Speed Control of PMSM with Finite Control Set Model Predictive Control Using General-purpose Computing on GPU

KOZUBÍK, M.; VÁCLAVEK, P.

Proceedings of the IECON 2020 - The 46th Annual Conference of the IEEE Industrial Electronics Society,pp. 379-383

eISBN: 978-1-7281-5413-8

ISSN: 2577-1647

DOI: https://doi.org/10.1109/IECON43393.2020.9254381

Accepted manuscript

Speed Control of PMSM with Finite Control Set Model Predictive Control Using General-purpose Computing on GPU

Michal Kozubik
CEITEC - Central European Institute of Technology
Brno University of Technology
Brno, Czech Republic
Email: Michal.Kozubik@ceitec.vutbr.cz

Pavel Vaclavek
CEITEC - Central European Institute of Technology
Brno University of Technology
Brno, Czech Republic
Email: Pavel.Vaclavek@ceitec.vutbr.cz

***Abstract*—In this article, novel approach in implementing finite control set predictive control is introduced. Algorithm is implemented using general-purpose computing on graphics processing unit. Predictions are computed using parallel threads on the GPU. Optimal switching state is then selected in dependence on the cost function given by angular speed error and constraints on the current. The algorithm is tested in the PIL simulation using Simulink and Jetson Nano. The ability of the algorithm to ensure the reference tracking and keeping the current within its limits are discussed.**

***Index Terms*—finite control set model predictive control, permanent magnet synchronous motor, general-purpose computing, graphics processing unit**

I. Introduction

Model predictive control (MPC) of the permanent magnet synchronous motor (PMSM) is a very popular topic in the field of industrial control. Predictive control offers an easy way of implementation of nonlinearities and constraints. On the other hand, it poses a challenge because of the need for the short sampling time.

Predictive control can be separated into two groups [1]. The first of them is continuous set (CS) MPC. Within this approach, continuous optimization is used. That means the speed of the search for the optima and its accuracy is heavily dependent on the used solver. Research in this field is focusing on the acceleration of the optimization [2]. In the field of drive control CS-MPC is used to calculate the exact value of the voltage which is then modulated by the pulse-width modulation (PWM) [3], [4].

In the finite control set (FCS) model predictive control, the part of modulation is skipped. The control algorithm directly computes the optimal switching state of the inverter [5]. This can be applicable on the PMSM with 2-level voltage source inverter (VSI) [6]–[8], matrix inverters [9] or the multi-level inverters [10].

The approach of the finite control set is not dependent on the optimization solver. In its nature it is a very straightforward process and offers an easy way to include nonlinear constraints. On the other hand, the number of combinations grows exponentially with the length of the prediction horizon. That makes the problem unsolvable on the CPU in real-time, which is necessary for the control of PMSM. This leads to the implementation of algorithms of FCS-NMPC on the digital signal processors or the field-programmable gate arrays [11].

Graphics processing units offer a huge amount of possible parallel threads and are usually used to accelerate the creating of an image for an output device. Nowadays, GPUs are used to general-purpose calculations, i.e. neural networks [12]. If the problem can be parallelized, the GPU offers a bigger capability in computation than the CPU. In the model predictive control GPU can be used as an accelerator for the model evaluation [13].

In this article, native parallelism of the possible switching states combinations is used and implemented on the GPU. This enables the obtaining of optimal switching state combination in a short time without the need for a reduction of combination tree. The proposed algorithm is tested on the simulation with the 2-level 3-phase voltage source inverter.

The goal of this article is to evaluate the possibility of using GPU-based computation of the optimal switching state in the implementation of the finite control set model predictive control of the fast system as permanent magnet synchronous motor.

II. Analysis

A. Plant

Continuous-time model of permanent magnet synchronous motor in the *dq* reference frame is

$$\begin{aligned}
\frac{\mathrm{d}i_d}{\mathrm{d}t} &= -\frac{R}{L_d}i_d + P_p\frac{L_q}{L_d}i_q\omega_m + \frac{1}{L_d}u_d \\
\frac{\mathrm{d}i_q}{\mathrm{d}t} &= -\frac{R}{L_q}i_q - P_p\frac{L_d}{L_q}i_d\omega_m - \frac{Pp\Psi_{PM}}{L_q}\omega_m + \frac{1}{L_q}u_q \\
\frac{\mathrm{d}\omega_m}{\mathrm{d}t} &= \frac{3}{2}\frac{P_p}{J}\left(\Psi_{PM}i_q + (L_d - L_q)i_d i_q - T_l\right) \\
\frac{\mathrm{d}\vartheta_m}{\mathrm{d}t} &= \omega_m,
\end{aligned} \tag{1}$$

where

i_d, i_q are stator current components in *dq* frame,

u_d, u_q	are stator voltage components in *dq* frame,
ω_m	is rotor mechanical angular speed,
ϑ_m	is rotor mechanical angle,
T_l	is load torque,
P_p	is number of pole pairs,
R	is stator winding resistance,
L_d, L_q	are rotor inductance components,
Ψ_{PM}	is permanent magnet flux,
J	is the moment of inertia.

Stator voltage components u_d, u_q can be derived from phase voltages u_a, u_b, u_c using Clarke and Park transformation

$$\begin{aligned} K_P &= \begin{bmatrix} \cos(\vartheta_e) & \sin(\vartheta_e) & 0 \\ -\sin(\vartheta_e) & \cos(\vartheta_e) & 0 \end{bmatrix} \\ K_C &= \begin{bmatrix} \sqrt{\frac{2}{3}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \\ \begin{bmatrix} u_d \\ u_q \end{bmatrix} &= K_P K_C \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix}, \end{aligned} \tag{2}$$

where ϑ_e is rotor electrical angle.

For the proper computation of state trajectories by the predictive controller, a discrete-time model is needed. Assuming that the sampling period T_s is much smaller than the electrical time constant, a continuous-time model can be rewritten as discrete-time using the Euler method

$$\begin{aligned} i_d(k+1) &= (1 - T_s\frac{R_s}{L_d})i_d(k) + T_s P_p \frac{L_q}{L_d}\omega_m(k) i_q(k) + \\ &\quad + T_s \frac{1}{L_d} u_d(k) \\ i_q(k+1) &= (1 - T_s\frac{R_s}{L_q})i_q(k) - T_s P_p \frac{L_d}{L_q}\omega_m(k) i_d(k) - \\ &\quad - T_s P_p \Psi_{PM} \frac{1}{L_q}\omega_m(k) + T_s \frac{1}{L_q} u_q(k) \\ \omega_m(k+1) &= \omega_m(k) + T_s \frac{3}{2}\frac{P_p}{J}\Psi_{PM} i_q(k) + \\ &\quad + T_s \frac{3}{2}\frac{P_p}{J}\left(L_d - L_q\right) i_d(k) i_q(k) \\ \vartheta_m(k+1) &= \vartheta_m(k) + T_s \omega_m(k). \end{aligned} \tag{3}$$

Phase voltages $u_a(k)$, $u_b(k)$, $u_c(k)$ are generated by VSI. Every VSI has finite set of possible switching states $s(k) \in S$ based on its architecture. Thus, the set of phase voltages $u_p(k) = [u_a(k) \quad u_b(k) \quad u_c(k)]^T$ (input) is also finite. Set of future states $x(k+1) = [i_d(k+1) \quad i_q(k+1) \quad \omega_m(k+1)]^T$ can be computed from the previous values of states and input and therefore is finite.

B. Controller

One of the basic demands on the controller is to ensure tracking the reference signal. In the case of speed control, demand on tracking the reference signal $\omega_{m,r}$ can be written as $\lim_{t\to\infty} \|\omega_{m,r} - \omega_m\| < \epsilon$, where $\epsilon \in \mathbb{R}$ is small.

Another common and important requirement is to work within constraints of states and inputs. In the case of drive control, the maximal value of stator current is rated current I_R. This constraint can be written as

$$\sqrt{i_d^2 + i_q^2} \leq I_R. \tag{4}$$

Other demands can be put on controller, such as field weakening for reaching angular speeds higher than the rated one.

III. Control algorithm

In this section, three main parts of the control algorithm are described: prediction, cost function evaluation and switching state selection.

1) Prediction: Prediction is made across all possible combinations of the VSI. The number of combinations is based on the architecture of VSI and the length of the prediction horizon N. For the two-level three-phase VSI there are $(2^3)^N$ possible combinations. A number of the combination in binary represents whether the phase is switched on or off, e.g. *5 - 101 - C-on B-off A-on*. The next three bits represent a step on the prediction horizon.

Binary values are then used to represent switching in the model. Model (3) with respect to (2) is evaluated across the whole prediction horizon.

The choice of N is affected by two things. As was shown, a number of combinations grows exponentially with N. Thus, the maximal value is limited by computational speed. The lower limit of the N is given by the controlled system.

2) Cost function evaluation: To get the best possible combination it is necessary to measure its performance on the prediction horizon. Based on the demands put on the controller, the specific cost function is evaluated. In this case, the cost function has three parts.

The first part is dealing with the reference tracking. The controller has to minimize the speed error. Hence, the cost of the tracking error c_{TE} is

$$c_{TE}(k) = \sum_{i=1}^{N} w_{TE}(\omega_{m,r}(k+i) - \omega_m(k+i))^2, \tag{5}$$

where w_{TE} is a weighting coefficient of tracking error cost.

In standard field-oriented control i_d has zero reference. In predictive control this can be represented as cost of i_d:

$$c_{i_d}(k) = \sum_{i=1}^{N} w_{i_d}(\imath_d(k+i))^2, \tag{6}$$

where w_{i_d} is a weighting coefficient of i_d cost.

Introduced algorithm does not use any optimization approach in finding the best combination. Therefore, it is necessary to somehow represent constraints put on the control law. The simplest way to do it is by representing it as the part of the cost function. The simplest way would be branch with two values - zero, when the value is within its limits,

and ideally infinity, when the value exceeds them. During implementation, this approach was tested, but it led to reduced computing efficiency due to the stalls.

To achieve better computing efficiency constraint (4) was represented as the barrier function [14]

$$c_{IC}(k) = \sum_{i=1}^{N} -\log\left(i_d^2(k+i) + i_q^2(k+i) - I_R^2\right) w_{IC}, \quad (7)$$

where w_{IC} is a weighting coefficient of the cost of exceeding current constraints.

The cost function for every possible combination is then given by the sum of all parts

$$C_J(k) = c_{TE} + c_{i_d} + c_{IC}, \quad (8)$$

where J is the index of combination.

3) Switching state selection: Given values of the cost function, the combination with the lowest value is selected. Proper selection is heavily affected by the choice of weighting coefficients. As the binary value of index describes used combination, switching state $s(k)$ can be found as the argument of the minima of the finite set of cost function values

$$s(k) = \underset{J}{\operatorname{argmin}}\{C_J(k) \mid J \in S\}. \quad (9)$$

In the case of the same value of the cost function, the combination with lower index is preferred, therefore the states with all phases turned off is preferred to state with all phases turned on.

IV. Simulation Results

This section covers the testing of the proposed algorithm. Firstly, the implementation of the algorithm on the platform Jetson Nano is presented. Also, the limitations of the platform are described. Secondly, the results of the simulation with the Simscape model of PMSM are presented.

A. Implementation

The algorithm can be separated into two parts: CPU part and GPU part. CPU part covers data transfer and preparation, for example, initial Clarke and Park transformation. GPU part deals with finding the best switching state.

Every combination can be computed separately. Thus, the problem of searching for the optimal combination is parallel by its nature. GPU computing is based on performing a large amount of the same mathematical functions on different data parallelly in separate threads. Drawing on this information, prediction and cost function evaluation for every combination are computed separately. Finding minima is also performed on the GPU but across all threads.

The number of called threads is limited by the hardware. For the best performance, it is necessary to have the number of possible combinations within this limit. Rule of thumb says that for every CUDA core there are 16 threads available. Maxwell GPU included in Jetson Nano has 128 CUDA cores. This means there are 2048 threads available. Using calculation for the combinations of 2-level VSI, it is possible to have the prediction horizon with length $N = 3$.

The computation time of the algorithm was influenced by the platform on which it was implemented. The maximal time required by the control algorithm itself was around 15 µs, but the memory manipulation and running APIs needed another 50 µs. This resulted in a sampling period of 100 µs.

B. Simulation

For the testing of the proposed algorithm PIL simulation was created. Its topology is in the Figure 1. For the communication between Jetson Nano and the Simscape model was used UDP protocol.

Data sent from the simulation (marked x_m in the figure) are processed by the CPU and sent to the GPU. The control algorithm is performed in the GPU. By this, the optimal switching state is found and its code is sent in the form of the number to the CPU. CPU decodes the switching state and sends it to the simulation.

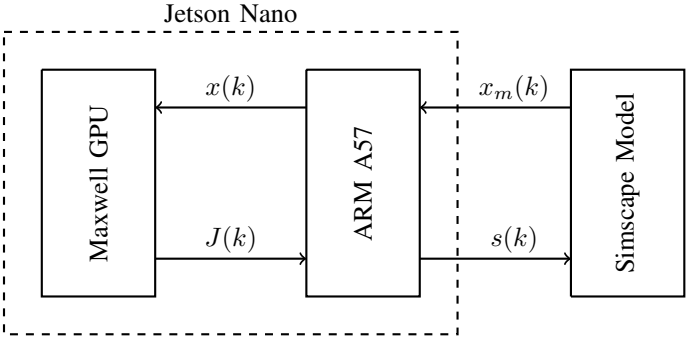

Fig. 1. Simulation topology

Parameters of the motor are shown in the Table I. Table II shows the parameters of the inverter and the parameters of the controller are shown in the Table III.

1) Reference tracking: First of all, the ability to ensure reference tracking was tested. This test was performed on the reference signal composed of the ramp which slope was larger then maximal torque generated by the motor. This style of ramp was preferred to reference step because steps are not usually used in practical applications. The ramp rises until rated angular speed is reached. After that, the reversibility of the algorithm is tested by slowing motor down and changing sense of the rotation. Finally, reference is restored to $0\ \mathrm{rad\,s^{-1}}$.

The results of the experiment are shown in the Figure 2. As it was expected motor was not able to track the initial ramp of the reference. After that, all demands were met: the controlled plant was able to keep rated speed and then to reverse.

Currents during the reference tracking experiment are shown in the Figure 3. Mean value of the quadrature component of the current during the initial ramp of the reference was 8.9364 A. Generated torque was about $89.4\,\%$ of the rated torque. The important part is whether the rated current was exceeded by the value of current or not. This

TABLE I
PARAMETERS OF PMSM

Parameter	Value
R_s	$0.822 \quad \Omega$
L_d	$0.016 \quad \mathrm{H}$
L_q	$0.024 \quad \mathrm{H}$
Ψ_{PM}	$0.097 \times 10^{-3} \quad \mathrm{Wb}$
Pp	5
J	$0.870 \times 10^{-3} \quad \mathrm{kg\,m^2}$
ω_r	$150 \quad \mathrm{rad\,s^{-1}}$
T_R	$7.275 \quad \mathrm{N\,m}$
I_R	$10 \quad \mathrm{A}$

TABLE II
PARAMETERS OF THE INVERTER

Parameter	Value
Type	2-level VSI
DC-BUS	$200 \quad \mathrm{V}$

TABLE III
PARAMETERS OF THE CONTROLLER

Parameter	Value
w_{TE}	3.2×10^7
w_{i_d}	2.5
w_{IC}	30
N	3
T_s	$100 \quad \mu\mathrm{s}$

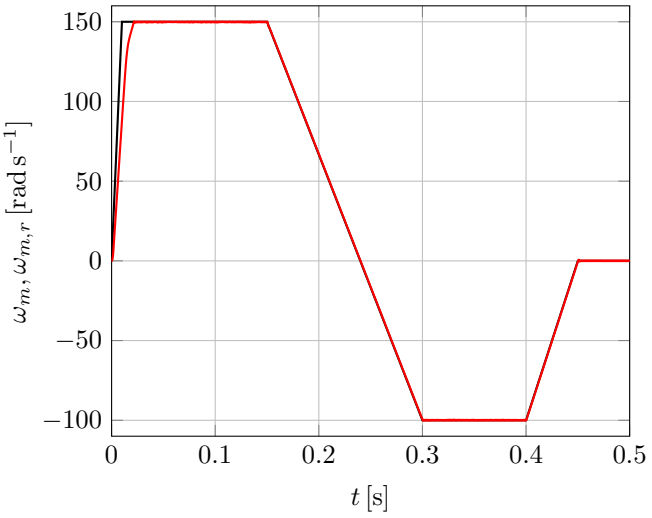

Fig. 2. Results of the reference tracking experiment; black - reference, red - measured speed.

can be seen in the Figure 4. Little overshoot occurred during the settling of the speed on its rated value. Rated current was exceeded by $1\,\%$ of its value, which can be acceptable.

2) Change of the load torque: This experiment tests whether the controller is able to compensate one of the most common disturbances - change of the load torque. While the motor was rotating by steady angular speed $100\,\mathrm{rad\,s^{-1}}$, two changes of the load torque were applied. First, the value of the load torque was changed to $2\,\mathrm{N\,m}$. After the stabilization of the angular speed, load torque was changed to $-2\,\mathrm{N\,m}$.

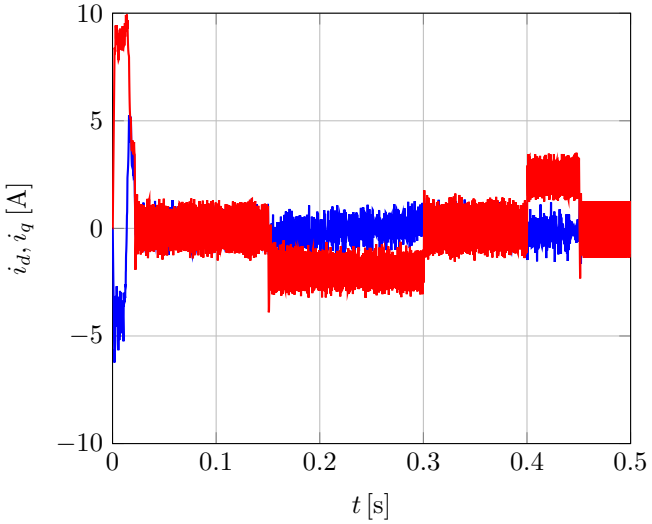

Fig. 3. Currents during reference tracking experiment; blue - i_d, red - i_q.

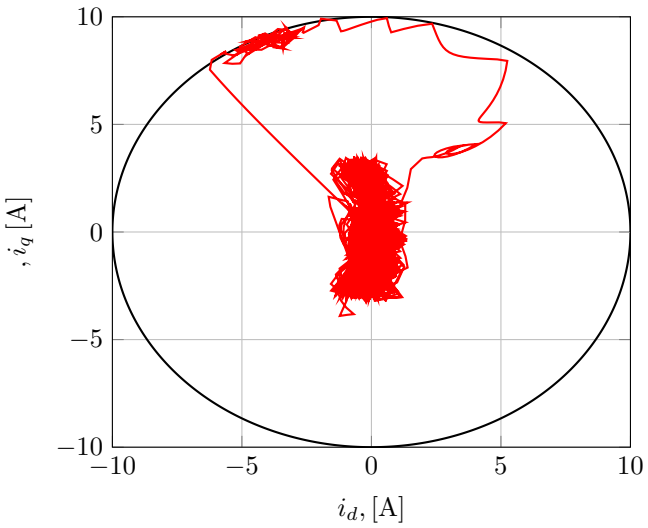

Fig. 4. Current in dq reference frame

In the Figure 5 are the results shown. The first change, occurred in $t = 2$ s resulted in undershoot lesser than $1\,\%$ and was compensated after 20 ms. The second change led to the oscillations with the biggest deviation of $2\,\%$ from the reference. These oscillations were caused by the absolute difference in the load torque, which was $4\,\mathrm{N\,m}$. This can be seen in the Figure 6. The value of the current i_q had to perform step change and after that, the oscillations occurred. When the oscillations stopped, settling of the measured angular speed was similar to the one in the first part of the experiment. Also, it can be seen current did not exceed its limit.

V. CONCLUSION

This paper proposed a novel approach to finite control set model predictive control. This approach is based on searching for the optimal switching state of the three-phase two-level

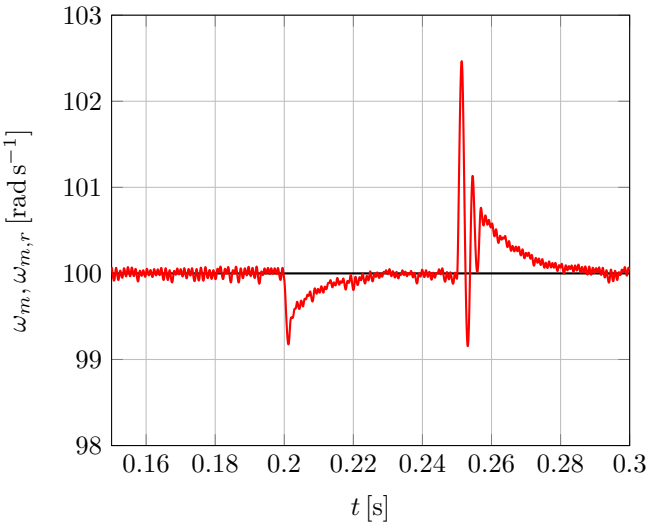

Fig. 5. Angular speed during load torque experiment; black - reference, red - measured speed

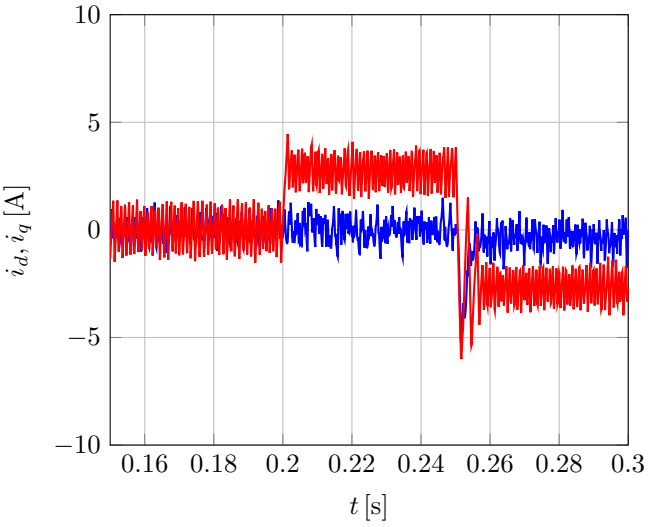

Fig. 6. Currents during load torque experiment; blue - i_d, red - i_q.

voltage source inverter. The optimum is found using general-purpose computing on the GPU by computing each possible combination for a given prediction horizon. Based on the value of the cost function, the optimal switching state is chosen.

The proposed algorithm was tested using PIL simulation with Simscape model for the simulation of the inverter and PMSM and the embedded device Jetson Nano for the execution of the control algorithm. Limitations of the used hardware were mentioned. The sampling period was chosen accordingly to them.

The simulation proved that the controlled plant was able to track the reference of the angular speed. Also, the controller was able to ensure compensation of the disturbance in the form of the load torque. Results are discussed, especially whether the limits of the current were met or not.

During the writing of this article, the algorithm is being implemented on the real physical system. Further research will contain experimental results of testing introduced algorithm on this physical system.

VI. Acknowledgment

This research has been financially supported by the Ministry of Education, Youth and Sports of the Czech republic under the project CEITEC 2020 (LQ1601) and by the ECSEL Joint Undertaking under Grant 826060/8A19001 (AI4DI).

References

[1] P. Cortes, M. P. Kazmierkowski, R. M. Kennel, D. E. Quevedo, and J. Rodriguez, "Predictive control in power electronics and drives," *IEEE Transactions on Industrial Electronics*, vol. 55, pp. 4312–4324, Dec 2008.

[2] H. Deng and T. Ohtsuka, "A parallel Newton-type method for nonlinear model predictive control," *Automatica*, vol. 109, p. 108560, 2019.

[3] M. Graf, L. Buchta, and L. Pohl, "Nonlinear predictive controller design of PMSM with field weakening performance," *IECON Proceedings (Industrial Electronics Conference)*, pp. 3001–3005, 2013.

[4] Z. Mynar, L. Vesely, and P. Vaclavek, "PMSM Model Predictive Control with Field Weakening Implementation," *IEEE Transactions on Industrial Electronics*, vol. 0046, no. c, pp. 1–1, 2016.

[5] S. Vazquez, J. Rodriguez, M. Rivera, L. G. Franquelo, and M. Norambuena, "Model Predictive Control for Power Converters and Drives: Advances and Trends," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 2, pp. 935–947, 2017.

[6] M. Preindl and S. Bolognani, "Model predictive direct torque control with finite control set for pmsm drive systems, part 2: Field weakening operation," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 2, pp. 648–657, 2013.

[7] Q. Liu and K. Hameyer, "A finite control set model predictive direct torque control for the pmsm with mtpa operation and torque ripple minimization," in *2015 IEEE International Electric Machines Drives Conference (IEMDC)*, pp. 804–810, May 2015.

[8] F. Wang, X. Mei, J. Rodriguez, and R. Kennel, "Model predictive control for electrical drive systems-an overview," *CES Transactions on Electrical Machines and Systems*, vol. 1, no. 3, pp. 219–230, 2017.

[9] M. Siami, D. A. Khaburi, and J. Rodriguez, "Simplified Finite Control Set-Model Predictive Control for Matrix Converter-Fed PMSM Drives," *IEEE Transactions on Power Electronics*, vol. 33, no. 3, pp. 2438–2446, 2018.

[10] B. S. Riar, T. Geyer, and U. K. Madawala, "Model predictive direct current control of modular multilevel converters: Modeling, analysis, and experimental evaluation," *IEEE Transactions on Power Electronics*, vol. 30, no. 1, pp. 431–439, 2015.

[11] S. Wendel, A. Dietz, and R. Kennel, "Fpga based finite-set model predictive current control for small pmsm drives with efficient resource streaming," in *2017 IEEE International Symposium on Predictive Control of Electrical Drives and Power Electronics (PRECEDE)*, pp. 66–71, 2017.

[12] K. S. Oh and K. Jung, "GPU implementation of neural networks," *Pattern Recognition*, vol. 37, no. 6, pp. 1311–1314, 2004.

[13] S. Ohyama and H. Date, "Parallelized nonlinear model predictive control on GPU," *2017 Asian Control Conference, ASCC 2017*, vol. 2018-January, pp. 1620–1625, 2018.

[14] S. Boyd and L. Vanderberghe, *Convex optimization*. Cambridge university press, 2004.