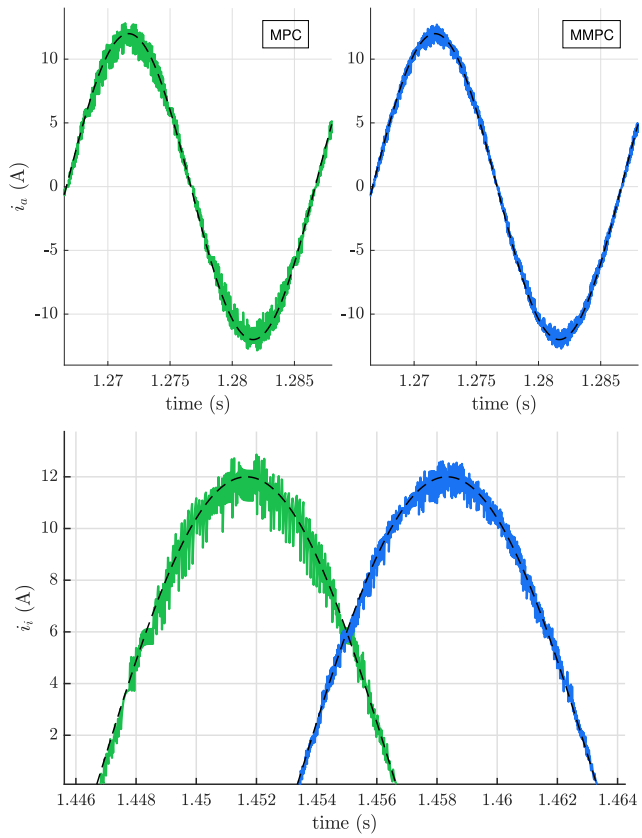

Fig. 10. Current i_a controlled by FCS-MPC, versus current i_a controlled by FCS-MMPC at $N_\alpha = 2$. Zoomed comparison in the bottom graphic.

However, when using the learning method, the computation time is hardly affected by the values of N_α and N.

Once the first testbench is established, the second step is preparing the learning technique prior to establishing the second testbench. Data are collected by performing simulations in this first testbench and following the guidelines of Section III-B. In our case, the initial number of output columns in the database is 6, since the control action u needs 3×2 dimensions for the three phases and the double frequency, and the input columns were 15, one per variable and components. In the data processing stage in Section III-B, after applying PCA in the input set, we obtained the following results for the percentage of variance explained by each principal component: [39.1 38.9 12.8 3.9 3.4 0.6 0.5 0.4 0.15 0.11 0.10 0.04 0.04 0 0]. The first two components are the ones most related to i_r, which account for almost 80% of the variance. This is expected because current tracking is the priority objective of this control problem, which is also the reason why the chosen value of the weighting factor λ_1 is the highest. The last two components are null and, thus, can be safely removed from the database, but after exhaustive testing in our case study, we had concluded that up to five more components can be removed for a total of seven, as the first eight components cover a combined 99.56% of data explanation. The final database that we considered optimal has 6 fixed outputs plus 8 inputs, down from the original 15 inputs. This also defines the dimensions for $T_P \in \mathbb{R}^{15\times 8}$. Regarding the length of the database, after the filtering process and exhaustive validating simulations to remove redundant points, our application required almost 32 000 data points to keep the learning error below 0.02%.

With the final database, a second testbench was established in MATLAB-Simulink in which FCS-MMPC was interchanged with the learned method. More simulations were performed to confirm the performance of the learned method versus the raw MPC, where THD in steady state would not increase over two hundredths at any scenario within the ranges trained during the data collection. However, when comparing the calculation time of both methods, FCS-MMPC takes several seconds to calculate the optimal solution, exponentially increasing with N_α and N, whereas the learned method does it within a millisecond regardless of N_α and N. The second testbench was still far from the target of microseconds, requiring an FPGA with the proposed parallel implementation.

Before the implementation of the embedded control, we would like to address the issue of the control computation delay. In control theory and during simulations, it is usually assumed that the control action is calculated and applied instantaneously, which is a good assumption for systems with slow time constants where the sampling time and control action can be considered instantaneous. This is, however, not true for MPC in power electronics in general. Whatever the state of the system is at instant k, a suitable control action for that state is not applied until $k\,T_s + t_c$ due to the computation time, and t_c is usually high enough to introduce considerable error with a negative impact on control performance that is difficult to address in power electronics [30]. However, in MPC schemes, this is easily alleviated: Let us call the state of the system $x(k)$ as the combination of system variables, e.g., i and v_d in this article, at the time instant k. Since the optimization problem cannot be solved immediately, a control action $u(k|k-1)$ must be calculated beforehand with the information available at $k-1$, but then applied at k, lasting from k to $k+1$. During this interval, a prediction step can be performed using the system model equation (2) to update $x(k)$ to $x(k+1|k)$, which is the predicted state at $k+1$ with the information available at k. This new predicted state can be used in the optimization problem of the MPC to obtain $u(k+1|k)$. This is calculated in the interval from k to $k+1$ and ready to be applied at $k+1$. For this method to work, the computing time t_c must be less than the sampling time.

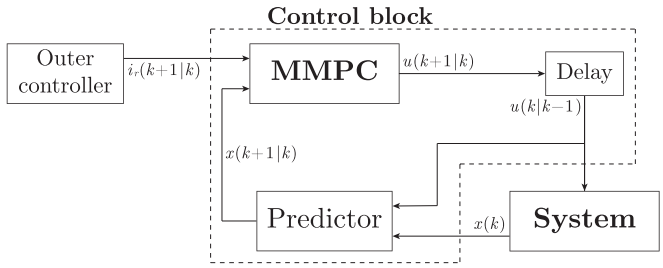

Fig. 11. Embedded control scheme with delay compensation.

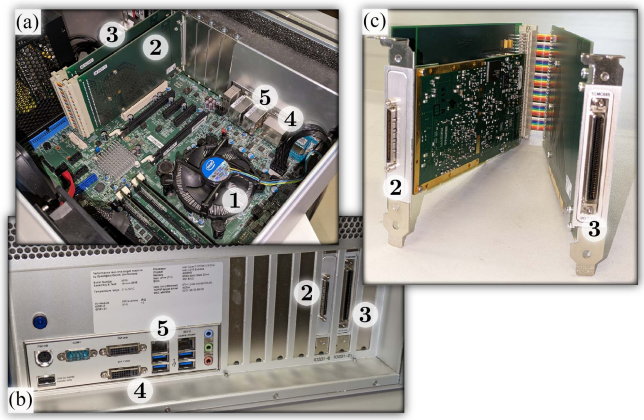

Fig. 12. Speedgoat real-time system. Photographs: (a) Upper view on the inside of the system; (b) front view for the external interface; (c) Modules IO331-X extracted from the motherboard. Components: (1) CPU under heatsink and fan, Intel Core i7 3770K chip; (2) module IO331-6, containing 16 analog-to-digital converters and the FPGA Xilinx Spartan 6 chip; (3) module IO331-21, as I/O expansion for the FPGA; (4) VGA port for real-time monitoring; (5) ethernet port for programming and data transfer.

Fig. 11 illustrates this control scheme with a block prior to the MMPC optimization problem. It is important to note that this additional prediction step does not increase computation time when using the nearest neighbor searching methodology to implement the control law. This is because the scheme does not introduce any new variables or affect the size of the database and only improves the solution of the control action by returning $u(k+1|k)$ instead of $u(k|k)$. The final inputs to the database are still the reference $i_r(k+1|k)$ from an outer controller and the system state $x(k)$.

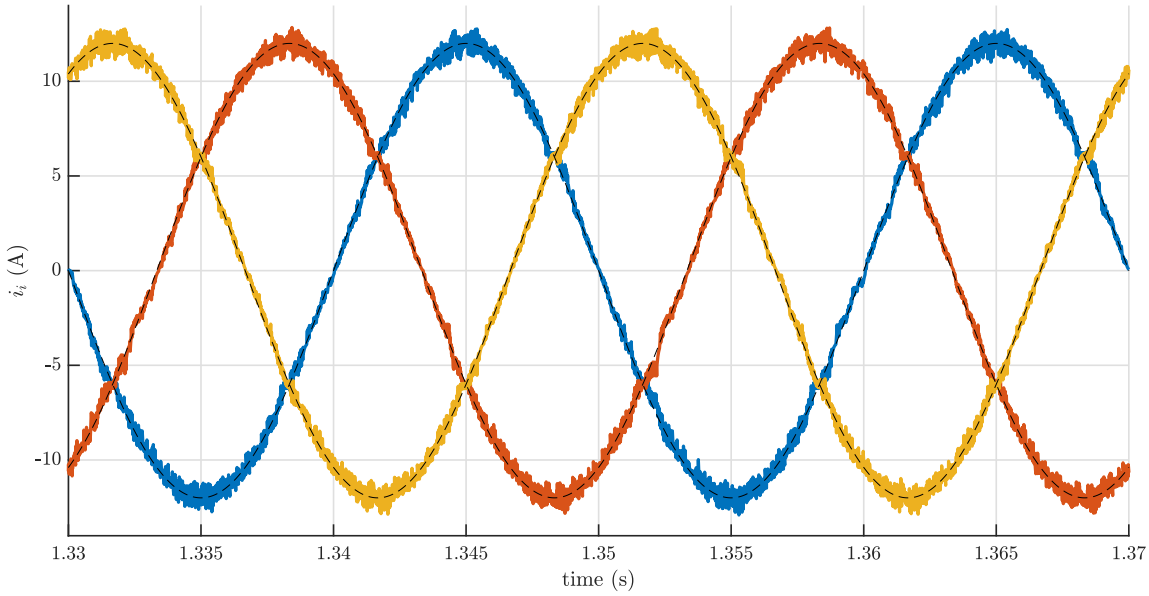

Fig. 13. Grid currents tracking a reference signal of 12 A at 50 Hz.

B. FPGA-in-the-Loop Experiment Results

The following experiments were carried out on a Speedgoat IO331-06-21 real-time system shown in Fig. 12. The Speedgoat mainly contains an FPGA located in the module IO331-6 that is connected through PCIe lines to an Intel-based CPU. It also has a second module IO331-21 to expand the I/O capabilities of the FPGA. The five-level inverter system was reproduced using modeling packages from Simscape Electrical toolbox for Simulink and computed on the CPU of the Speedgoat. The system and control parameters are the same as in the simulation in Section III, which can be found in Table IV. The embedded implementation portrayed in Fig. 9 was programmed in the FPGA to read and control the simulator in the CPU in real time.

The delay involved in the analog-to-digital converters (ADC) stage of this setup is also included for this hardware-in-the-loop implementation by triggering them and waiting for the ready signal of the measurement to arrive at the FPGA, but then discarding the resulting value. In a full experiment with a real inverter, ADCs will be necessary to translate the signals from measurement sensors. These converters obviously need some time to obtain the digital measure, and that will introduce delays to consider in the experiment. The ADCs found in this Speedgoat have a resolution of 16 b, and in order to be closer to the real experiment, we also considered that the 2 least significant bits of every measurement going through the ADC can be noisy.

For our database, the resolution of the inputs was matched to 16 bits in fixed-point representation to match the resolution of the ADC. For the outputs, which are the control action for each phase, only 3 b of resolution are required as there are only 5 levels. This results in a database with 429 kB of size, which is 92% of the high-speed memory available in the FPGA. With the implementation detailed in Section IV, the FPGA was measured to take 936 clock cycles to obtain the output each time, which is 9.36 μs. This, added to 8 ± 0.1 μs of ADC delay,

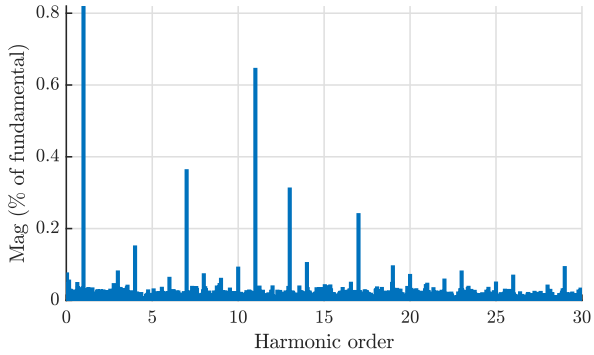

Fig. 14. Harmonic spectrum of the grid current.

results in a maximum of 16.46 μs, leaving some room before the limit of 20 μs corresponding to the sampling period for possible delays in the signals that arrive from sensor measurements. This result is very important since the control scheme requires the computation time to be lower than the sampling time.

Regarding the control results of the experiment, starting from a random initial state, Fig. 13 depicts the grid currents achieved at steady state with the proposed approach of MMPC learned with the nearest neighbor searching adaptation. Harmonic spectrum is shown in Fig. 14 with a THD value of 3.05%, just 0.21% higher than the full-resolution simulation carried out in Section III-D. The small increment in THD is expected due to the limited resolution and the artificial noise introduced in the least 2 significant bits, which is common in a real experiment. This result matches the performance of the interpolated kernel method of the previously cited [13] while requiring less resources and being computationally faster due to skipping the interpolation phase.

Fig. 15 shows the control result of the capacitor balancing variables. The experiment starts from an unknown unbalanced state to evaluate the performance of the learned control. Steady

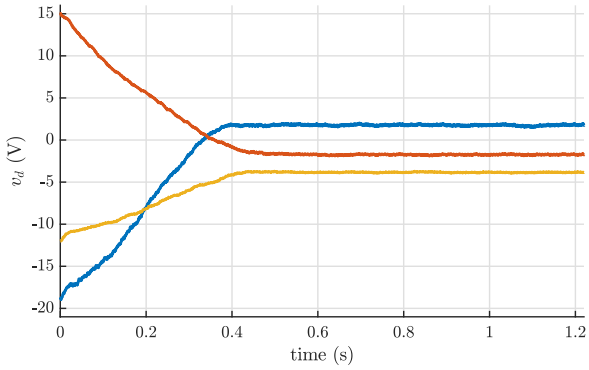

Fig. 15. Capacitor voltage differences starting from an unbalanced condition.

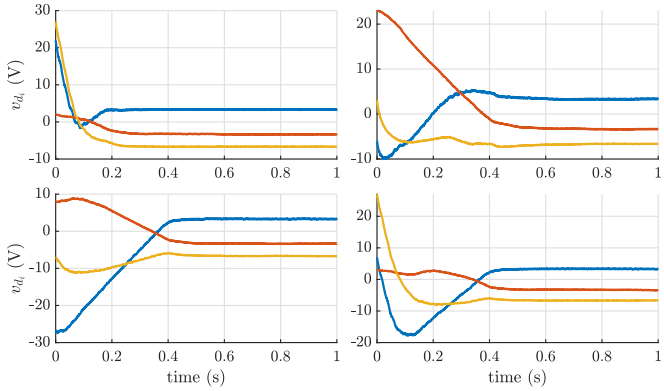

Fig. 16. Four experiments for transient analysis starting from different unbalanced conditions.

state is reached at around 0.4 s, about twice as long as it takes in an ideal simulation with MMPC without the learning layer. This result is very much expected, as the initial conditions for this experiment were chosen to be different from those selected to train the learning algorithm. However, it reaches the same deviation from zero as the MMPC without the learning layer. Consider that these deviations can be neglected compared to capacitor voltage values. And while it is possible to tune the optimization problem of the MPC to drive v_d closer to zero, this can negatively affect the other control objectives. Nevertheless, good performance in a steady state compared to the raw MMPC simulation is what validates the feasibility of the proposed learning approach.

Fig. 16 includes four more experiments starting from different unbalanced states, where the behavior of v_d is shown. Minor differences are expected to appear in the settling time, but the capacitor voltage balance objective has always been reached as long as v_c is within the range trained in the data-gathering phase. Fig. 17 comes from another experiment to test robustness under input voltage changes, which is a disturbance easy to reject for MPC that the kernel technique has succeeded to capture. At the moments of V_{dc} changes, the maximum recorded increment in THD was 0.11 during the first cycle, but quickly recovered within a tenth of a second.

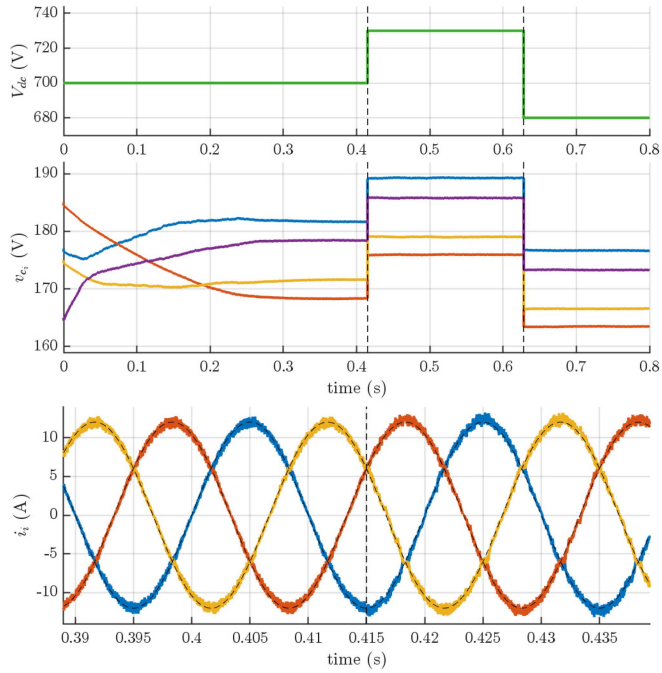

Fig. 17. System response to step changes in the input voltage V_{dc}.

VI. Conclusion

We have presented a machine learning technique based on k-NN adapted to power inverters that allows the embedded implementation of more advanced and complex control methods. This technique is parallelizable and its computation time is very low when implemented in dedicated hardware such as FPGAs. Multirate FCS-MPC was showcased as the learned control method and results are promising—nearest neighbor searching succeeds in learning the control law with minor performance drops when compared to the raw control method, and harmonic distortion is significantly reduced compared to other finite-control-set implementations without the multirate approach. Experiments were conducted in an FPGA-in-the-loop setup while proving its scalability to full experiments with an FPGA-based real-time control system.

References

[1] P. Cortés, M. P. Kazmierkowski, R. M. Kennel, D. E. Quevedo, and J. Rodríguez, "Predictive control in power electronics and drives," *IEEE Trans. Ind. Electron.*, vol. 55, no. 12, pp. 4312–4324, Dec. 2008.

[2] L. G. Franquelo, J. Rodriguez, J. I. Leon, S. Kouro, R. Portillo, and M. A. M. Prats, "The age of multilevel converters arrives," *IEEE Ind. Electron. Mag.*, vol. 2, no. 2, pp. 28–39, Jun. 2008.

[3] C. Albea, O. L. Santos, D. Z. Prada, F. Gordillo, and G. Garcia, "Hybrid control scheme for a half-bridge inverter," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9336–9341, 2017.

[4] S. Kouro, P. Cortés, R. Vargas, U. Ammann, and J. Rodríguez, "Model predictive control—A simple and powerful method to control power converters," *IEEE Trans. Ind. Electron.*, vol. 56, no. 6, pp. 1826–1838, Jun. 2008.

[5] S. Vazquez et al., "Model predictive control: A review of its applications in power electronics," *IEEE Ind. Electron. Mag.*, vol. 8, no. 1, pp. 16–31, Mar. 2014.

[6] P. Karamanakos, E. Liegmann, T. Geyer, and R. Kennel, "Model predictive control of power electronic systems: Methods, results, and challenges," *IEEE Open J. Ind. Appl.*, vol. 1, pp. 95–114, Aug. 2020.
[7] P. Montero-Robina, F. Umbría, F. Salas, and F. Gordillo, "Integrated control of five-level diode-clamped rectifiers," *IEEE Trans. Ind. Electron.*, vol. 66, no. 9, pp. 6628–6636, Sep. 2019.
[8] T. Geyer, *Model Predictive Control of High Power Converters and Industrial Drives*. Hoboken, NJ, USA: Wiley, 2016.
[9] P. Karamanakos, T. Geyer, N. Oikonomou, F. D. Kieferndorf, and S. Manias, "Direct model predictive control: A review of strategies that achieve long prediction intervals for power electronics," *IEEE Ind. Electron. Mag.*, vol. 8, no. 1, pp. 32–43, Mar. 2014.
[10] B. Stellato, T. Geyer, and P. J. Goulart, "High-speed finite control set model predictive control for power electronics," *IEEE Trans. Power Electron.*, vol. 32, no. 5, pp. 4007–4020, May 2017.
[11] M. Jeong, S. Fuchs, and J. Biela, "When FPGAs meet regionless explicit MPC: An implementation of long-horizon linear MPC for power electronic systems," in *Proc. 46th Annu. Conf. IEEE Ind. Electron. Soc.*, 2020, pp. 3085–3092.
[12] J. G. Ordonez, F. Gordillo, P. Montero-Robina, and D. Limon, "Suboptimal multirate MPC for five-level inverters," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 13424–13429, 2020.
[13] J. G. Ordonez, J. Nadales, D. Limon, and F. Gordillo, "Data-driven multirate predictive control of power inverters based on kinky inference," in *Proc. 60th IEEE Conf. Decis. Control*, 2021, pp. 4358–4363.
[14] S. Zhao, F. Blaabjerg, and H. Wang, "An overview of artificial intelligence applications for power electronics," *IEEE Trans. Power Electron.*, vol. 36, no. 4, pp. 4633–4658, Apr. 2021.
[15] L. Huang, J. Coulson, J. Lygeros, and F. Dörfler, "Data-enabled predictive control for grid-connected power converters," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 8130–8135.
[16] A. Brosch, S. Hanke, O. Wallscheid, and J. Böcker, "Data-driven recursive least squares estimation for model predictive current control of permanent magnet synchronous motors," *IEEE Trans. Power Electron.*, vol. 36, no. 2, pp. 2179–2190, Feb. 2021.
[17] I. S. Mohamed, S. Rovetta, T. D. Do, T. Dragicević, and A. A. Z. Diab, "A neural-network-based model predictive control of three-phase inverter with an output *LC* filter," *IEEE Access*, vol. 7, pp. 124737–124749, 2019.
[18] C. N. Bhende, G. N. V. Mohan, and Y. Raghuwanshi, "Machine learning based cooperative control of photovoltaic systems for voltage regulation," in *Proc. Int. Conf. Power Electron. Energy*, 2023, pp. 1–6.
[19] S. Lucia, D. Navarro, B. Karg, H. Sarnago, and Ó. Lucía, "Deep learning-based model predictive control for resonant power converters," *IEEE Trans. Ind. Informat.*, vol. 17, no. 1, pp. 409–420, Jan. 2021.
[20] D. Wang et al., "Model predictive control using artificial neural network for power converters," *IEEE Trans. Ind. Electron.*, vol. 69, no. 4, pp. 3689–3699, Apr. 2022.
[21] F. Simonetti, G. D. Di Girolamo, A. D'Innocenzo, and C. Cecati, "Machine learning for model predictive control of cascaded h-bridge inverters," in *Proc. IEEE 21st Mediterranean Electrotech. Conf.*, 2022, pp. 1241–1246.
[22] P. Montero-Robina, F. Gordillo, F. Gómez-Estern, and F. Salas, "Voltage balance for five-level DCC based on mixed-integer linear programming," *Int. J. Elect. Power Energy Syst.*, vol. 124, 2021, Art. no. 106302.
[23] E. F. Camacho and C. Bordons, *Model Predictive Control*. Berlin, Germany: Springer, 2013.
[24] A. Blaas, J. M. Manzano, D. Limon, and J. Calliess, "Localised kinky inference," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 985–992.
[25] J. Nadales, J. M. Manzano, A. Barriga, and D. Limón, "Implementation of fast predictive controllers on FPGA platforms based on parallel Lipschitz interpolation," in *Proc. Eur. Control Conf.*, 2021, pp. 1537–1542.
[26] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscipl. Rev.: Comput. Statist.*, vol. 2, no. 4, pp. 433–459, 2010.
[27] J. M. Manzano, D. Limon, D. Munoz de la Pena, and J. P. Calliess, "Output feedback MPC based on smoothed projected kinky inference," *IET Control Theory Appl.*, vol. 13, no. 6, pp. 795–805, 2019.
[28] J. Nadales, J. Manzano, A. Barriga, and D. Limon, "Efficient FPGA parallelization of Lipschitz interpolation for real-time decision-making," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 5, pp. 2163–2175, Sep. 2022.
[29] J. G. Ordonez, D. Limon, and F. Gordillo, "Multirate predictive control for diode clamped inverters with data-based learning implementation," in *Proc. 22nd IFAC World Congr.*, 2023.
[30] M. Lu, X. Wang, P. C. Loh, F. Blaabjerg, and T. Dragicevic, "Graphical evaluation of time-delay compensation techniques for digitally controlled converters," *IEEE Trans. Power Electron.*, vol. 33, no. 3, pp. 2601–2614, Mar. 2018.

Joaquin G. Ordonez received the B.Sc. and M.Sc. degrees in electrical and computer science from the University of Seville, Seville, Spain, in 2016 and 2018, respectively.

In 2018, he was a Research Intern with Mitsubishi Electric Research Laboratories, Boston, MA, USA, studying HVac efficiency and predictive control in buildings. He is currently a Ph.D. candidate with the University of Seville, where he is focusing on data-driven and predictive control techniques, and their applications and real-time implementation in power inverters. He was a Visiting Researcher with Seoul National University, South Korea, in 2022, testing and prototyping inverter topologies in HiL simulators.

Pablo Montero-Robina (Member, IEEE) received the Ph.D. degree in industrial engineering from the University of Seville, Seville, Spain, in 2021.

He also collaborated in teaching activities in the public and private sectors during this period. He worked on Hitachi Energy Research from 2021 to 2023 in Västerås, Sweden, where he collaborated and directed research projects on HVdc facilities and applications. Afterward, he was with the R&D Center of Autel Iberia S.L. as a Power Electronic Engineer, where he is focused on developing power modules for the EV charging industry. He has field experience on setting up grid-connected converters, testing and developing control and modulation algorithms, and selecting hardware components for power electronic equipment. His research interests include control, design, and development of power converters, topologies, and thermal design for automative, industrial, and energy distribution and transportation.

Daniel Limon received the M.Eng. and Ph.D. degrees in electrical engineering from the University of Seville, Seville, Spain, in 1996 and 2002, respectively.

From 1999 to 2007, he was an Assistant Professor with the Department of Automation and Systems Engineering, University of Seville, where he was an Associate Professor from 2007 to 2017 and has been a Full Professor since 2017. He was a Visiting Researcher with the University of Cambridge, Cambridge, U.K., in 2016, and with Mitsubishi Electric Research Laboratories in 2018. His current research interests include model predictive control, stability and robustness analysis, tracking control, and data-based control.

Dr. Limon was a Keynote Speaker at the International Workshop on Assessment and Future Directions of Nonlinear Model Predictive Control in 2008 and a Semiplenary Lecturer at the IFAC Conference on Nonlinear Model Predictive Control in 2012. He was the Chair of the fifth IFAC Conference on Nonlinear Model Predictive Control in 2015.

Francisco Gordillo received the M.Eng. and Ph.D. degrees in industrial engineering from the University of Seville, Seville, Spain, in 1988 and 1994, respectively.

Since 1989, he has been with the Department of Automatic Control, Escuela Superior de Ingenieros, Universidad de Sevilla, where he is currently a Full Professor. He is the coauthor of Dinámica de Sistemas (Madrid: Alianza Editorial, 1997) and Stability Issues in Fuzzy Control (Berlin: Physica-Verlag, 2000), and author or coauthor of more than 150 publications including book chapters, journal articles, and conference proceedings. His research interest includes nonlinear control with applications to mechanical, electromechanical, and power electronic systems.