IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

An Efficient Implementation of Online Model Predictive Control with Field Weakening Operation in Surface Mounted PMSM

OKAN ARPACIK[1,2], (Student Member, IEEE), MUSTAFA MERT ANKARALI[2], (Member, IEEE)
[1]Defense Systems Technologies Division, Aselsan Inc., Ankara, 06370, Turkey (e-mail: oarpacik@aselsan.com.tr)
[2]Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara, 06800, Turkey (e-mail: mertan@metu.edu.tr)

This work was supported in part by the Aselsan Inc. The methods and some of the materials presented in this paper have been reported in the first author's MSc thesis [1].

ABSTRACT Model-predictive-controller (MPC), one of the optimal control policies, has gained more attention in servo drive and other industrial applications in recent years due to evident control performance benefits compared to more classical control methods. However, an MPC algorithm solves a constrained optimization problem at each step that brings a substantial computational burden over classical control policies. This study focuses on improving the computational efficiency of an online MPC algorithm and then demonstrates its practical feasibility on the field weakening operation in high-speed PMSM control applications where the sampling frequency is in the order of μs. We implement the existing dual active set solver by replacing two standard methods in the matrix update step to reduce the overall computational cost of the algorithm. We also rearrange the linear approximation for the constraints on voltage and current by taking the tradeoff between accuracy and speed into account. We finally verify the efficiency and effectiveness of the proposed structure via processor-in-the-loop simulations and physical platform experiments.

INDEX TERMS Field weakening, online model predictive control (MPC), quadratic programming (QP), synchronous machine

I. INTRODUCTION

MODEL predictive control (MPC) is the name of a modern family of model-based constrained optimization-based controllers. Its relatively simple objective, ability to satisfy the constraints on both input and state trajectories, and explicitly handling multi-input-multi-output systems are the core features of the model predictive control [2], [3]. Due to its high computational complexity compared to the classical approaches, the initial practical implementations of MPC concentrated on systems with slow plant dynamics and relatively mild control performance specifications. However, as the computational power of embedded platforms evolved, the adoption of MPC in a wide range of control applications, including but not limited to high-bandwidth plants and challenging performance specifications, has also increased in the past two decades [4]. Nowadays, it is even possible to see MPC implementations in the control of power electronics and electrical drives where sampling frequency requirements are several orders of magnitude higher than ones used in the initial deployments of MPC [5], [6].

Permanent magnet synchronous motor (PMSM) is one of the most common electric machines used in various industries due to its torque-speed characteristic and reliability for long-term use. One of the most common techniques to control the PMSM is vector control, also known as field-oriented control (FOC). Increased demand for high-performance operations in electrical drive systems has led engineers to adopt MPC and similar advanced model-based optimal algorithms for the control PMSM machines.

Even though MPC provides a solid framework for advanced and high-performance control system design, it brings some undeniable disadvantages, precisely extra computational burden and increased implementation complexity, over dominantly adopted classical approaches. In that

respect, researchers in the MPC domain mainly focus on solving computational and implementation complexity problems. Bemporad et al. [7], [8] proposed the explicit quadratic regulator for constrained systems which technically pushes the majority of the computations of the MPC algorithm to an offline pre-processing phase that increased the applicability and popularity of the MPC in experimental and industrial platforms. Specifically, this method generates a lookup table of Piece-wise Affine (PWA) functions that can radically reduce the execution time compared to online MPCs with a trade-off that now demands a boost in memory size can grow dramatically with the dimension of the system and constraints.

In the literature, there exist studies that integrated MPC-type algorithms for the control of electrical drive systems. Several studies adopted online optimization-based MPC algorithms for regulating torque output of PMSMs (MP-TC) [9], [10]. In addition to these, some researchers also integrated field weakening operation with predictive control for some applications that require over-speed capabilities of the motors [11]–[15]. Like our motivation in this paper, Mynar et al. [12] implemented explicit MPC in a "high-performance" dual-core i7 processor and consuming a large amount of memory space. The authors also intended to improve their methodology to make it implementable with a low-cost embedded platform with far less CPU power and memory space in the future work section of their paper. Most of the other studies [11], [14], [15] adapted Finite Control Set (FCS) which basically finds the optimal switching position instead of producing the duty cycle as in Continuous Control Set(CCS) [12]. This approach couples the sample and switching frequency that causes to operate the inverter at variable frequency.

The scarce resources in online CCS-MPC with the field weakening operation and its implementation in low-cost platform boost us to investigate this application. The challenge of reducing the computational complexity in the online optimization problem with sample time in order of μs is another motivation source. In this context, we first focus on increasing the computational capability of the MPC algorithm and then demonstrate its practical feasibility on the field weakening operation in a high-speed PMSM control application, which are the core contributions of the paper. We adopt the existing dual active set solver with the combination of the two standard matrix update methods to increase the algorithm's efficiency. We also compare our implementation with Active Set Solver inside MPC Toolbox provided by Matlab to showcase a true advantage. As a further contribution, we address hints and some suggestions for the ease of practical implementation. In this paper, we employ an efficient implementation of MPC with field weakening operation for PMSM and perform the MPC in both PIL simulations and experimental setup by using different type processors.

This paper is organized as to give the mathematical modeling and field weakening operations in section-II. The quadratic programming that depends on the dual active set solver is expressed in section-III. PIL simulation and experimental results are given in section-IV. In section–V, we conclude this paper via a discussion of the paper.

II. MODELING & ANALYSIS OF PERMANENT MAGNET SYNCHRONOUS MOTORS

MPC is a state-space domain model-based control policy; thus, we need to derive a state-space representation for the PMSM system. We utilize synchronous reference frame model (d-q model) because it makes the electrical variables stationary in the steady-state [16] and enables the regulation of torque and flux content separately (vector control) to achieve the maximum efficiency and perform the field-weakening operation. Following equations models the dynamics for a PMSM based on $d-q$ reference frame model [17].

$$\frac{di_d(t)}{dt} = \frac{1}{L_d}(V_d(t) - R_s i_d(t) + \omega_e(t) L_q i_q(t)) \tag{1a}$$

$$\frac{di_q(t)}{dt} = \frac{1}{L_q}(V_q(t) - R_s i_q(t) - \omega_e(t)(L_d i_d(t) + \lambda_m)) \tag{1b}$$

$$T_e(t) = \frac{3}{2} pp(\lambda_m i_q(t) + (L_d - L_q) i_d(t) i_q(t)) \tag{1c}$$

where

λ_m	magnetic flux of the PM	Wb
$i_d - i_q$	*d axis*, *q axis* current	A
$V_d - V_q$	*d axis*, *q axis* voltage	V
R_s	stator phase resistance	Ω
$L_d - L_q$	*d axis*, *q axis* inductance	H
ω_e	rotor electrical speed	rad/s
T_e	electromechanical torque output	Nm
pp	number of pole-pairs	$-$

Finding a linear state-space approximation that can accurately capture the PMSM dynamics is critical for implementing (linear) MPC algorithms. In this study, we follow the approximation by Bolognani et al. [16] and treat the bi-linear terms, $\omega_e(t) i_d(t)$ and $\omega_e(t) i_q(t)$, as new state variables and further assume that they are constant over prediction horizon. We also integrate the measured rotor speed as an exogenous input and again assume that it is constant over the prediction horizon. As a result, the state- and input-vectors of the linear state-space PMSM model takes the form $[i_d \; i_q \; \widehat{\omega_e i_d} \; \widehat{\omega_e i_q}]^T$ and $[V_d \; V_q]^T$ respectively. We finally discretize the CT state-space model via the forward-Euler method with sample time T_s to obtain the discrete-time state-space model structure as

$$x_{k+1} = Ax_k + B_u u_k + G w_k \quad , \quad y_k = Cx_k \tag{2a}$$

$$A = \left[\begin{array}{cc|cc} 1 - \frac{T_s R_s}{L_d} & 0 & 0 & \frac{T_s L_q}{L_d} \\ 0 & 1 - \frac{T_s R_s}{L_d} & -\frac{T_s L_q}{L_d} & 0 \\ \hline \multicolumn{2}{c|}{\mathbf{0}_{2\times 2}} & \multicolumn{2}{c}{\mathbf{I}_{2\times 2}} \end{array}\right] \tag{2b}$$

$$B = \left[\begin{array}{cc} \frac{T_s}{L_d} & 0 \\ 0 & \frac{T_s}{L_q} \\ \hline \multicolumn{2}{c}{\mathbf{0}_{2\times 2}} \end{array}\right], G = \left[\begin{array}{c} 0 \\ -T_s \frac{\lambda_m}{L_q} \\ \hline \mathbf{0}_{2\times 1} \end{array}\right], C = \left[\begin{array}{c} \mathbf{I}_{2\times 2} \\ \hline \mathbf{0}_{2\times 2} \end{array}\right]^T$$

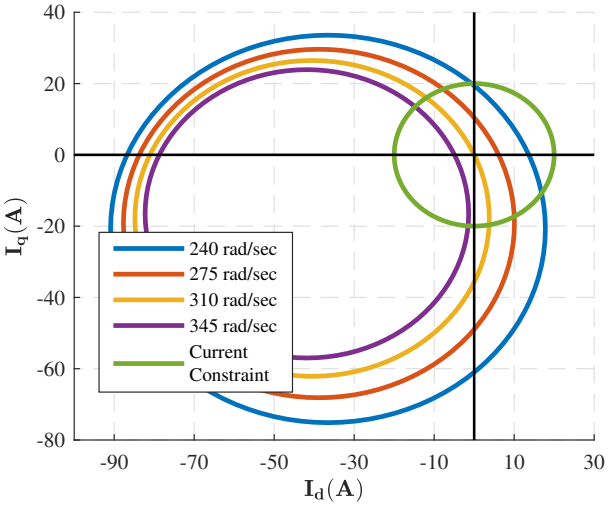

FIGURE 1. Current and voltage circles for different speed values of the PMSM.

Even if all state variables in the model are measurable in real-time, the experimental system unavoidably suffers from some measurement noise and process model uncertainty. For this reason, instead of using potentially noisy measurements directly, which causes undesirable characteristics both during transients and steady-state, we pass the real-time measurements from a steady-state Kalman filter in the experimental setup to obtain a smoother state and input trajectories. In this context, MPC uses the output of the filter at each step in the computation of control actions, which is very a common practice in practical MPC applications [9], [18].

A. FIELD WEAKENING OPERATION

The torque generated by the PMSM given by (1c) has two sources: permanent magnet flux and rotor saliency. In a surface-mount PMSM, d-q axis inductances are almost equal to each other ($L_d \approx L_q = L$), and the torque is generated dominantly by the magnet flux and q axis current. Therefore, in the normal operation of PMSM, i.e., below-rated speed, d axis current is set to 0 ($i_d = 0$) to produce maximum torque per amperes (MTPA). In order to exceed the rated speed, negative i_d values are used to limit the back EMF ($\omega_e L_d i_d + \omega_e \lambda_m$) by creating an opposite magnetic flux to the permanent magnet in the field-weakening region. A given i_q^* reference can only be achieved if a certain amount of negative i_d^* current is applied. Applying negative i_d current to reduce the back EMF so that the PMSM gives more torque output beyond the rated speed is called a field-weakening operation. It technically modifies the current references (i_d^*, i_q^*) at the operating speed to get maximum torque output. A common way of vector control of the PMSM is realized by a voltage source inverter, which includes 6 transistors fed by a DC voltage source. Therefore, electrical sources are limited to DC value of the voltage source (V_{max}) and the maximum current capability of the transistors (I_{max}). These limits are

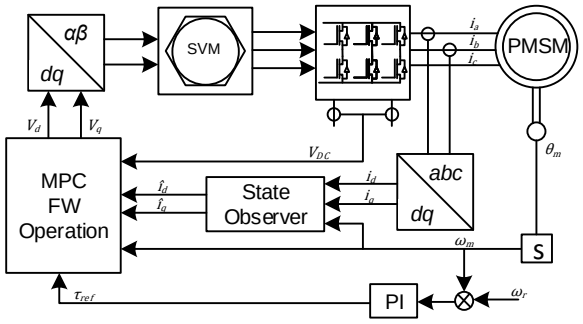

FIGURE 2. The schematic of the proposed structure for field weakening operation.

related to $d-q$ axis variables as

$$V_{max}^2 \geq V_d^2 + V_q^2 \quad , \quad I_{max}^2 \geq I_d^2 + I_q^2 \tag{3}$$

In the steady-state, derivative terms are equal to zero in (1a) and (1b). Rewriting (3) by substituting steady-state forms of (1a) and (1b) yields

$$\frac{V_{max}^2}{R_s^2 + L^2\omega_e^2} \geq (i_d + \frac{L\omega_e^2\lambda_m}{R_s^2 + L^2\omega_e^2})^2 + (i_q + \frac{R_s\omega_e\lambda_m}{R_s^2 + L^2\omega_e^2})^2 \tag{4}$$

The current equation in (3) represents the inner region of a circle whose center is at the origin and radius is equal to (I_{max}). This circle is called the *current circle*. On the other hand, the voltage equation represents the inner region of the *voltage circle*, for which the center and radius are equal to

$$\text{center} = \left(-\frac{L\omega_e^2\lambda_m}{R_s^2 + L^2\omega_e^2}, -\frac{R_s\omega_e\lambda_m}{R_s^2 + L^2\omega_e^2} \right), \tag{5}$$

$$\text{radius} = \frac{V_{max}^2}{\sqrt{R_s^2 + L^2\omega_e^2}}, \tag{6}$$

respectively. The VSI cannot operate outside the current and voltage circles, physically. Therefore, the operating point (i_d, i_q) should be inside the circles at any time. Note that the current circle is stationary while voltage circle parameters are functions of the rotor electrical speed. To set a proper reference operating point (i_d^*, i_q^*) voltage circle equation should be dynamically calculated and the limit values should be updated. Fig. 1 illustrates the voltage and current circles for a PMSM with $I_{max} = 20A$ for different speed values. As shown in the figures, voltage circle shrinks as the speed increases and separates from the current circle after a critical speed value. 240 rad/s rotor speed value approximately separates constant torque and constant power region. Voltage limit circle crosses the origin approximately at 310 rad/s rotor speed, which is the theoretical no-load speed. It is inevitable to have $i_d < 0$ for higher speed demand than 310 rad/s, even if no load applies to the rotor shaft. Operating point for the PMSM should be inside the intersection of the two circles. Therefore, as the speed increases, maximum achievable i_q is limited below I_{max} value.

III. MPC DESIGN FOR PMSM

The standard structure of FOC utilizes a cascaded loop, which closes the inner-loop with current feedback, whereas the outer-loop with angular velocity feedback. In this paper, we preserve the higher level cascaded topology in the standard architecture but replace the controller of the inner loop with the MPC algorithm. In this context, the task of the MPC block is to track the torque reference that is provided by the velocity loop (or any other high-level control policy) for the q axis. The algorithm is also responsible for dynamically generating the required d axis reference by monitoring q axis current and motor velocity to perform field weakening operation, which enables to produce MTPA at beyond rated speed. Fig. 2 illustrates the overall proposed structure that we implement for the CCS-MPC field weakening operation.

A. MPC BASICS

MPC computes the sequence of control actions over a prediction horizon by using the state-space model of the system given in (2) by predicting the future responses of the system dynamics. The cost function fundamentally determines the desired performance land space of the system, i.e., minimizing a weighted sum of tracking error and control effort, which is dominantly formulated as a quadratic cost function. In addition to reducing the quadratic cost function to achieve the desired level of control performance, MPC must also ensure that state, output, and input trajectories satisfy some constraints, generally formulated as linear equalities and inequalities. In that respect, the sub-problem of MPC at each step turns into a quadratic programming problem. In this context, the quadratic programming problem of the MPC action takes the following form

$$\min_{\Delta u} \sum_{i=1}^{N_p} ||Q^{\frac{1}{2}}(y_{k+i} - r_k)||_2^2 + \sum_{j=0}^{N_u-1} ||R^{\frac{1}{2}}\Delta u_{k+j}||_2^2 \quad (7a)$$

$$\text{subject to: } x_{k+i+1} = Ax_{k+i} + B_u u_{k+i} + Gw_k \quad (7b)$$

$$y_{k+i+1} = Cx_{k+i+1} \quad (7c)$$

$$u_{k+i} = u_{k+i-1} + \Delta u_{k+i} u_i^{min} \leq u_{k+i} \leq u_i^{max} \quad (7d)$$

$$y_i^{min} \leq y_{k+i+1} \leq y_i^{max} \quad (7e)$$

$$\Delta u_{k+j+N_u} = 0 \quad (7f)$$

$$j = 0, \ldots, N_p - N_u - 1, i = 0, \ldots, N_p - 1 \quad (7g)$$

where Q and R are the positive definite weight matrices associated with tracking error and input effort respectively. Feasible values are selected for N_p and N_u so as to keep the balance between capturing the dynamics of the system and reducing the complexity of the resulted QP problem [16].

B. CONSTRAINTS

Unlike the classical linear controllers, MPC does not require add-on implicit structures to handle the system's constraints. Under the MPC umbrella, the controllers explicitly address linear convex constraints (equalities and inequalities) on state variables, system outputs, and control/input signals in the system representation. In PMSM drive applications, upper and lower limits on *voltage* and *current* variables are the most dominant type of adopted constraints. The supply voltage on DC-link enforces a maximum value on the voltage supply to the drive, and it is $V_{max} = V_{DC}/\sqrt{3}$ for space vector modulation [19]. The peak stator current determines an upper bound constraint on the stator current variable.

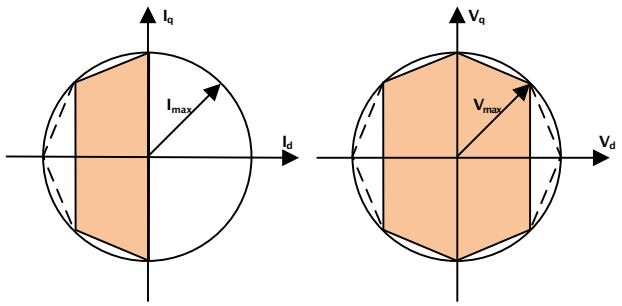

FIGURE 3. The schematic of voltage and current constraint which depends on the liner approximations

We can transform the *voltage* and *current* constraints to 2-norm condition in (d, q) axis plane, such that;

$$\begin{aligned} x \in \mathbb{R}^2 \text{ s.t. } ||x||_2 < I_{max} \\ u \in \mathbb{R}^2 \text{ s.t. } ||u||_2 < V_{max} \end{aligned} \quad (8)$$

We apply two polygonal approximations to transform of these two quadratic constraints into linear form to adapt them into the MPC framework as well as reduce the implementation complexity [16]. As for the voltage limit, we adopt an octagonal shape that combines the two constraint lines into one in the d axis direction. Since the q axis is responsible for generating torque to meet the load torque in the motor shaft, it is undesirable to give all DC-link voltage only for the d axis.This approximation has an acceptable level of underestimation, 70% of the maximum voltage value for the d axis. We can formally define the voltage constraint using the following equation by taking $m = (\sqrt{2} + 1)$;

$$\begin{bmatrix} -\frac{1}{m} & 0 & -\frac{1}{m} & \frac{1}{m} & 0 & \frac{1}{m} \\ 1 & -\frac{m+1}{m} & -1 & -1 & \frac{m+1}{m} & 1 \end{bmatrix}^T \begin{bmatrix} V_d \\ V_q \end{bmatrix} = \begin{bmatrix} V_{max} \end{bmatrix} \quad (9)$$

We modify the constraint adaption above for the current implementation by cropping out the polygon's right half-plane since the only negative current of the d axis is used for field weakening operation. Fig. 3 illustrates the schema for these two constraints. The constraint on current according to the figure is given by the following equation;

$$\begin{bmatrix} -\frac{1}{\sqrt{2}+1} & 0 & -\frac{1}{\sqrt{2}+1} \\ 1 & -\frac{\sqrt{2}+2}{\sqrt{2}+1} & -1 \end{bmatrix}^T \begin{bmatrix} I_d \\ I_q \end{bmatrix} = \begin{bmatrix} I_{max} \end{bmatrix}_{3x1} \quad (10)$$

In practical application, it is highly suggested setting the voltage limit to DC-link voltage measurement, if available, in every step for the case of any change in voltage that may be different from its constant value.

C. COMPUTING CONTROL ACTION

Solving the optimization problem in each sample plays a key role in the constraint MPC. It is possible to transform MPC formulation into a convex quadratic optimization problem with linear constraints for linear-dynamical-systems. Thus, researchers and engineers have long been focused on adapting quadratic programming(QP) techniques in the MPC domain. The transformation of the MPC problem into a formulation of parametric quadratic programming is as follows:

$$\min_{z} \ \frac{1}{2} z^T H z + z^T g_k \tag{11a}$$

$$\text{s.t. } Wz \leq b_k \tag{11b}$$

where g_k and b_k are dynamically changing vectors depending on the information of the system at k^{th} step. The idea of the active set method is to impose all constraints as equalities when they are active at the current iteration. The problem is reduced to the sub-problem of QP with equality constraints i.e. $W_{\mathbb{A}} z = b_{\mathbb{A}}$ where $\mathbb{A}$ denotes active set. The solution is found iteratively starting from a feasible point by adding the violated constraint into or dropping the blocking constraint from the active set in each iteration. The active set method has different variations: *primal*, *dual*, and *primal-dual* [20]. The primal method starts from the primal feasible initial point z^0 and iterates by maintaining primal feasibility in each iteration until dual feasibility is reached [20], [21]. On the other hand, the dual method starts from dual feasible initial points z^0,λ^0. It aims for primal feasibility, and maintains dual feasibility in each iteration until no violated constraints exist [22], [23]. In this work, we use a dual-active set solver that is based on the works of Goldfarb and Idnani [22]. The pair (z,λ) is the solution of problem (11), if they satisfy the first-order necessary condition by solving the following linear equation:

$$\underbrace{\begin{bmatrix} H & W_{\mathbb{A}}^T \\ W_{\mathbb{A}} & 0 \end{bmatrix}}_{\text{KKT}} \begin{bmatrix} z \\ \lambda \end{bmatrix} = \begin{bmatrix} -g_k \\ b_k \end{bmatrix} \tag{12}$$

The solution of (12) exists if the KKT matrix is not singular. MPC formulation enables that the Hessian matrix H is positive definite i.e. nonsingular, and if $W_{\mathbb{A}}$ is full row-rank in the current iteration, the inverse of the matrix exists. Several direct methods can be applied to explicitly take inversion of the KKT matrix, such as Schur complement, null-space, and range-space approach [21]. The range-space approach is used to express explicit inversion of the KKT matrix.

$$\begin{bmatrix} H & W_{\mathbb{A}}^T \\ W_{\mathbb{A}} & 0 \end{bmatrix} \begin{bmatrix} H^{-1}(I - W_{\mathbb{A}}^T \mathcal{Z} W_{\mathbb{A}} H^{-1}) & H^{-1} W_{\mathbb{A}}^T \mathcal{Z} \\ \mathcal{Z} W_{\mathbb{A}} H^{-1} & -\mathcal{Z} \end{bmatrix} = I \tag{13}$$

where $\mathcal{Z} = (W_{\mathbb{A}} H^{-1} W_{\mathbb{A}}^T)^{-1}$. To calculate the pair (Δz,$\Delta\lambda$), two operators are defined as;

$$\mathcal{K} = \mathcal{Z} W_{\mathbb{A}} H^{-1} \tag{14}$$

$$\mathcal{G} = H^{-1}(I - W_{\mathbb{A}}^T \mathcal{K}) \tag{15}$$

where $W_{\mathbb{A}} \in \mathbb{R}^{n_a x n}$ and n_a denotes the number of the constraints in the active set. Goldfarb and Idnani [22] suggest that the dual-method can use the unconstrained optimum point of the objective function as their initial value. In contrast, the primal method needs an auxiliary algorithm that provides a feasible starting point and active set. Thus, the dual-method, except for some basic matrix operations, does not require a preliminary calculation phase as in the primal method. Also, the dual-method finds the optimal solution with fewer iterations compared to the primal method that makes the dual method more efficient than primal method [22]. Algorithm 1 summarizes the processes in the dual active set solver.

Algorithm 1 Dual Active Set Solver [22]

Input: (H, g, W, b)
Step-1. Initialization of the algorithm:
set $i \leftarrow 0$, $(z,\lambda) \leftarrow (-H^{-1}g, 0)$, $\mathbb{A} \leftarrow \emptyset$, $J \leftarrow Z^{-1}$, $n_a^0 \leftarrow 0$
Step-2. Constraint violation check:
set $q \leftarrow \begin{cases} k \in (\mathbb{I} \backslash \mathbb{A} | (W_k z_k^i - b_k) > 0) \\ 0, \quad \text{otherwise} \end{cases}$
Step-3. Termination condition check:
if $q = 0$ **then**, **stop** and **return** optimal active set: i.e. (z^*, λ^*) and corresponding active set $\mathbb{A}^i(z^*)$
end if
Step-4. Calculate step directions:
$\Delta z^i = \mathcal{G} W_{\mathbb{A}}^T$ (primal direction)
if $n_a^i > 0$ **then**, calculate the dual direction
$\Delta\lambda^i = \mathcal{K} W_{\mathbb{A}}^T$ (dual direction)
end if
Step-5. Calculate step lengths:
$\alpha_{primal}^i \leftarrow \begin{cases} \infty, & \text{if } \|\Delta z^i\| = 0 \\ \frac{W_q^T z^i - b_q}{W_q^T \Delta z^i}, & \text{o/w} \end{cases}$
$\alpha_{dual}^i \leftarrow \begin{cases} \infty, & \text{if } \nexists k \\ -\frac{\lambda_k^i}{\Delta\lambda_k^i}, & \text{o/w} \end{cases}$ s.t. $\underset{k \in \mathbb{A}}{\text{argmin}}(-\frac{\lambda^i}{\Delta\lambda^i} | \Delta\lambda_k^i < 0)$
$\alpha^i \leftarrow min(\alpha_{primal}^i, \alpha_{dual}^i)$
Step-6. Check infeasibility, take partial step in dual space:
if $\alpha_{primal}^i = \infty$ **then**
  if $\alpha_{dual}^i = \infty$ **then**,stop and **return** QP is **infeasible**
  else Drop blocking constraint i.e.
    $\mathbb{A}^{i+1} \leftarrow \mathbb{A}^i \backslash k$: k calculated in **Step-5**
    $\lambda_p^{i+1} \leftarrow \lambda_p^i + \alpha^i$, $\lambda^{i+1} \leftarrow \lambda^i + \alpha^i \Delta\lambda^i$
    $n_a^{i+1} \leftarrow n_a^i - 1$, and $i \leftarrow i + 1$
    Update R and J matrices and go to **Step-4**
  end if
end if
Step-7. Take a full step in primal and dual space
$z^{i+1} \leftarrow z^i + \alpha^i \Delta z^i$, $\lambda^{i+1} \leftarrow \lambda^i + \alpha^i \Delta\lambda^i$, $\lambda_p^{i+1} \leftarrow \lambda_p^i + \alpha^i$
if $\alpha^i = \alpha_{primal}^i$ **then**,Add current violated constraint
  $\mathbb{A}^{i+1} \leftarrow \mathbb{A}^i \cup q$, $n_a^{i+1} \leftarrow n_a^i + 1$, $i \leftarrow i + 1$
  Update R and J matrices and go to **Step-2**
else Drop blocking constraint i.e.
  $\mathbb{A}^{i+1} \leftarrow \mathbb{A}^i \backslash k$: k calculated in **Step-5**
  $n_a^{i+1} \leftarrow n_a^i - 1$, and $i \leftarrow i + 1$
  Update R and J matrices and go to **Step-4**
end if

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3136614, IEEE Access

Output: (z^*, λ^*) and $\mathbb{A}(z^*)$ and return flag

We utilize matrix factorization to calculate primal- and dual-step directions in Step 4 instead of updating operators, $\mathcal{G}$ and $\mathcal{K}$, explicitly in each iteration for computational efficiency. Let Cholesky decomposition of H be

$$H = ZZ^T \tag{16}$$

where Z is the lower triangular matrix, and QR decomposition of L:

$$L = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}, \text{ such that } L = Z^{-1} W_{\mathbb{A}}^T \tag{17}$$

Rewriting the operators in (14) and (15) by replacing Cholesky and QR decompositions, we can obtain

$$\mathcal{G} = J_2 J_2^T, \ \mathcal{K} = R^{-1} J_1, \tag{18}$$

$$\text{where } J = \begin{bmatrix} J_1 & J_2 \end{bmatrix} = Z^{-T} \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \tag{19}$$

Givens Rotations and House-Holder Reflection are two common approaches to compute the QR factorization [24]. In the active set method, there is only single constraint addition to or deletion from active set in each iteration. Therefore, it is more costly to utilize QR decomposition of L in each iteration than updating J and R matrices via numerically stable methods [22]. The algorithm starts with the $J = Z^{-T}$ and $R = \emptyset$ as their initial point and then, continuing to update them whenever $W_{\mathbb{A}}$ matrix changes until the algorithm gives the optimum points. Our implementation uses the House-Holder Reflection method to update J and R matrices in the constraint addition step. When the new constraint is added into active set the relations in (17) becomes;

$$Z^{-1} \begin{bmatrix} W_{\mathbb{A}}^T & W_i^T \end{bmatrix} = Q \begin{bmatrix} R & r_{n_i^1} \\ 0_{(n-n_a)x(n_a-1)} & r_{n_i^2} \end{bmatrix} \tag{20}$$

There is only one rotation matrix applied to the every new constraint to make the R matrix upper triangular due to they enter into the active set from the end. Applying the rotation matrix to each side;

$$Q_n Z^{-1} \begin{bmatrix} W_{\mathbb{A}}^T & W_i^T \end{bmatrix} = Q_n Q \begin{bmatrix} R & r_{n_i^1} \\ 0_{(n-n_a)x(n_a-1)} & r_{n_i^2} \end{bmatrix} \tag{21}$$

$$= \widetilde{Q} \begin{bmatrix} R & r_{n_i^1} \\ 0_{(1)x(n_a-1)} & r \\ 0_{(n-n_a-1)x(n_a-1)} & 0_{(n-n_a-1)x(1)} \end{bmatrix} \tag{22}$$

Since the rotation matrix Q_n is orthogonal, it does not hinder to hold the property (16) after multiplication with Z^{-1} [18]. Our implementation uses Givens Rotation method to update matrices in the constraint deletion from active set. Unlike the constraint addition step, there might be more rotations in the deletion step depending on the constraint place since the deletion can be anywhere inside the active set. We combine two different methods for updating the matrices to increase the algorithm's computational efficiency. We compare the two algorithms in both addition and deletion step over their total floating-point operations (flops) inside the algorithms, enabling us to assess the overall computational cost in the updating process independent of the adopted hardware.

TABLE 1. The comparison of House-Holder and Givens Rotation Methods about their counts of operations

	House-Holder	Givens Rotation
Constraint Addition		
$(+, -, *)\|(\div, \sqrt{\ })$	93|2	84|9
Constraint Deletion		
$(+, -, *)\|(\div, \sqrt{\ })$	147|6	102|9

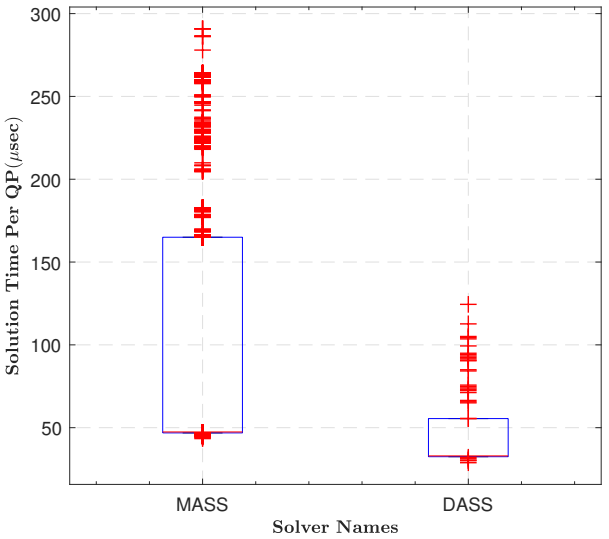

FIGURE 4. Comparison of PIL simulation results of the Active Set Solver provided as standard packages to the MPC toolbox by Matlab(MASS) and our efficient dual active set solver(DASS) implementation under the same conditions.

The number of flops in our measurements also includes $\sqrt{\ }$ operation in addition to 4 basic mathematical operations i.e. $(\pm, *, \div)$. We divide the operations into 2 subgroups; $(\pm, *)$ and $(\div, \sqrt{\ })$ since the number of the cycles to execute the $\div$ and $\sqrt{\ }$ operations costs more than the $(\pm, *)$ in almost all types of computing units. To be more precise, suppose that the active set is empty and let n be the dimension of solution space. In the worst-case scenario, when the first constraint is added into the active set, the methods shall perform $n-1$ calculations to transform R into upper triangular form if the Givens Rotation method is adopted. In contrast, there is a single column operation which enables less number of calculations in the House-Holder Reflection method. Suppose now that the active set has n constraints and the first column i.e. first-in constraint is removed from the set. Both methods now shall perform $n-1$ step to complete updating the process. The number of flops needed to be executed in House-Holder method is almost 50% less than the Givens Rotation in QR decomposition [25]. However, using the fact that the R is

in upper Hessenberg form in constraint deletion that brings more flexibility in executing fewer flops.

Table 1 provides the overall number of flops for constraint addition and deletion steps of both algorithms. The numbers in the table represent the total number in our efficient C-code implementation by taking the dimension of the solution space as 4. The count of $(\pm, *)$ is slightly different in the constraint addition, whereas there is a significant difference in the number of $(\div, \sqrt{\ })$ which is the main part of the execution speed. On the other hand, the main difference comes from $(\pm, *)$ operations in constraint deletion procedure that dominates the overall performance in terms of speed.

To clarify that our implementation is feasible according to the execution speed, we made a computational comparison of our solver with Active Set Solver QP provided as standard packages to the MPC Toolbox by Matlab. In the first trial, we spotted that the solver could not find the solution within the allowed sample time, i.e., 200 μs. We increased the sample time to identify the exact solution time encountered with no overrun during the operations. We run the same PIL simulations except for replacing our solver with Matlab's. Figure 4 shows the solution time of the solver provided by Matlab. It is clear that our solver finds the solution approximately $2.5x$ faster than that of the solver provided by Matlab under the same conditions. The result shows that our implementation is feasible according to execution speed via comparing the solver that finds the solutions with or without the same matrix updating strategy.

IV. RESULTS

We tested and verified the MPC coupled with the developed QP Solver algorithm first in a processor-in-the-loop(PIL) simulation and then on an experimental platform with a PMSM. We chose $C2000$ and $C6000$ processor families provided by Texas Instruments for PIL simulation and experiments since they are cost-effective embedded platforms widely used in industrial motion control applications. We optimized (for computational efficiency) and implemented our MPC algorithm (with a custom QP solver) in the C programming language. We also tested our codes in $F28377S$ and $C6713B$ processors to ensure the consistency of our implementation in different types of processors with different specifications.

The MPC design parameters along with all the related motor parameters are listed in Table 2. Since the algorithm uses a cascaded structure, two different sample times are scheduled; $T_s^{fast} = 200\mu s$ is for current controller, and $T_s^{slow} = 1ms$ is for speed controller.

A. PIL SIMULATION RESULTS

To analyze the efficiency of the MPC implementation, we performed PIL simulations and computed the execution speed and memory usage of the proposed algorithm. PIL simulation environment enables effective debugging of the algorithmic implementation and provides the execution speed/time of selected blocks using the associated CPU timer. Thus, it is possible to detect the part of the algorithm where the primary computational lies. We chose MATLAB/Simulink environment to perform PIL simulation in the $F28377S$ processor. Since the main reason for PIL simulation is to test the feasibility of our implementation, we only evaluated the tracking performances under constraints and the results on the execution time presented in Fig. 6.

Fig. 5 illustrates the tracking performances of the proposed algorithm for both speed and current loop. Since the maximum reachable speed of the PMSM at rated voltage level is 310 rad/s, the i_d current enters the scene at $2s$ to weak the flux to achieve the desired speed setpoint, i.e., 320 rad/s.

TABLE 2. PMSM and CONTROLLER Parameters

Parameter	Value
(J, B)	$(6.10^{-3} kgm^2, 49.10^{-5} Nm/(rad/s))$
λ_m	0.0106 Wb
R_s	120 $m\Omega$
$L_d \approx L_q$	220 μH
Torque Constant	0.09 Nm/A_{rms}
pp	4
T_s	200μs
(N_p, N_u)	$(4, 2)$
(Q, R)	$(\mathbf{I}_{2\times2}, \frac{1}{20}\mathbf{I}_{2\times2})$
(P, I)	$(2, 0.5)$
V_{max}	$24/\sqrt{3}$ V
I_{max}	20 A

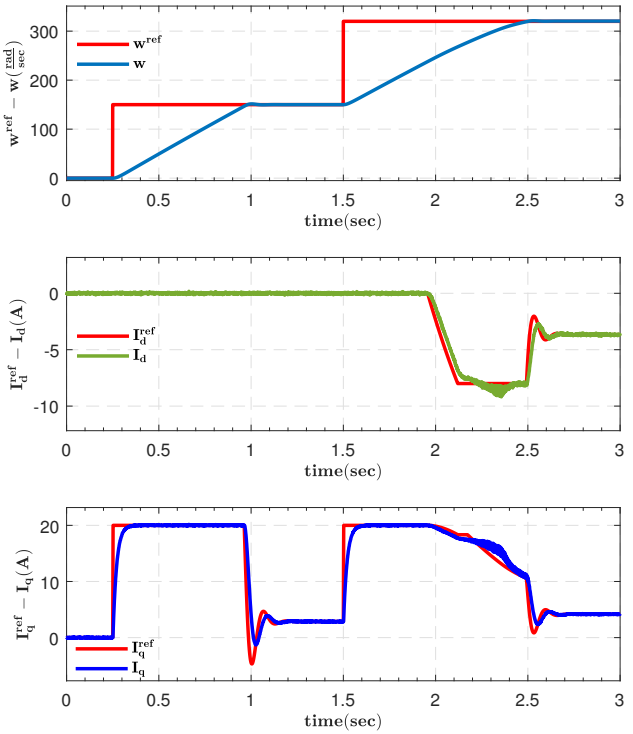

FIGURE 5. PIL simulation results of tracking performances for speed and current references. The i_d current enters the scene after $t = 2s$ to weaken the flux that enables the motor tracks the desired speed value.

B. EXPERIMENTAL RESULTS

After successfully demonstrating the proposed algorithm via PIL simulations, we embedded our MPC algorithm in the $C6713B$ platform and controlled a physical experimental

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3136614, IEEE Access

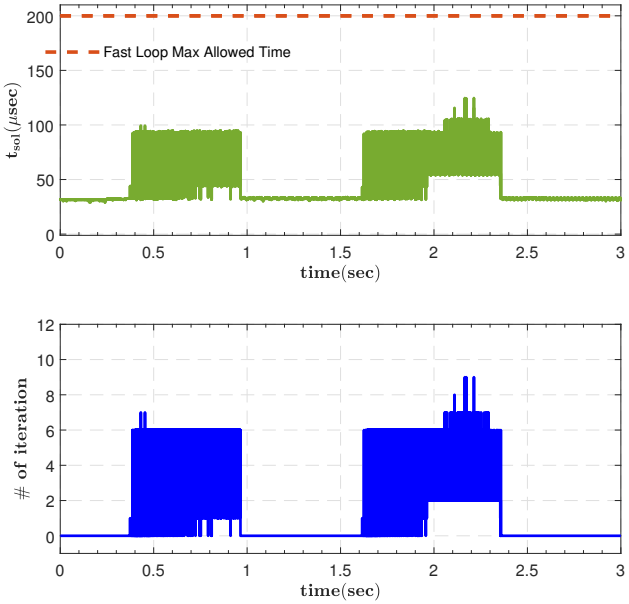

FIGURE 6. Execution time strictly locates under the maximum allowed time through the operation in PIL simulation. The execution time and the number of iterations are consistent except at a point at which the maximum overshoot on i_d current occurs.

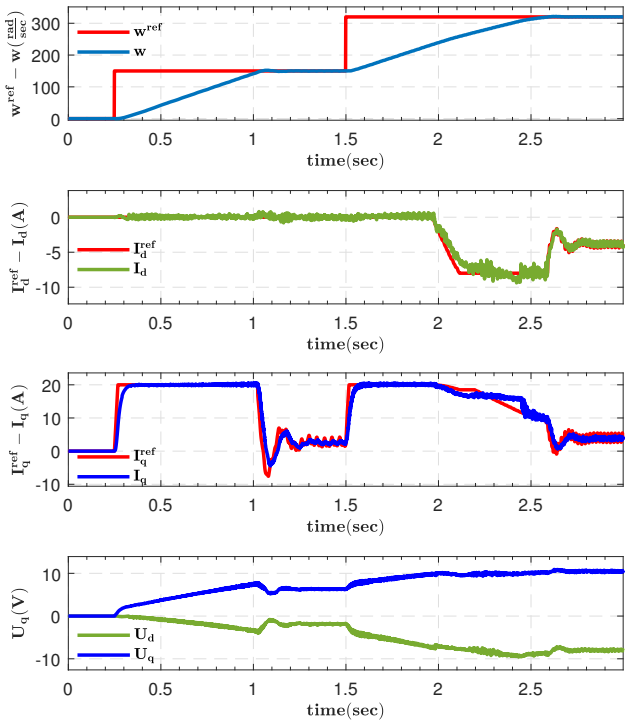

FIGURE 7. Tracking performances of speed and current loops and the voltages which are generated from online MPC in experimental testing.

PMSM setup. Fig. 9 shows the test bench that is used throughout the experimental testing. The setup includes a dynamometer that is connected to the motor shaft and the motor driver unit. We measure the motor's position via a resolver that is attached to the rotor shaft. A resolver to digital converter (RDC) provides the angular velocity measurement.

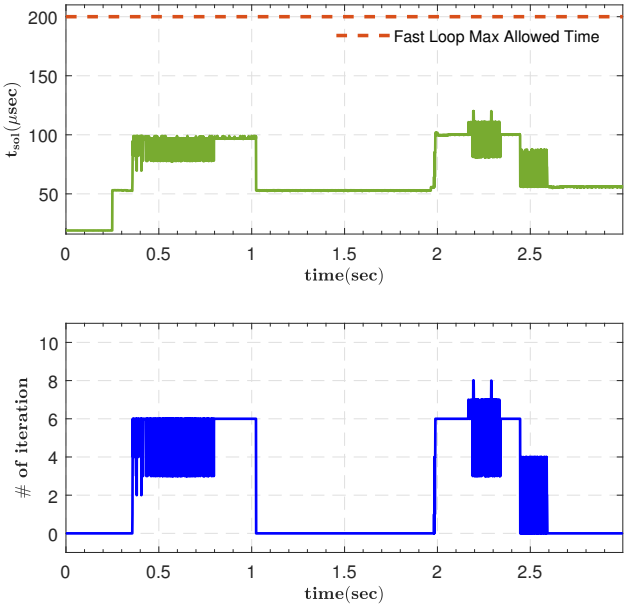

FIGURE 8. Execution time and the number of iteration in the real time experiment to generate the suitable voltages for both axis while satisfying constraints.

Fig. 7 presents the $d-q$ axis voltages and overall tracking performances of the algorithm for speed and current loops. Our experimental scenario split the reference angular velocity signal into two regions to evaluate the "constant"-torque and "constant"-power regions. We feed step-input type reference signals in both zones to push the motor to operate around its limits (torque and power) and stressing the optimization solver. Specifically, at $t = 0$, the system starts at initially at rest condition, and at $t = 0.25$s, we supply the combined cascaded control algorithm a constant angular velocity reference signal of $\omega_{ref} = 150$rad/s until $t = 1.5$s, where we jump the reference signal to $\omega_{ref} = 320$rad/s. In the first zone, i.e. $t \in (0.25, 1.5)$s, the motor dominantly operates at the torque limit (where q axis current is constant) until the motor velocity reaches to the desired value (at $t \approx 1.0$s), and thus angular velocity increases almost linearly during this period.

In the second region, where the desired angular velocity is $\omega_{ref} = 320\text{rad}/s$, the behavior of the motor and controller is similar to the first zone until around $t \approx 2$s, where at that point field weakening operation activates. During this period, the algorithm introduces a negative d axis current that reduces q axis current to respect the maximum torque per ampere criteria. We observe the effect of the linear approximation on the current constraint, especially in the q axis, after $t \approx 2$s since the algorithm puts extra effort to satisfy the constraints that reduce tracking performance. Once the motor angular velocity reaches its final desired value, the algorithm applies consistent $d-q$ axis currents to keep the motor at the desired speed and compensate for the friction. Fig. 8 illustrates the execution time in the experimental test that demonstrates feasibility since computation time always stays inside the sampling time during the operation. One

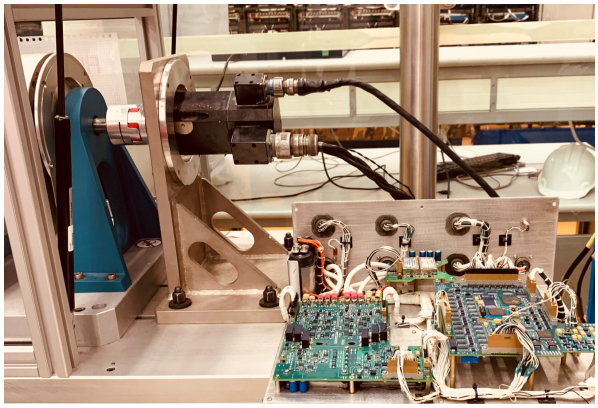

FIGURE 9. Test bench used in experiment of field weakening operation. It consists of PMSM connected to the dynamo and the custom-made driver unit.

should also note that the execution time also includes the ADC readings, their parsing process, and the DAC phases. In addition to execution time, our implementation is also feasible in terms of using minimal memory. The memory occupancy of both algorithm and necessary QP data is only $6kB$ out of $192kB$ in $C6713B$.

We also evaluate the state dependencies between $d-q$ axis current and voltage by picturing the actual motor data to illustrate the performances based on our linear approximation on the constraints. Fig. 10 presents the linear approximation polygons on voltage and current constraints together with real-time experiment data. Apart from the minor deviations on the lines because of noisy measurements and process uncertainty, the electrical state's values always respect the linear constraints.

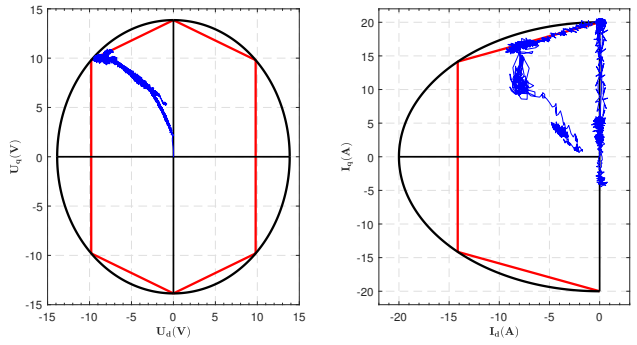

FIGURE 10. The real time experiment values of both voltages and currents lie inside the linear approximation polygons that represents the circle. There is small deviation on the edge of the linear approximations because of the noise level in our measurements.

V. CONCLUSION

In this paper, we demonstrated the feasibility of online MPC with field weakening operation in PMSM by applying the beneficial properties of linear MPC over classical control strategies. In addition to constant power region, we evaluate the performances of the algorithm in constant torque region to demonstrate the feasibility of our implementation under direct-torque control operation.We also successfully carried out the dual active set solver with an efficient matrix updating procedure to find the optimal control signals for the system by satisfying the system constraints. We verified the feasibility of the proposed controller structure via both PIL simulation and physical experiment on the two different processors to certify that our implementation is viable in the low-cost motion control unit. The results demonstrated the practical feasibility and effectiveness of the algorithm to control the PMSM in the laboratory environment.

REFERENCES

[1] O. Arpacik, "An efficient implementation of online model predictive control with practical industrial applications," Master's thesis, Middle East Technical University, 2021.

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

[3] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[4] S. Kouro, M. A. Perez, J. Rodriguez, A. M. Llor, and H. A. Young, "Model predictive control: Mpc's role in the evolution of power electronics," *IEEE Industrial Electronics Magazine*, vol. 9, no. 4, pp. 8–21, 2015.

[5] P. Cortés, M. P. Kazmierkowski, R. M. Kennel, D. E. Quevedo, and J. Rodríguez, "Predictive control in power electronics and drives," *IEEE Transactions on industrial electronics*, vol. 55, no. 12, pp. 4312–4324, 2008.

[6] S. Vazquez, J. I. Leon, L. G. Franquelo, J. Rodriguez, H. A. Young, A. Marquez, and P. Zanchetta, "Model predictive control: A review of its applications in power electronics," *IEEE industrial electronics magazine*, vol. 8, no. 1, pp. 16–31, 2014.

[7] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[8] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit solution of model predictive control via multiparametric quadratic programming," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 2. IEEE, 2000, pp. 872–876.

[9] G. Cimini, D. Bernardini, S. Levijoki, and A. Bemporad, "Embedded model predictive control with certified real-time optimization for synchronous motors," *IEEE Transactions on Control Systems Technology*, 2020.

[10] M. Preindl and S. Bolognani, "Model predictive direct torque control with finite control set for pmsm drive systems, part 1: Maximum torque per ampere operation," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 1912–1921, 2013.

[11] B. Preindl, Matthias and Silverio, "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, 2012.

[12] Z. Mynar, L. Vesely, and P. Vaclavek, "Pmsm model predictive control with field-weakening implementation," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 8, pp. 5156–5166, 2016.

[13] J. Liu, C. Gong, Z. Han, and H. Yu, "Ipmsm model predictive control in flux-weakening operation using an improved algorithm," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 12, pp. 9378–9387, 2018.

[14] Y. Zhang, B. Zhang, H. Yang, M. Norambuena, and J. Rodriguez, "Generalized sequential model predictive control of im drives with field-weakening ability," *IEEE Transactions on Power Electronics*, vol. 34, no. 9, pp. 8944–8955, 2018.

[15] Z. Zheng and D. Sun, "Model predictive flux control with cost function-based field weakening strategy for permanent magnet synchronous motor," *IEEE Transactions on Power Electronics*, vol. 35, no. 2, pp. 2151–2159, 2019.

[16] S. Bolognani, S. Bolognani, L. Peretti, and M. Zigliotto, "Design and implementation of model predictive control for electrical motor drives," *IEEE Transactions on industrial electronics*, vol. 56, no. 6, pp. 1925–1936, 2008.

[17] S. Chai, L. Wang, and E. Rogers, "A cascade mpc control structure for a pmsm with speed ripple minimization," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 8, pp. 2978–2987, 2012.

[18] A. G. Wills, G. Knagge, and B. Ninness, "Fast linear model predictive control via custom integrated circuit architecture," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 59–71, 2011.
[19] M. Preindl, S. Bolognani, and C. Danielson, "Model predictive torque control with pwm using fast gradient method," in *2013 Twenty-Eighth Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*. IEEE, 2013, pp. 2590–2597.
[20] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.
[21] P. E. Gill, N. I. Gould, W. Murray, M. A. Saunders, and M. H. Wright, "A weighted gram-schmidt method for convex quadratic programming," *Mathematical programming*, vol. 30, no. 2, pp. 176–195, 1984.
[22] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical programming*, vol. 27, no. 1, pp. 1–33, 1983.
[23] R. A. Bartlett and L. T. Biegler, "Qpschur: a dual, active-set, schur-complement method for large-scale and structured convex quadratic programming," *Optimization and Engineering*, vol. 7, no. 1, pp. 5–32, 2006.
[24] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013, vol. 3.
[25] N. J. Higham, *Accuracy and stability of numerical algorithms*. SIAM, 2002.

O. ARPACIK received his B.S. degree in control and automation engineering from Istanbul Technical University, Istanbul, in 2017. He is currently working toward the M.Sc. degree in the Middle East Technical University. He is also working in Aselsan Inc. as a control systems engineer. His research interest include model predictive control, online optimization, motor control and control of inertially stabilized platforms.

M. MERT ANKARALI received his B.S. degree in Mechanical Engineering and M.S. degree in Electrical and Electronics Engineering, both from Middle East Technical University (METU) in 2007 and 2010 respectively. He received his PhD in Mechanical Engineering from Johns Hopkins University in 2015. Following his PhD, he was a Postdoctoral Fellow in the same department until 2016. Between 2016 and 2017, he worked as a senior research scientist for Desistek Robotics Corporation. In 2017, he joined the Electrical and Electronics Engineering Department at METU, where he is now an Assistant Professor.