On the SDP Relaxation of Direct Torque Finite Control Set Model Predictive Control

Luca M. Hartmann, Orcun Karaca, *Senior Member*, *IEEE*, Tinus Dorfling, Tobias Geyer, *Fellow*, *IEEE*, Adam Kurpisz

***Abstract*—This paper formulates a semidefinite programming relaxation for a long horizon direct-torque finite-control-set model predictive control problem. In parallel with this relaxation, a conventional branch-and-bound algorithm tailored for the original problem, but with an iteration limit to restrict its computational burden, is also solved. An input sequence candidate is extracted from the solution of the semidefinite program in the lifted space. This sequence is then compared with the so-called early-stopping branch-and-bound solution, and the best of the two is applied in a receding horizon fashion. In simulated case studies, the proposed approach exhibits significant improvements in torque transients, as the branch-and-bound alone struggles to find a meaningful solution due to the imposed limit.**

***Index Terms*—Power electronics, model predictive control, semidefinite programming**

I. Introduction

Direct model predictive control (MPC) methods, which combine control and modulation into one stage by directly manipulating the switching positions of a converter, have garnered significant attention within the power electronics community [1]–[3]. Among these methods, finite-control-set MPC (FCS-MPC) has especially gained prominence due to its intuitive design, straightforward implementation, and high dynamic performance. Since the switch positions are discrete variables, FCS-MPC results in an integer program. Its first variants utilized a horizon of one step and an exhaustive enumeration as solution approach [4], [5]. Subsequent studies revealed that long horizons play a crucial role in mitigating harmonic distortion, and increasing the stability margin [6], [7]. Nonetheless, they render the exhaustive enumeration approach computationally infeasible. Whenever a linear prediction model is available, an efficient variant of a branch-and-bound algorithm called the sphere decoder has been shown to enable the use of long horizons within short sampling intervals [8].

Our interest lies in direct-torque FCS-MPC (DTFCS-MPC), in which the torque and stator flux magnitude are controlled along their references. DTFCS-MPC constitutes an important class of long horizon FCS-MPC. In particular, the prediction models have nonlinear input-output relations. In case the state-update function is linear except a nonlinear operation on the input itself, e.g., the absolute value operator found in the linearized neutral-point dynamics of a three-level neutral-point-clamped (3L-NPC) converter, the work in [9] has shown that one can lift the input space by augmenting it with the so-called "pseudo-inputs" and implement the sphere decoder in the lifted space. When dealing with torque, the output function involves a nonlinear mapping from the states, rendering this approach inapplicable. On the other hand, the slack variables associated with certain nonlinear constraints, e.g., on the switching frequency, have recently been integrated into the sphere decoder, while still maintaining its computational efficiency [10]. Nevertheless, this approach is applicable only if a computationally lightweight, non-trivial, and provable lower bound exists for the future cost to be incurred from the slack variables; and no similar readily computable bound exists for any term involving torque.

For the aforementioned reasons, the long horizon FCS-MPC literature commonly adopts the concept of field orientation [11], see [9], [12]–[14] for example. This notion enables the mapping of torque and flux references to stator current references through outer loops in the synchronous dq-frame aligned with the angular position of the rotor flux vector [15], [16]. By doing so, this approach circumvents the nonlinear relation by capitalizing on the linearity of the current dynamics. Nevertheless, it is widely-recognized that controlling the torque directly maintains a consistent high-bandwidth torque control across various operating conditions [17]–[20], especially when high switching frequencies are not viable, as in medium-voltage applications [21].

An alternative approach to solve DTFCS-MPC is to develop a conventional branch-and-bound algorithm [22], [23]. During transients, when a meaningful initial guess is unavailable, such heuristics may necessitate traversing a large portion of the decision tree before identifying the optimal solution. Since long horizons—thus, large trees—are desirable, this procedure needs to be stopped early to limit the computation time. One such stopping condition occurs when the number of nodes traversed exceeds a certain threshold. Even in the cases where the computationally efficient sphere decoder is applicable, similar limits are imposed [24]. When the limit is reached, suboptimal early-stopping solutions would occasionally be applied. This could compromise control performance and even stability.

In this paper, we formulate the convex semidefinite programming (SDP) relaxation of DTFCS-MPC. Formulating SDP relaxations of such polynomial optimization problems (with or without integer variables) has been extensively studied in the literature, e.g., consider the moment/sum-of-squares hierarchies of [25], [26]. Furthermore, these relaxations have re-

L. M. Hartmann is with University of California San Diego, USA. email: lhartmann@ucsd.edu. O. Karaca and T. Dorfling are with ABB Corporate Research, Switzerland. emails: orcun.karaca@ch.abb.com, martinusdorfling@protonmail.com. T. Geyer is with ABB System Drives, Switzerland. email: t.geyer@ieee.org. A. Kurpisz is with ETH Zürich and BFH, Switzerland. email: adam.kurpisz@ifor.math.ethz.ch.

cently found practical applications in power electronics for offline computation of optimized pulse patterns [27]. In our case, an input sequence candidate is extracted from the solution of the resulting SDP relaxation. While this sequence may only be approximately optimal for minimizing tracking errors, it often offers a consistent and superior solution during transients compared to the early-stopping branch-and-bound approach, and it can be obtained efficiently. Therefore, we solve the SDP relaxation in parallel with the branch-and-bound algorithm and select the better sequence out of the two. This approach introduces a novel and practical use case for convex relaxations in FCS-MPC, paving also the way for future research on theoretical approximation guarantees and hardware-based verification.

Finally, let us contrast this paper with broader control literature beyond converter control. For MPC problems with linear input-output relations and both binary and continuous inputs, SDP relaxations have been formulated in [28]; and [29] demonstrated that they outperform QP-based approaches numerically. In contrast, our problem involves a nonlinear relation and ternary input, however, [29] still provides additional motivation. Alongside the SDP, we employ a continuous relaxation of the lifted integer variables. Continuous relaxations have been studied for MPC, and when they alone yield nonconvex problems that can still be solved to optimality (which is possible depending on the sampling time) one can even prove zero gap under certain structural conditions. For example, [30] presents such an MPC, one example of structure being that the relaxation gives a probability simplex. However, in power electronics, where problems must be solved within microseconds range, a convex relaxation of the nonconvex polynomial optimization becomes necessary to ensure tractability. In this general case, finding theoretical guarantees remains an open research problem [31], except for the quadratic case [32]. Our work shows that the SDP relaxation yields a consistent solution improving tracking during transients where the node-limited branch-and-bound can be arbitrarily poor. The consistency is attributed to the approximate convexity in our problem, which has originally been presented in [33].

Our contributions are as follows:

(i) We propose a branch-and-bound algorithm for FCS-MPC with nonlinear models as in torque and stator flux magnitude.

(ii) We formulate the SDP relaxation for DTFCS-MPC. To the best of our knowledge, this is the first work to do so. Moreover, the resulting formulation differs from the existing works from the mathematical programming community on the SDP relaxations of polynomial optimization, as they work over the Boolean hypercube, and our variables lie in the ternary hypercube, see [34] and the references therein. Although a transformation from the ternary to the Boolean is known, this leads to an increase in dimension and complexity, and does not preserve theoretical guarantees. Our formulation instead relaxes the ternary problem directly into an SDP.

(iii) Case studies show that the extracted input sequence is advantageous during transients, since the node-limited branch-and-bound often fails to find a meaningful input.

The paper is organized as follows. Section II presents the preliminaries, including the modeling, the DTFCS-MPC formulation, and the branch-and-bound algorithm. Section III derives the SDP relaxation, and presents the proposed solution. Finally, Section IV showcases the performance through numerical case studies. Our code is publicly available at https://github.com/lucahart/Torque-tracking.

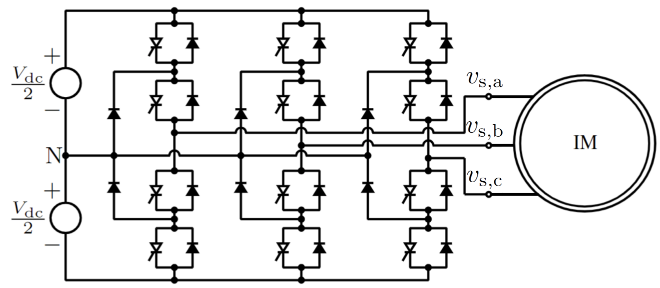


Fig. 1: NPC inverter with an IM and a fixed neutral-point.

Notation: Vectors are denoted bold, $\boldsymbol{x} \in \mathbb{R}^n$. A vector could be formed by restricting the indices, $\boldsymbol{x}_{i:j} = [x_i \, x_{i+1} \cdots x_j]^\top \in \mathbb{R}^{j-i}$. Matrices are capitalized, $\boldsymbol{P} \in \mathbb{R}^{m \times n}$. $\boldsymbol{P}_{i:j,k}$ restricts the matrix to its k^{th} column and to the row range i to j. Denote the trace of $\boldsymbol{P} \in \mathbb{R}^{n \times n}$ by $\operatorname{tr}(\boldsymbol{P})$, whereas $\operatorname{diag}(\boldsymbol{P}) \in \mathbb{R}^n$ denotes the diagonal. The set of $n \times n$ real symmetric matrices is denoted by $\mathbb{S}^n$. The matrix $\boldsymbol{P} \in \mathbb{S}^n$ is positive semidefinite (PSD), if $x^\top \boldsymbol{P} x \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$, and denoted by $\boldsymbol{P} \succeq 0$, i.e., $\boldsymbol{P} \in \mathbb{S}^n_+$. The set $\mathbb{S}^n_+$ defines the PSD cone. $\times$ is the cross-product, whereas $\odot$ is the Hadamard product.

II. Preliminaries

Consider a 3L-NPC converter connected to a squirrel cage induction motor (IM), as depicted in Figure 1. The half dc-link voltages $\frac{V_{\rm dc}}{2}$ are realized by ideal sources, thus fixing the neutral-point to zero. The reader is referred to [9] for details on neutral-point balancing. Throughout the paper, the dynamical equations are based on the $\alpha\beta$-reference frame (i.e., the stationary frame), denoted by $\boldsymbol{\xi}_{\alpha\beta} = [\xi_\alpha \;\; \xi_\beta]^\top$, where $\boldsymbol{\xi}_{\alpha\beta} = \boldsymbol{K}\boldsymbol{\xi}_{abc}$ is the Clarke transformation with $\boldsymbol{K} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}$. The inverse transformation is defined as $\boldsymbol{\xi}_{abc} = \boldsymbol{K}^{-1}\boldsymbol{\xi}_{\alpha\beta}$, where the (pseudo) inverse of $\boldsymbol{K}$ is $\boldsymbol{K}^{-1} = \frac{3}{2}\boldsymbol{K}^\top$. We model the states of IM in terms of its stator and rotor flux linkages $\boldsymbol{\psi}_{{\rm s},\alpha\beta} = [\psi_{{\rm s},\alpha} \;\; \psi_{{\rm s},\beta}]^\top$ and $\boldsymbol{\psi}_{{\rm r},\alpha\beta} = [\psi_{{\rm r},\alpha} \;\; \psi_{{\rm r},\beta}]^\top$. The switch positions are denoted by $\boldsymbol{u}_{\rm abc} = [u_{\rm a} \;\, u_{\rm b} \;\, u_{\rm c}]^\top \in \{-1, 0, 1\}^3$. The stator voltages $\boldsymbol{v}_{\rm s,abc} = [v_{\rm s,a} \;\; v_{\rm s,b} \;\; v_{\rm s,c}]^\top = \frac{V_{\rm dc}}{2}\boldsymbol{u}_{\rm abc}$ are the converter voltages at the ac-side. We drop the subscript abc from $\boldsymbol{u}_{\rm abc}$.

A. Physical system model

The continuous-time flux linkage dynamics of IM are:

$$\frac{d\boldsymbol{\psi}_{{\rm s},\alpha\beta}(t)}{dt} = -R_{\rm s}\frac{X_{\rm r}}{D}\boldsymbol{\psi}_{{\rm s},\alpha\beta}(t) + R_{\rm s}\frac{X_{\rm m}}{D}\boldsymbol{\psi}_{{\rm r},\alpha\beta}(t) + \frac{V_{\rm dc}}{2}\boldsymbol{K}\boldsymbol{u}(t),$$

$$\frac{d\boldsymbol{\psi}_{{\rm r},\alpha\beta}(t)}{dt} = R_{\rm r}\frac{X_{\rm m}}{D}\boldsymbol{\psi}_{{\rm s},\alpha\beta}(t) - \left(R_{\rm r}\frac{X_{\rm s}}{D} - \omega_{\rm r}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\right)\boldsymbol{\psi}_{{\rm r},\alpha\beta}(t),$$

where $R_{\rm s}$, $R_{\rm r}$ are the stator and rotor resistances, respectively; $X_{\rm s}$, $X_{\rm r}$, $X_{\rm m}$ are the stator, rotor, and main inductances, respectively; and $D = X_{\rm s}X_{\rm r} - X_{\rm m}^2$ [33, §4.3]. For the sake of simplicity, via time-scale separation, we treat the rotor's

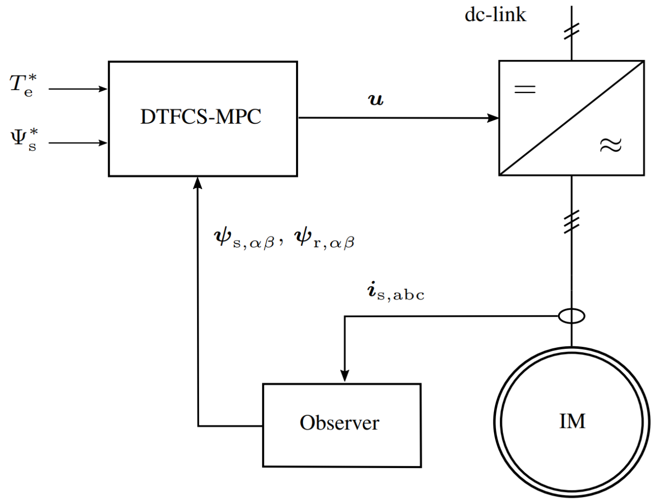


Fig. 2: A high-level DTFCS-MPC block diagram.

angular speed ω_r as a parameter, and discussions on the outer speed controller are considered out-of-scope for this paper.

Using exact Euler discretization with the control sampling interval T_c, the discrete-time model becomes

$$\begin{aligned}\boldsymbol{\psi}_{s,\alpha\beta}(k+1) &= \boldsymbol{A}_{11}\boldsymbol{\psi}_{s,\alpha\beta}(k)+\boldsymbol{A}_{12}\boldsymbol{\psi}_{r,\alpha\beta}(k)+\boldsymbol{B}_1\boldsymbol{u}(k),\\ \boldsymbol{\psi}_{r,\alpha\beta}(k+1) &= \boldsymbol{A}_{21}\boldsymbol{\psi}_{s,\alpha\beta}(k)+\boldsymbol{A}_{22}\boldsymbol{\psi}_{r,\alpha\beta}(k),\end{aligned} \tag{1}$$

where $k \in \mathbb{N}$ is the discrete-time index, and $\boldsymbol{A}_{11}$, $\boldsymbol{A}_{12}$, $\boldsymbol{A}_{21}$, $\boldsymbol{A}_{22}$, and $\boldsymbol{B}_1$ can be inferred from the continuous dynamics.

The control task is to track the references of the electromagnetic torque T_e and the stator flux magnitude Ψ_s. The torque is

$$\begin{aligned}T_e(k) &= \frac{1}{\text{pf}}\frac{X_m}{D}\boldsymbol{\psi}_{r,\alpha\beta}(k)\times\boldsymbol{\psi}_{s,\alpha\beta}(k)\\ &= T_{fct}\boldsymbol{\psi}_{r,\alpha\beta}(k)^\top\boldsymbol{J}\boldsymbol{\psi}_{s,\alpha\beta}(k).\end{aligned} \tag{2}$$

The power factor is pf $= P_{rat}/S_{rat}$, where P_{rat} and S_{rat} are the rated real and apparent power, respectively. Moreover, $T_{fct} = \frac{1}{\text{pf}}\frac{X_m}{D}$, and $\boldsymbol{J} = \begin{bmatrix} 0 & 1\\ -1 & 0\end{bmatrix}$. The stator flux magnitude is

$$\Psi_s(k) = ||\boldsymbol{\psi}_{s,\alpha\beta}(k)||_2 = \sqrt{\psi_{s,\alpha}(k)^2+\psi_{s,\beta}(k)^2}. \tag{3}$$

B. Finite-control-set model predictive control problem

A high-level block diagram of DTFCS-MPC is shown in Figure 2. The prediction horizon is denoted by $N \in \mathbb{N}$. It tracks both the torque and the stator flux magnitude references, usually generated by certain outer loops, e.g., the speed controller. A study on the impact of the observer design is considered out-of-scope for this paper; kindly refer to [35, §5.3] for more details.

Define a compact dynamical system with states $\boldsymbol{x}(k) = [\boldsymbol{\psi}_{s,\alpha\beta}(k)^\top \ \boldsymbol{\psi}_{r,\alpha\beta}(k)^\top]^\top$, input $\boldsymbol{u}(k)$, and output $\boldsymbol{y}(k) = [T_e(k) \ \Psi_s^2(k)]^\top$.[1] The state dynamics are given by $\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}\boldsymbol{u}(k)$, where $\boldsymbol{A}$ and $\boldsymbol{B}$ can be inferred from (1). Note that the state-update function is linear. The output function is nonlinear, and it follows from the definitions of $T_e(k)$ and $\Psi_s(k)$ in (2) and (3), respectively.

[1]Squaring the magnitude in (3) turns it into a polynomial.

Let $e_T(k) = T_e^* - y_1(k)$ and $e_\Psi(k) = \Psi_s^{*2} - y_2(k)$ be the tracking errors for torque and stator flux magnitude, respectively.[2] The objective of DTFCS-MPC is

$$\begin{aligned}f_N(\boldsymbol{U}(k)) =&\sum_{\ell=1}^{N_p}\lambda_T(e_T(k+\ell))^2\\ &+(1-\lambda_T)(e_\Psi(k+\ell))^2+\lambda_u||\Delta\boldsymbol{u}(k+\ell-1)||_2^2,\end{aligned}$$

where $\ell \in \{1,\dots,N\}$ is used to iterate over the prediction horizon. The parameter λ_T sets the trade-off between the two objectives. Additionally, a penalty with the weight λ_u on the switching transitions $\Delta\boldsymbol{u}(k) = \boldsymbol{u}(k)-\boldsymbol{u}(k-1)$ is included to limit the switching frequency. Not penalizing control effort results in a deadbeat behaviour [36, §IV.A]. Here, we also introduce the full-horizon switching sequence $\boldsymbol{U}(k) = [\boldsymbol{u}(k)^\top \ \boldsymbol{u}(k+1)^\top \ \cdots \ \boldsymbol{u}(k+N-1)^\top]^\top$.

DTFCS-MPC solves the following optimization problem at time step k:

$$\begin{aligned}f_N^{opt} =&\min_{\boldsymbol{U}(k)} f_N(\boldsymbol{U}(k))\\ &\text{s.t. } \boldsymbol{x}(k+\ell) = \boldsymbol{A}\boldsymbol{x}(k+\ell-1)+\boldsymbol{B}\boldsymbol{u}(k+\ell-1),\\ &\quad y_1(k+\ell) = T_{fct}\boldsymbol{x}_{3:4}(k+\ell)^\top\boldsymbol{J}\boldsymbol{x}_{1:2}(k+\ell),\\ &\quad y_2(k+\ell) = ||\boldsymbol{x}_{1:2}(k+\ell)||_2^2,\\ &\quad \boldsymbol{u}(k+\ell-1) \in \{-1,\ 0,\ 1\}^3, \quad \forall\ell.\end{aligned} \tag{$\mathcal{P}$}$$

After solving ($\mathcal{P}$) for its optimal solution $\boldsymbol{U}^{opt}(k)$, DTFCS-MPC applies the first input $\boldsymbol{u}^{opt}(k)$ in a receding horizon fashion. As will be discussed further in Section III, ($\mathcal{P}$) is a polynomial optimization problem over integer variables. The objective $f_N(\cdot)$ is of degree 4 (quartic). This class is known to be NP-hard [37]. Let $\mathcal{U}_N = \{-1,\ 0,\ 1\}^{3N}$ denote all possible input sequences for ($\mathcal{P}$). Since $\text{card}(\mathcal{U}_N) = 27^N$, enumeration is computationally infeasible for long horizons, i.e., large N.

C. A branch-and-bound algorithm for DTFCS-MPC

We briefly recall a conventional branch-and-bound algorithm. Let $\boldsymbol{U}^{ini}(k)$ denote the initial solution with the objective value $f_N^{ini} = f_N(\boldsymbol{U}^{ini}(k))$. For example, it can be obtained by an educated guess using a shifted version of the optimal solution of the previous time step $k-1$, i.e., $\boldsymbol{U}^{opt}(k-1)$. Starting with this being the incumbent solution, the algorithm then systematically searches for the optimal solution. Whenever a better input sequence is identified, it updates its incumbent solution. During its exploration, it can shrink the search space based on the objective value of the incumbent solution.

This exploration procedure with shrinking relies on the additive nonnegative structure of the objective. Instead of directly calculating $f_N(\boldsymbol{U}(k))$ for a full candidate solution, we create a solution tree of depth N, where at each node, we select the input $\boldsymbol{u}(k+\ell-1)$ of the time step $k+\ell-1$ at the depth level $\ell \in \{1,\dots,N\}$. As we choose new inputs at later time steps, we can iteratively increment the objective to obtain

$$\begin{aligned}f_\ell =&\sum_{j=1}^{\ell}\lambda_T(e_T(k+j))^2\\ &+(1-\lambda_T)(e_\Psi(k+j))^2+\lambda_u||\Delta\boldsymbol{u}(k+j-1)||_2^2.\end{aligned} \tag{4}$$

[2]For the sake of generality, the references are time-invariant incorporating the fact that the reference changes may not be scheduled events.

Algorithm 1 Conventional branch-and-bound algorithm:

1: **Function:** branch_and_bound
2: **Input:** $f_{\ell-1}$, f_N^{inc}, ℓ, $\boldsymbol{U}$, $\boldsymbol{U}^{\text{inc}}$, $\boldsymbol{x}_{\text{prev}}$, $\boldsymbol{u}_{\text{prev}}$, n_{p}
3: **Output:** $\boldsymbol{U}^{\text{inc}}$
4: **if** $n_{\text{p}} > n_{\text{p,max}}$ **then return** $\boldsymbol{U}^{\text{inc}}$
5: **else** $n_{\text{p}} \leftarrow n_{\text{p}} + 1$
6: **end if**
7: **for all** $\boldsymbol{u} \in \mathcal{U}_1$ **do**
8: $\quad \boldsymbol{x}(k+\ell) \leftarrow \boldsymbol{A}\boldsymbol{x}_{\text{prev}} + \boldsymbol{B}\boldsymbol{u}$
9: $\quad y_1(k+\ell) \leftarrow T_{\text{fct}}\boldsymbol{x}_{3:4}(k+\ell)^\top \boldsymbol{J}\boldsymbol{x}_{1:2}(k+\ell)$
10: $\quad y_2(k+\ell) \leftarrow ||\boldsymbol{x}_{1:2}(k+\ell)||_2^2$
11: $\quad f_\ell \leftarrow f_{\ell-1} + \lambda_{\text{T}}(e_{\text{T}}(k+\ell))^2 + (1-\lambda_{\text{T}})(e_\Psi(k+\ell))^2 + \lambda_{\text{u}}||\boldsymbol{u} - \boldsymbol{u}_{\text{prev}}||_2^2$
12: $\quad$ **if** $f_\ell < f_N^{\text{inc}}$ **then**
13: $\quad\quad \boldsymbol{U}_{3\ell-2:3\ell} \leftarrow \boldsymbol{u}$
14: $\quad\quad$ **if** $\ell < N$ **then**
15: $\quad\quad\quad \boldsymbol{U}^{\text{inc}} \leftarrow$ branch_and_bound(f_ℓ, f_N^{inc}, $\ell+1$, $\boldsymbol{U}$, $\boldsymbol{U}^{\text{inc}}$, $\boldsymbol{x}(k+\ell)$, $\boldsymbol{u}$, n_{p})
16: $\quad\quad$ **else**
17: $\quad\quad\quad \boldsymbol{U}^{\text{inc}} \leftarrow \boldsymbol{U}$
18: $\quad\quad\quad f_N^{\text{inc}} \leftarrow f_\ell$
19: $\quad\quad$ **end if**
20: $\quad$ **end if**
21: **end for**
22: **return** $\boldsymbol{U}^{\text{inc}}$

Whenever the incremental objective f_ℓ exceeds the objective of the incumbent solution, the sub-tree can be discarded.

With this in mind, the branch-and-bound algorithm is presented in Algorithm 1. To initialize, we let $f_0 = 0$, $\ell = 1$, $\boldsymbol{U} = \boldsymbol{U}^{\text{inc}} = \boldsymbol{U}^{\text{ini}}(k)$, $f_N^{\text{inc}} = f_N^{\text{ini}}$, $\boldsymbol{u}^{\text{prev}} = \boldsymbol{u}(k-1)$, $\boldsymbol{x}^{\text{prev}} = \boldsymbol{x}(k-1)$, $n_{\text{p}} = 0$. After deciding upon $\boldsymbol{u}(k+\ell-1)$, the resulting incremental objective (4) is compared to the best so far, i.e., the incumbent solution, in Line 12. We continue to traverse if $f_\ell < f_N^{\text{inc}}$ holds. Otherwise, the sub-tree is pruned. This is repeated until $f_N < f_N^{\text{inc}}$ is found at a leaf node, and then the incumbent solution is updated. The algorithm starts back-tracking. The optimal solution $\boldsymbol{U}^{\text{opt}}(k)$ is found and its optimality is verified if all the branches have been cut. Otherwise, the algorithm stops early if the limit, $n_{\text{p,max}}$, is hit.

The branch-and-bound algorithm has a worst-case exponential time performance. Nevertheless, it is efficient during steady-state operation. During transients, however, the initial solution may not be meaningful. Moreover, in contrast to the sphere decoder of [8], which was proposed for linear systems with integer manipulated variables, it does not incorporate the future cost incurred by an input decision at an earlier time step. In Algorithm 1, the cost is simply incremented by the stage cost incurred at this time step. Thus, it might need to traverse a large portion of the tree to find the optimal solution and to verify its optimality. It is common practice to impose a time limit, e.g., a node limit as in Line 4, which might then result in the algorithm returning suboptimal solutions, and in particular, these will be arbitrarily poor during large transients.

III. SDP RELAXATION OF DTFCS-MPC

This section formulates the SDP relaxation of $(\mathcal{P})$. Different from $(\mathcal{P})$, SDPs are well-studied conic convex optimization problems with many solvers developed to address them.[3] To this end, we first formulate $f_N(\boldsymbol{U}(k))$ explicitly as a function of $\boldsymbol{U}(k)$, the initial state $\boldsymbol{x}(k)$ and the previous input $\boldsymbol{u}(k-1)$. From (1), derive the maps

$$\begin{aligned}\boldsymbol{x}_{1:2}(k+\ell) &= \boldsymbol{\psi}_{\text{s}}(k+\ell) = \boldsymbol{\Gamma}_{\text{s},\ell}\boldsymbol{x}(k) + \boldsymbol{\Upsilon}_{\text{s},\ell}\boldsymbol{U}(k),\\ \boldsymbol{x}_{3:4}(k+\ell) &= \boldsymbol{\psi}_{\text{r}}(k+\ell) = \boldsymbol{\Gamma}_{\text{r},\ell}\boldsymbol{x}(k) + \boldsymbol{\Upsilon}_{\text{r},\ell}\boldsymbol{U}(k),\end{aligned}$$

where $\boldsymbol{\Gamma}_{\text{s},\ell}$, $\boldsymbol{\Gamma}_{\text{r},\ell}$, $\boldsymbol{\Upsilon}_{\text{s},\ell}$, $\boldsymbol{\Upsilon}_{\text{r},\ell}$ are relegated to an online appendix [39, A1]. Notice they are k-invariant, and can thus be computed offline.

Now, augment the full-horizon variable with 1 as follows $\tilde{\boldsymbol{U}}(k) = [1\ \boldsymbol{U}(k)^\top]^\top$. Given the definitions of $T_{\text{e}}(k)$ and $\Psi_{\text{s}}(k)$ in (2) and (3), respectively, we have

$$\begin{aligned}y_1(k+\ell) &= T_{\text{e}}(k+\ell) = \tilde{\boldsymbol{U}}(k)^\top \boldsymbol{Q}_\ell(k)\tilde{\boldsymbol{U}}(k),\\ y_2(k+\ell) &= \Psi_{\text{s}}^2(k+\ell) = \tilde{\boldsymbol{U}}(k)^\top \boldsymbol{W}_\ell(k)\tilde{\boldsymbol{U}}(k),\end{aligned} \tag{5}$$

where $\boldsymbol{W}_\ell(k)$ and $\boldsymbol{Q}_\ell(k)$ are functions of $\boldsymbol{x}(k)$ and relegated to an online appendix [39, A1].

For switching transitions, we have

$$\Delta u_z(k+\ell-1) = \boldsymbol{Z}_{z,\ell}(k)\tilde{\boldsymbol{U}}(k),\ \forall z \in \{\text{a},\text{b},\text{c}\}, \tag{6}$$

where $\boldsymbol{Z}_{z,\ell}$ is relegated to [39, A1]. Its first instance $\boldsymbol{Z}_{z,1}$ is a function of $\boldsymbol{u}(k-1)$, and the rest can be computed offline.

Using the definitions above, the objective of $(\mathcal{P})$ becomes

$$\begin{aligned}f_N(\boldsymbol{U}(k)) = \lambda_T\sum_{\ell=1}^{N}&\Big(T_{\text{e}}^* - \tilde{\boldsymbol{U}}(k)^\top \boldsymbol{Q}_\ell(k)\tilde{\boldsymbol{U}}(k)\Big)^2\\ +(1-\lambda_T)\sum_{\ell=1}^{N}&\Big(\Psi_s^{*2} - \tilde{\boldsymbol{U}}(k)^\top \boldsymbol{W}_\ell(k)\tilde{\boldsymbol{U}}(k)\Big)^2\\ +\lambda_u\sum_{\ell=1}^{N}&\sum_{z\in\{\text{a},\text{b},\text{c}\}} \tilde{\boldsymbol{U}}(k)^\top \boldsymbol{Z}_{z,\ell}(k)^\top \boldsymbol{Z}_{z,\ell}(k)\tilde{\boldsymbol{U}}(k).\end{aligned} \tag{7}$$

With this, we restate $(\mathcal{P})$ without the state variables

$$f_N^{\text{opt}} = \min_{\boldsymbol{U}(k)} f_N(\boldsymbol{U}(k)) \text{ s.t. } \boldsymbol{U}(k) \in \mathcal{U}_N. \tag{8}$$

It is now straightforward to see that we have been working with a degree-4 (quartic) polynomial over a ternary hypercube.

A. SDP relaxation of DTFCS-MPC optimization problem

With a change of variables, and the identity $\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} = \text{tr}(\boldsymbol{P}\boldsymbol{x}\boldsymbol{x}^\top)$, (7) is equivalent to

$$\begin{aligned}f_N^{\boldsymbol{\Theta}}(\boldsymbol{\Theta}(k)) = \lambda_T\sum_{\ell=1}^{N}&(T_{\text{e}}^* - \text{tr}(\boldsymbol{Q}_\ell(k)\boldsymbol{\Theta}(k)))^2\\ +(1-\lambda_T)\sum_{\ell=1}^{N}&\big(\Psi_s^{*2} - \text{tr}(\boldsymbol{W}_\ell(k)\boldsymbol{\Theta}(k))\big)^2\\ +\lambda_u\sum_{\ell=1}^{N}&\sum_{z\in\{\text{a},\text{b},\text{c}\}} \text{tr}\big(\boldsymbol{Z}_{z,\ell}(k)^\top \boldsymbol{Z}_{z,\ell}(k)\boldsymbol{\Theta}(k)\big),\end{aligned} \tag{9}$$

[3] In terms of complexity, SDPs are considered to be in PSPACE [38].

where $\mathbf{\Theta}(k) = \tilde{\boldsymbol{U}}(k)\tilde{\boldsymbol{U}}(k)^\top \in \{-1,\ 0,\ 1\}^{3N+1\times 3N+1}$. The variable $\mathbf{\Theta}(k)$ lies in the lifted space and the relation $\mathbf{\Theta}(k) = \tilde{\boldsymbol{U}}(k)\tilde{\boldsymbol{U}}(k)^\top$ captures its dependence to the original variable, so that $f_N^{\Theta}(\mathbf{\Theta}(k)) = f_N(\boldsymbol{U}(k))$. With (9), (8) can be reformulated as the following

$$\begin{aligned} f_N^{\mathrm{opt}} = &\min_{\mathbf{\Theta}(k)} \ f_N^{\mathbf{\Theta}}(\mathbf{\Theta}(k)) \\ &\ \text{s.t.} \quad \mathbf{\Theta}(k) = \tilde{\boldsymbol{U}}(k)\tilde{\boldsymbol{U}}(k)^\top, \\ &\qquad\quad \boldsymbol{U}(k) \in \mathcal{U}_N. \end{aligned} \tag{10}$$

The constraint $\mathbf{\Theta}(k) = \tilde{\boldsymbol{U}}(k)\tilde{\boldsymbol{U}}(k)^\top$ is equivalent to having that $\mathbf{\Theta}(k)$ is PSD (i.e., has a Cholesky decomposition), and is of rank 1. Reflecting on this observation, we revisit a relaxation method, first proposed and analyzed for the Max-Cut problem by the seminal works of [40], [41]. This method substitutes the constraints of type $\mathbf{\Theta}(k) = \tilde{\boldsymbol{U}}(k)\tilde{\boldsymbol{U}}(k)^\top$ with the PSD cone and a rank constraint. Subsequently, we can disregard the rank and integer constraints to obtain an SDP relaxation.

First, we obtain the equivalent reformulation:

$$\begin{aligned} f_N^{\mathrm{opt}} = &\min_{\mathbf{\Theta}(k)} \ f_N^{\mathbf{\Theta}}(\mathbf{\Theta}(k)) \\ &\ \text{s.t.} \quad \mathbf{\Theta}(k) \succeq 0,\ \Theta_{1,1}(k) = 1, \\ &\qquad\quad \mathbf{\Theta}(k) \in \{-1,\ 0,\ 1\}^{3N+1\times 3N+1}, \\ &\qquad\quad \mathrm{rank}(\mathbf{\Theta}(k)) = 1, \end{aligned} \tag{11}$$

where $\Theta_{1,1}(k) = 1$ imposes the augmented 1.[4] We now drop the nonconvex rank and integer constraints as follows:

$$\begin{aligned} f_N^{\mathrm{opt}} \geq &\min_{\mathbf{\Theta}(k)} \ f_N^{\mathbf{\Theta}}(\mathbf{\Theta}(k)) \\ &\ \text{s.t.} \quad \mathbf{\Theta}(k) \succeq 0,\ \Theta_{1,1}(k) = 1, \\ &\qquad\quad \mathbf{\Theta}(k) \in [-1,\ 1]^{3N+1\times 3N+1}. \end{aligned} \tag{$\mathcal{P}_{\mathrm{sdp}}$}$$

Let $\mathbf{\Theta}^{\mathrm{sdp}}(k)$ denote its optimal solution. As a remark, the problem above is not in the standard form of an SDP, as it involves a convex quadratic function of a trace of a matrix in (9). Since trace is a linear map, this problem remains convex, and it can be reformulated with a linear objective using an epigraph representation and the Lorentz cone.[5] SDPs are convex optimization problems that can be handled by a variety of off-the-shelf solvers. For case studies, we will use a first-order solver, SCS [43], to obtain a solution fast without stringent precision requirements. Some of the other solvers that can handle SDPs can be listed as [44]–[46].

After solving ($\mathcal{P}_{\mathrm{sdp}}$), and obtaining $\mathbf{\Theta}^{\mathrm{sdp}}(k)$ in the lifted space, we need to extract an input sequence candidate. Towards this, we provide the following four options:

1) *First column:* We take the first column of $\mathbf{\Theta}^{\mathrm{sdp}}(k)$ and round it, i.e., $\boldsymbol{U}^{\mathrm{sdp}}(k) = \mathrm{round}\left(\mathbf{\Theta}^{\mathrm{sdp}}_{2:3N,1}(k)\right)$.
2) *Diagonal:* We take the square-root of the diagonal, multiply it with the sign of the first column, and round the result, i.e., $\boldsymbol{U}^{\mathrm{sdp}}(k) = \mathrm{round}\left(\mathrm{sign}(\mathbf{\Theta}^{\mathrm{sdp}}_{2:3N,1}(k)) \odot \sqrt{\mathrm{diag}(\mathbf{\Theta}^{\mathrm{sdp}}_{2:3N,2:3N}(k))}\right)$.
3) *Maximal Eigenvector:* From the eigenvalue decomposition (EVD) of $\mathbf{\Theta}^{\mathrm{sdp}}(k)$ we round the eigenvector with the largest eigenvalue, i.e., $\boldsymbol{U}^{\mathrm{sdp}}(k) = \mathrm{round}\left(\sqrt{\lambda_1}\boldsymbol{v}_1\right)$, where λ_1 is the largest eigenvalue of $\mathbf{\Theta}^{\mathrm{sdp}}(k)$ and $\boldsymbol{v}_1$ the corresponding eigenvector.
4) *Weighted Eigenvectors:* From the EVD of $\mathbf{\Theta}^{\mathrm{sdp}}(k)$ we round the eigenvalue-weighted sum of the eigenvectors, i.e., $\boldsymbol{U}^{\mathrm{sdp}}(k) = \mathrm{round}\left(\sum_j \sqrt{\lambda_j}\boldsymbol{v}_j\right)$.

Our case studies will rely on the computationally-simplest first option, as no significant control performance differences have been observed during transients.[6]

Note that for this class of problems, in the literature, no theoretical approximation guarantee is known for the extracted candidates [31]. This remains open for future research. For quadratic optimization problems over the Boolean hypercube, for example, [32] provides an approximation guarantee by generalizing the analysis of the GW algorithm of [41]. However, convex relaxations are also known to perform well when the underlying problem exhibits approximate convexity. In fact, this observation strongly motivates our use of an SDP relaxation. The torque and flux magnitude tracking problem is known to have such approximate convexity, as originally presented in [33, §4.3]. This is evident in [33, Fig. 4.25] from the approximately elliptical cost contours of the objective in ($\mathcal{P}$). To abide by the page limitations, these observations from the literature are not repeated here.

Nevertheless, the term

$$f_N(\boldsymbol{U}^{\mathrm{sdp}}(k)) - f_N^{\mathbf{\Theta}}(\mathbf{\Theta}^{\mathrm{sdp}}(k)),$$

bounds the optimality gap, since

$$f_N(\boldsymbol{U}^{\mathrm{sdp}}(k)) \geq f_N^{\mathrm{opt}} \geq f_N^{\mathbf{\Theta}}(\mathbf{\Theta}^{\mathrm{sdp}}(k)).$$

Moreover, regardless of whether the situation is transient or steady-state, SDPs are convex programs that can be solved within a reasonable and consistent time frame.

Next, we present our proposed approach that exploits the advantages of both the SDP and the branch-and-bound.

B. *Proposed algorithm*

We propose solving the SDP relaxation in ($\mathcal{P}_{\mathrm{sdp}}$) in parallel to Algorithm 1. In the final step, we choose the best of the early-stopping solution from Algorithm 1 and the sequence extracted from the SDP solution in the lifted space. This procedure is formalized in Algorithm 2, where the *first column* extraction method is adopted. This proposal is constructed to harness the benefits of both the SDP and the branch-and-bound algorithm. It achieves optimality during the steady-state, given that the branch-and-bound concludes prior to hitting the node limit. The SDP relaxation is also solved fast consistently in parallel and provides a meaningful close-to-optimal input sequence even during transient events.

Remark. There exist alternatives for leveraging the SDP relaxations that we are not examining in this study. One can use the best of the sequence extracted from the SDP and the educated guess to warm-start the branch-and-bound. However, this would be computationally wasteful as the two would be

[4] $\mathrm{diag}(\mathbf{\Theta}(k)) \geq 0$ holds by definition for the PSD cone.

[5] The Lorentz cone is linearly isomorphic to the 2×2 PSD cone [42], and can be handled by SDP solvers.

[6] The EVD has the complexity $\mathcal{O}(N^3)$. In practice, it can be computed with very good precision via several iterations of the Power Method, each with the complexity $\mathcal{O}(N^2)$.

Algorithm 2 Branch-and-bound with the SDP relaxation

Function: branch_and_bound_with_SDP
Input: $\boldsymbol{x}(k-1), \boldsymbol{u}(k-1), \boldsymbol{U}^{\text{opt}}(k-1)$
Output: $\boldsymbol{U}^{\text{inc}}, n_{\text{p}}$
Start Parallel computation:
Thread 1 (Branch-and-bound)
$\boldsymbol{U}^{\text{ed}} \leftarrow [\boldsymbol{U}^{\text{opt}}_{4:3N}(k-1)^\top \boldsymbol{U}^{\text{opt}}_{3N-2:3N}(k-1)^\top]^\top$
$f^{\text{ed}} \leftarrow f_N(\boldsymbol{U}^{\text{ed}})$
$f_0 \leftarrow 0,\ \ell \leftarrow 1,\ n_{\text{p}} \leftarrow 0$
$\boldsymbol{U}^{\text{b\&b}} \leftarrow$ branch_and_bound$(f_0, f^{\text{ed}}, \ell, \boldsymbol{U}^{\text{ed}}, \boldsymbol{U}^{\text{ed}}, \boldsymbol{x}(k-1), \boldsymbol{u}(k-1), n_{\text{p}})$
End Thread 1
Thread 2 (SDP relaxation)
Solve $(\mathcal{P}_{\text{sdp}})$ for $\boldsymbol{\Theta}^{\text{sdp}}(k)$, $\boldsymbol{U}^{\text{sdp}} \leftarrow \text{round}(\boldsymbol{\Theta}^{\text{sdp}}_{2:3N,1}(k))$
End Thread 2
if $f_N(\boldsymbol{U}^{\text{b\&b}}) < f_N(\boldsymbol{U}^{\text{sdp}})$ **then**
  $\boldsymbol{U}^{\text{inc}} \leftarrow \boldsymbol{U}^{\text{b\&b}}$
else
  $\boldsymbol{U}^{\text{inc}} \leftarrow \boldsymbol{U}^{\text{sdp}}$
end if
return $\boldsymbol{U}^{\text{inc}}$

TABLE I: Physical parameters.

Rated values		Machine parameters [pu]	
Voltage	3300 V	Stator resistance	$R_s = 0.0108$
Current	356 A	Rotor resistance	$R_r = 0.0091$
Real power	1.587 MW	Stator leak. react.	$X_{ls} = 0.1493$
Apparent power	2 MVA	Rotor leak. react.	$X_{lr} = 0.1104$
Base frequency	50 Hz	Main reactance	$X_m = 2.3489$
Rotational speed	596 rpm	Number of pole pairs	$p = 5$
dc-link voltage	1.5937 pu		

run sequentially. Solving SDPs during the branch-and-bound iterations is another option, as the SDP could provide a lower bound on the future branch cost. However, this might not be realistic from a computational standpoint.

IV. Case Studies

First, we present a case study illustrating the variability in the efficiency of Algorithm 1, due to the initial guess and the operational conditions, when no node limits are enforced. Subsequently, we demonstrate through other case studies how Algorithm 1 could potentially fail to obtain a meaningful solution during transients. Algorithm 2 then effectively remedies these issues. The implementations of the algorithms are publicly available at https://github.com/lucahart/Torque-tracking.

Physical parameters are provided in Table I and they are taken from [47, Tab. 7.13]. We emphasize that this parameter set originates from an actual IM to ensure the practical relevance of the proposed methodology. The control parameters of Table II are chosen as follows. The torque weighting factor, $\lambda_{\text{T}} = 0.052$, is adopted from [33, §4.3]. The switching penalty $\lambda_{\text{u}} = 3.8 \times 10^{-3}$ is tuned to achieve a desired device switching frequency of $f_{\text{sw}} = 215\,\text{Hz}$. The sampling interval is $T_{\text{c}} = 25\,\mu\text{s}$, and the prediction horizon is $N = 5$. Our comparisons will be based on the steady-state behaviour and on the step-up of the torque reference T_{e}^*. The torque step-up will occur at $t = 0.705$s from $T_{\text{e}}^* = 0.2$ to $T_{\text{e}}^* = 1$.

TABLE II: Controller and simulation parameters.

Parameter name	Parameter symbol	Parameter values
torque weighting factor	λ_{T}	0.052
switching weight	λ_{u}	3.8×10^{-3}
controller sampling interval	T_{c}	$25\,\mu\text{s}$
horizon	N	5
upper bound on traversed nodes	$n_{\text{p,max}}$	500
simulation sampling interval	T_{s}	$0.5\,\mu\text{s}$

During all studies, the stator flux magnitude reference will be kept at a constant level $\Psi_{\text{s}}^* = 1$. The system is simulated with the time step of $T_{\text{s}} = 0.5\,\mu\text{s}$. A measurement noise of $\eta \sim \mathcal{U}(-2.5\times10^{-3},\ 2.5\times10^{-3})$ pu is imposed, where $\mathcal{U}(a,\ b)$ is the uniform distribution in the interval $[a,\ b]$.

The number of nodes traversed by Algorithm 1 is assumed to be a good indicator of the complexity of the branch-and-bound. In fact, we computed the Pearson correlation coefficient of $\text{Corr}(T_b, n_p) = 0.9989$ for the computational time T_b and the number of nodes traversed n_{p}. Considering this, as a node limit, $n_{\text{p,max}} = 500$ is imposed in our simulation environment. On the other hand, the SDP is solved via SCS [43] called through YALMIP [48], with an iteration limit of 120 and a solution tolerance of 10^{-4}. For the SDP under consideration, this limit influences the high-order precision, which is later rounded during the extraction process. As a remark, an assessment on whether these limits provide a fair comparison requires an implementation on an embedded platform, such as an FPGA. The goal of this paper is to showcase the capabilities of SDP relaxations for direct torque control while promoting the integration of tools from two research communities. As a further justification of this choice, computation limits tighter than the one we imposed on the branch-and-bound have already been demonstrated for the sphere-decoder based on the number of operations, see [24].

A. *Efficiency analysis for branch-and-bound*

This study shows how the node limit $n_{\text{p,max}} = 500$ could be significantly violated by Algorithm 1 if we do not impose it explicitly within the algorithm. Specifically, the efficiency of the branch-and-bound relies on two conditions: *(i)* the accuracy of the initial guess, *(ii)* the operational condition of the system. We study both in the following.

The impact of the first condition can be observed during the steady-state. We warm-start Algorithm 1 with an all-zero initial guess, an educated guess, and the optimal solution. We include the optimal solution here to specifically demonstrate how the effectiveness of the bounding decreases later during torque transients. Table III shows the resulting maximum number of nodes traversed by the branch-and-bound. During steady-state, observe that the educated guess performs rather similar to the optimal solution, and both stay well under the limit $n_{\text{p,max}} = 500$. The all-zero initial guess, on the other hand, can result in violating the node limit $n_{\text{p,max}} = 500$, making it unsuitable for Algorithm 1.

The impact of operational condition is apparent during a unit torque step. Table III shows that, especially in the first instance after the step command, the number of traversed nodes peaks. Regardless of the initial guess, the node limit is significantly

TABLE III: Number of nodes traversed for different initial guesses and situations. *Max-steady-state* refers to the maximum during steady-state. *Torque-step-instance* refers to the solution during the first instance of the step.

Initial solution	Max-steady-state	Torque-step-instance
Zero guess	1363	69950
Educated guess	361	59674
Optimal guess	358	35524

surpassed. This is because the inaccurate tracking during the step-up leads to much higher cost values. The branch-and-bound algorithm, having no information on the future cost, needs to traverse deep into the tree before deciding on pruning the sub-trees. As an interesting remark, the educated guess performs almost as bad as the all-zero initial guess during transients. The next two studies will demonstrate that the large number of traversed nodes is not merely for optimality verification, and early-stopping solutions can be arbitrarily poor.

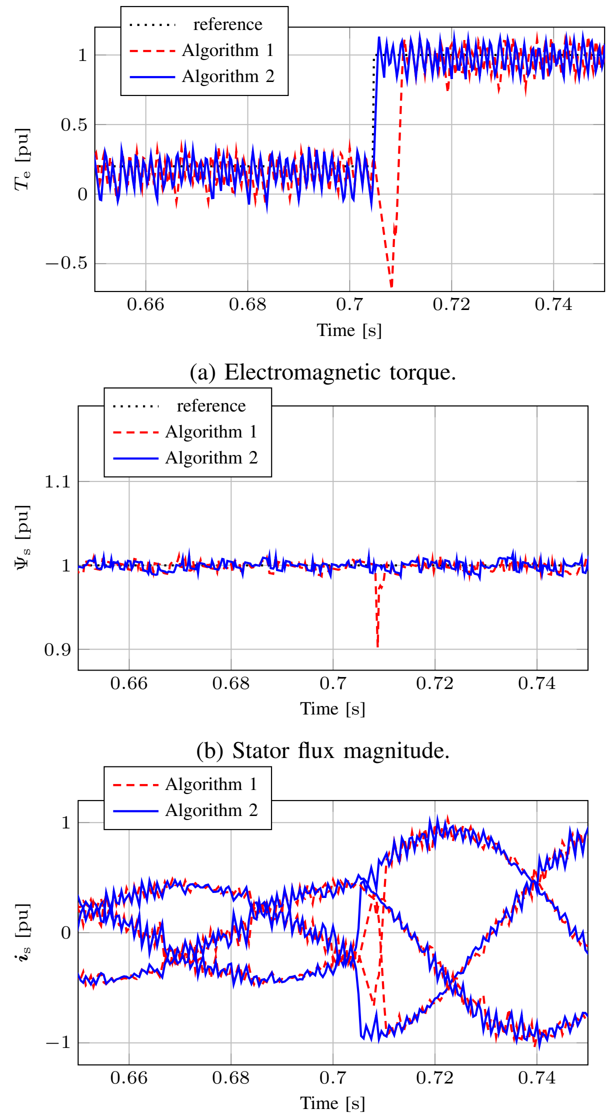
B. Performance improvement during torque steps

To further increase the challenge for Algorithm 2, we reduce the node limit of its parallel branch-and-bound algorithm to $n_{\mathrm{p,max}} = 250$, while maintaining $n_{\mathrm{p,max}} = 500$ for Algorithm 1. This adjustment also implicitly accounts for the additional computational load introduced by solving the SDP in Algorithm 2. To study an even more challenging scenario, Section IV-C will consider also a case where Algorithm 1 operates with an increased node limit of $n_{\mathrm{p,max}} = 2000$.

Figure 3a shows the torque trajectories of DTFCS-MPC with Algorithms 1 and 2. The input sequence from the node-limited Algorithm 1 cannot track the torque, whereas Algorithm 2 can find a close-to-optimal input sequence thanks to the SDP, leading to high dynamical performance. Figure 3b shows that Algorithm 1 creates large deviations in stator flux. All the aforementioned input sequence extraction methods gave the same result during the steady-state in terms of harmonic distortions thanks to the performance of branch-and-bound. Moreover, all these methods were capable of fixing the failures observed during torque transients due to node limit. SDP relaxations have been demonstrated to be consistently and reliably solved within an acceptable time frame. As mentioned before, we used SCS [43] called through YALMIP [48]. The average solver total time (includes both the setup and the solve stages of SCS) was $8.3\,\mathrm{ms}$ and the maximum total time was $10.4\,\mathrm{ms}$ for both the steady-state and the transient phases.[7] Considering the times reported in past literature [12, Fig. 5], one could obtain a significant improvement in execution times by an FPGA. Nonetheless, this observation needs to be verified for the SDPs on an embedded platform.

C. A comprehensive and randomized case study

We now compare Algorithm 2 with $n_{\mathrm{p,max}} = 250$ against Algorithm 1 across a sweep of node limits $n_{\mathrm{p,max}} \in \{250, 500, 750, 1000, 1500, 2000\}$. To further support our claims regarding the consistency of the SDP relaxation in



(a) Electromagnetic torque.

(b) Stator flux magnitude.

(c) Three-phase stator current measurements.

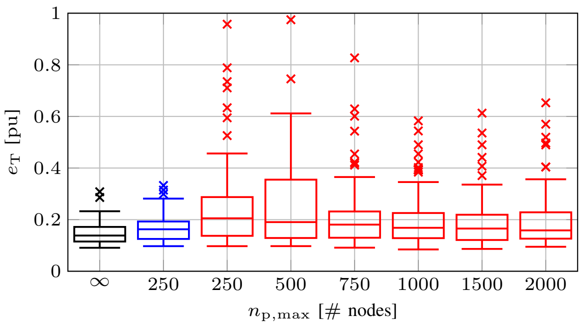
Fig. 3: Torque reference step-up simulation results. Depicted are DTFCS-MPC with solvers as in Algorithms 1 and 2.

TABLE IV: Key quantities for total tracking error ($\times 10^{-3}$)

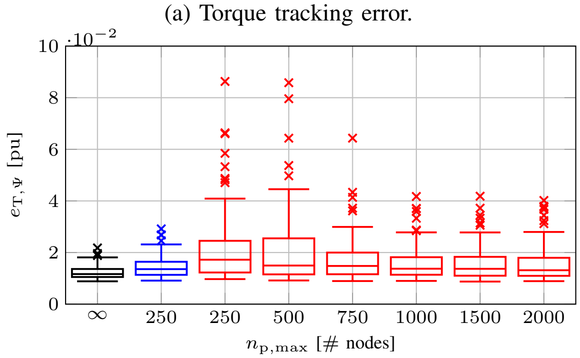
	∞ Alg. 1	250 Alg. 2	250 Alg. 1	500 Alg. 1	750 Alg. 1	1000 Alg. 1	1500 Alg. 1	2000 Alg. 1
Max	21.8	29.2	232	85.8	64.3	41.7	41.8	40.2
Quart. 3	13.6	16.5	24.6	25.5	20.0	18.9	18.4	18.2
Quart. 1	10.6	11.4	12.4	11.6	11.7	11.4	11.1	11.0
Mean	12.5	14.7	23.5	20.6	17.1	16.3	15.9	16.0

improving the node-limited branch-and-bound, we generate randomized examples. The setup consists of 10 distinct torque steps, each also triggered at 10 different time stamps, giving us 100 cases in total. The magnitudes of the steps are sampled from $0.5 \leq |\Delta T_{\mathrm{e}}| \leq 1.5$, while meeting the torque limits. The evaluation is based on closed-loop tracking performance after each step. Let Δk denote the number of samples within the first $5\,\mathrm{ms}$ after a step, and let k_0 be the torque step time. Define two tracking errors: $e_{\mathrm{T}} = \frac{1}{\Delta k}\sum_{k=k_0}^{k_0+\Delta k}(e_{\mathrm{T}}(k))^2$ and $e_{\mathrm{T},\Psi} = \frac{1}{\Delta k}\sum_{k=k_0}^{k_0+\Delta k}\lambda_{\mathrm{T}}(e_{\mathrm{T}}(k))^2 + (1-\lambda_{\mathrm{T}})(e_{\Psi}(k))^2$.

[7] Solved via MATLAB on a computer equipped with $32\,\mathrm{GB}$ RAM and a $2.5\,\mathrm{GHz}$ Intel i7 processor.



(a) Torque tracking error.



(b) Total tracking error.

Fig. 4: Box plots. The black is the optimal solution. The blue is Algorithm 2. The red ones are Algorithm 1.

Figure 4a presents box plots of the torque tracking error e_{T}. In terms of both mean and quartiles simultaneously, Algorithm 2 with $n_{\mathrm{p,max}} = 250$ outperforms Algorithm 1 up to $n_{\mathrm{p,max}} = 1500$. Moreover, Algorithm 2 exhibits significantly fewer outliers than Algorithm 1, even up to $n_{\mathrm{p,max}} = 2000$. Figure 4b shows the total tracking error $e_{\mathrm{T},\Psi}$, and Table IV provides the key statistical quantities. Here, the mean and quartiles of Algorithm 2 remain superior to those of Algorithm 1 up to $n_{\mathrm{p,max}} = 750$. Algorithm 2 continues to show fewer outliers across all cases. These results demonstrate that Algorithm 2 consistently avoids outliers, such as the failure case of Algorithm 1 illustrated in Figure 3. We conclude that Algorithm 2 is an effective strategy for complementing a node-limited branch-and-bound during large transients. Given its proximity to the optimal solution, this also provides statistical evidence of the aforementioned approximate convexity.

V. Conclusion

We formulated the SDP relaxation of DTFCS-MPC, and proposed solving it in parallel to a node-limited branch-and-bound designed for the original problem. Case studies showcased that a solution extracted from the SDP can be advantageous during torque transients. Future work could solve the SDP relaxation of DTFCS-MPC on an FPGA, and provide a theoretical approximation guarantee on the extracted input sequences.

References

[1] P. Karamanakos, E. Liegmann, T. Geyer, and R. Kennel, "Model predictive control of power electronic systems: Methods, results, and challenges," *IEEE Open J. of Ind. Appl.*, vol. 1, pp. 95–114, 2020.

[2] J. Rodriguez, C. Garcia, A. Mora, F. Flores-Bahamonde, P. Acuna, M. Novak, Y. Zhang, L. Tarisciotti, S. A. Davari, Z. Zhang *et al.*, "Latest advances of model predictive control in electrical drives—Part I: Basic concepts and advanced strategies," *IEEE Trans. on Power Elec.*, vol. 37, no. 4, pp. 3927–3942, 2021.

[3] E. Zafra, S. Vazquez, T. Geyer, R. P. Aguilera, and L. G. Franquelo, "Long prediction horizon FCS-MPC for power converters and drives," *IEEE Open J. of the Ind. Elec. Society*, 2023.

[4] S. Muller, U. Ammann, and S. Rees, "New modulation strategy for a matrix converter with a very small mains filter," in *IEEE PESC*, vol. 3, 2003, pp. 1275–1280.

[5] J. Rodrıguez, J. Pontt, C. Silva, M. Salgado, S. Rees, U. Ammann, P. Lezana, R. Huerta, and P. Cortés, "Predictive control of three-phase inverter," *Elec. letters*, vol. 40, no. 9, 2004.

[6] 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. Elec. Mag.*, vol. 8, no. 1, pp. 32–43, 2014.

[7] T. Geyer and D. E. Quevedo, "Performance of multistep finite control set model predictive control for power electronics," *IEEE Trans. on Power Elec.*, vol. 30, no. 3, pp. 1633–1644, 2014.

[8] ——, "Multistep finite control set model predictive control for power electronics," *IEEE Trans. on Power Elec.*, vol. 29, no. 12, pp. 6836–6846, 2014.

[9] E. Liegmann, P. Karamanakos, T. Geyer, T. Mouton, and R. Kennel, "Long-horizon direct model predictive control with active balancing of the neutral point potential," in *IEEE PRECEDE*, 2017, pp. 89–94.

[10] L. M. Hartmann, O. Karaca, T. Dorfling, and T. Geyer, "Switching frequency limitation with finite control set model predictive control via slack variables," *IEEE Trans. on Contr. Sys. Tech.*, vol. 33, no. 3, pp. 1125 – 1133, 2025.

[11] K. Hasse, "Zum dynamischen verhalten der asynchronmaschine bei betrieb mit variabler standerfrequenz und standerspannung," *ETZ-A Bd.*, vol. 89, p. 77, 1968.

[12] B. Stellato, T. Geyer, and P. J. Goulart, "High-speed finite control set model predictive control for power electronics," *IEEE Trans. on Power Elec.*, vol. 32, no. 5, pp. 4007–4020, 2017.

[13] P. Acuna, C. A. Rojas, R. Baidya, R. P. Aguilera, and J. E. Fletcher, "On the impact of transients on multistep model predictive control for medium-voltage drives," *IEEE Trans. on Power Elec.*, vol. 34, no. 9, pp. 8342–8355, 2019.

[14] E. Liegmann, P. Karamanakos, and R. Kennel, "Real-time implementation of long-horizon direct model predictive control on an embedded system," *IEEE Open J. of Ind. Appl.*, vol. 3, pp. 1–12, 2021.

[15] T. Laczynski and A. Mertens, "Predictive stator current control for medium voltage drives with lc filters," *IEEE Trans. on Power Elec.*, vol. 24, no. 11, pp. 2427–2435, 2009.

[16] R. Vargas, P. Cortés, U. Ammann, J. Rodríguez, and J. Pontt, "Predictive control of a three-phase neutral-point-clamped inverter," *IEEE Trans. on Ind. Elec.*, vol. 54, no. 5, pp. 2697–2705, 2007.

[17] I. Takahashi and T. Noguchi, "A new quick-response and high-efficiency control strategy of an induction motor," *IEEE Trans. on Ind. Appl.*, no. 5, pp. 820–827, 1986.

[18] D. Casadei, F. Profumo, G. Serra, and A. Tani, "Foc and dtc: two viable schemes for induction motors torque control," *IEEE Trans. on Power Elec.*, vol. 17, no. 5, pp. 779–787, 2002.

[19] G. S. Buja and M. P. Kazmierkowski, "Direct torque control of pwm inverter-fed ac motors-a survey," *IEEE Trans. on Ind. Elec.*, vol. 51, no. 4, pp. 744–757, 2004.

[20] H. Miranda, P. Cortés, J. I. Yuz, and J. Rodríguez, "Predictive torque control of induction machines based on state-space models," *IEEE Trans. on Ind. Elec.*, vol. 56, no. 6, pp. 1916–1924, 2009.

[21] T. Geyer, G. Papafotiou, and M. Morari, "Model predictive direct torque control—part i: Concept, algorithm, and analysis," *IEEE Trans. on Ind. Elec.*, vol. 56, no. 6, pp. 1894–1905, 2008.

[22] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Oper. Res.*, vol. 14, no. 4, pp. 699–719, 1966.

[23] T. Geyer, "Computationally efficient model predictive direct torque control," *IEEE Trans. on Power Elec.*, vol. 26, no. 10, pp. 2804–2816, 2011.

[24] P. Karamanakos, T. Geyer, and R. Kennel, "Suboptimal search strategies with bounded computational complexity to solve long-horizon direct model predictive control problems," in *ECCE*. IEEE, 2015, pp. 334–341.

[25] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM J. on Opt.*, vol. 11, no. 3, pp. 796–817, 2001.

[26] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Math. Prog.*, vol. 96, pp. 293–320, 2003.

[27] L. Wachter, O. Karaca, G. Darivianakis, and T. Charalambous, "A convex relaxation approach for the optimized pulse pattern problem," in *ECC*. IEEE, 2021, pp. 2213–2218.

[28] P. A. Parrilo and S. Lall, "Semidefinite programming relaxations and algebraic optimization in control," *Eur. J. of Contr.*, vol. 9, no. 2-3, pp. 307–321, 2003.

[29] D. Axehill, L. Vandenberghe, and A. Hansson, "Convex relaxations for mixed integer predictive control," *Automatica*, vol. 46, no. 9, pp. 1540–1545, 2010.

[30] N. Li, K. Han, I. Kolmanovsky, and A. Girard, "Coordinated receding-horizon control of battery electric vehicle speed and gearshift using relaxed mixed-integer nonlinear programming," *IEEE Trans. on Contr. Sys. Tech.*, vol. 30, no. 4, pp. 1473–1483, 2021.

[31] D. Henrion and J.-B. Lasserre, "Detecting global optimality and extracting solutions in gloptipoly," in *Positive polynomials in control*. Springer, 2005, pp. 293–310.

[32] Y. Nesterov, "Semidefinite relaxation and nonconvex quadratic optimization," *Opt. Methods and Software*, vol. 9, no. 1-3, pp. 141–160, 1998.

[33] T. Geyer, *Model Predictive Control of High Power Converters and Industrial Drives*. Wiley, November 2016.

[34] M. Laurent, "A comparison of the sherali-adams, lovász-schrijver, and lasserre relaxations for 0–1 programming," *Math. of Oper. Res.*, vol. 28, no. 3, pp. 470–496, 2003.

[35] S.-K. Sul, *Control of electric machine drive systems*. John Wiley & Sons, 2011.

[36] P. Karamanakos and T. Geyer, "Guidelines for the design of finite control set model predictive controllers," *IEEE Trans. on Power Elec.*, vol. 35, no. 7, pp. 7434–7450, 2019.

[37] J. Hastad, "Clique is hard to approximate within $n^{1-\epsilon}$," in *Proceedings of 37th Conf. on Foundations of Comp. Sci.* IEEE, 1996, pp. 627–636.

[38] S. P. Tarasov and M. N. Vyalyi, "Semidefinite programming and arithmetic circuit evaluation," *Discrete Applied Mathematics*, vol. 156, no. 11, pp. 2070–2078, 2008.

[39] L. M. Hartmann, O. Karaca, T. Dorfling, T. Geyer, and A. Kurpisz, "On the SDP relaxation of direct torque finite control set model predictive control," *arXiv:2412.11666 online appendix*, 2024.

[40] C. Delorme and S. Poljak, "Laplacian eigenvalues and the maximum cut problem," *Math. Prog.*, vol. 62, pp. 557–574, 1993.

[41] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. of the ACM*, vol. 42, no. 6, pp. 1115–1145, 1995.

[42] A. Ben-Tal and A. Nemirovski, *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[43] B. O'donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *J. of Opt. Theory and Appl.*, vol. 169, pp. 1042–1068, 2016.

[44] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Opt. Methods and Software*, vol. 11, no. 1-4, pp. 625–653, 1999.

[45] P. J. Goulart and Y. Chen, "Clarabel: An interior-point solver for conic programs with quadratic objectives," *arXiv:2405.12762*, 2024.

[46] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 10.1.*, 2024.

[47] T. Geyer, "Low complexity model predictive control in power electronics and power systems," in *Diss. ETH No. 15953*, 2005.

[48] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *ICRA*. IEEE, 2004, pp. 284–289.