Computationally Efficient Model Predictive Direct Torque Control

Tobias Geyer, *Member, IEEE*

***Abstract*—For medium-voltage drives Model Predictive Direct Torque Control (MPDTC) significantly reduces the switching losses and/or the harmonic distortions of the torque and stator currents, when compared to standard schemes such as direct torque control or pulse width modulation. Extending the prediction horizon in MPDTC further improves the performance. At the same time the computational burden is greatly increased due to the combinatorial explosion of the number of admissible switching sequences. Adopting techniques from mathematical programming, most notably branch and bound, the number of switching sequences explored can be significantly reduced by discarding suboptimal sequences. This reduces the computation time by an order of magnitude thus enabling MPDTC with long prediction horizons to be executed on today's available hardware.**

***Index Terms*—Model predictive control, electrical drive control, branch and bound, medium-voltage drive, medium-voltage converter**

I. Introduction

Direct Torque Control (DTC) is ABB's method of choice for controlling the motor's torque and flux in medium voltage drive applications, with a typical product example being the ACS 6000 drive [1]. Over the past decade DTC has demonstrated a high degree of reliability, robustness and a superior performance during transients. The switching losses, however, which represent a major part of the overall losses of the drive, can be substantial in DTC.

As shown in [2], [3], the switching losses can be significantly reduced through Model Predictive Direct Torque Control (MPDTC), which is based on the concept of Model Predictive Control (MPC) [4], [5]. The initial version of MPDTC is available in two forms with switching horizons of one or two steps [2], [3] and prediction horizons in the range of a few dozen steps. MPDTC was generalized in [6] allowing for an extended switching horizon, which is composed of multiple hinges (groups of switch transitions) linked by several extrapolation or extension segments. This enables prediction horizons of 100 steps and more, despite relatively short switching horizons.

The computational complexity of MPDTC is proportional to the number of admissible switching transitions at every time-step to the power of the number of switching events considered in the prediction horizon. The former, the number of switching transitions per time-step, is determined by the inverter topology – most prominently by the number of voltage levels available. The latter, the number of switching events, is set by the switching horizon. Long switching horizons greatly boost the performance of MPDTC, in the sense that the switching losses and/or the current or torque Total Harmonic Distortions (THDs) are further lowered [7]. However, long switching horizons lead to a combinatorial explosion of the number of admissible switching sequences to be explored. So far, to find the optimal switching sequence, all admissible sequences were investigated by the MPDTC algorithm using full enumeration. This brute-force concept becomes computationally very expensive and thus prohibitive for long switching horizons. As a result, it has only been possible to implement and run MPDTC on a control hardware platform when restricting oneself to very short switching horizons [8].

This shortcoming motivates this paper, which presents techniques to drastically reduce the number of switching sequences explored and thus to lessen MPDTC's computational burden. The first technique, branch and bound, uses upper and lower bounds on the objective function to discard large parts of the search tree [9]. As a result, the optimal solution is found quicker and the *average* number of computations is reduced. To limit the *maximal* number of computations, the optimization procedure can be stopped if the number of computational steps exceeds a certain threshold. Despite leading to potentially suboptimal results, if the threshold is chosen carefully, the impact on the performance is small. Alternatively, one may stop if the current best solution is sufficiently close to the optimum.

Simulation results indicate that these techniques reduce the computation time by an order of magnitude when compared to full enumeration. MPDTC with long switching horizons is thus expected to become implementable on today's available control hardware so as to take full advantage of MPDTC's performance benefits.

The paper is structured as follows. After revisiting the drive control problem in Sect. II, Sect. III briefly recapitulates the key concepts of MPDTC, introduces the notion of the search tree and presents a slightly modified MPDTC algorithm that is solved using full enumeration. Sect. IV proposes a computationally efficient version of MPDTC based on branch and bound and an upper bound on the number of computations. Computational results are presented in Sect. V before conclusions are drawn in the final Sect. VI.

II. Control Problem

The DTC control objectives are to keep the so called output variables, namely the electromagnetic torque, the length (or magnitude) of the stator flux vector and the neutral point potential(s), within given hysteresis bounds. In MPDTC, these objectives are inherited from DTC. In addition, the inverter losses are to be minimized. An indirect way of achieving this is

T. Geyer is currently with the Department of Electrical and Computer Engineering, The University of Auckland, New Zealand; e-mail: t.geyer@ieee.org

to minimize the (short-term) average switching frequency [2], [3] or to directly minimize the switching losses [6], [10].

The MPC controller is endowed with a discrete-time model of the drive that enables it to anticipate the impact of its decisions. The control objectives are mapped into an objective function that yields a scalar cost (here the short-term switching losses) that is to be minimized. At every time-step, the controller computes a sequence of switch positions over a certain time-interval, the prediction horizon, that entails the minimal switching losses over this interval. Out of this sequence, only the first gating signal is applied at the current sampling instant, and the optimization step is repeated with new measurements at the next sampling instant thus providing feedback.

Writing the above control problem as a closed-form optimization problem leads to

$$J^*(x(k)) = \min_{U(k)} \frac{1}{N_p} \sum_{\ell=k}^{k+N_p-1} E_{\text{sw}}(x(\ell), u(\ell), u(\ell-1)) \quad \text{(1a)}$$

$$\text{s.t. } x(\ell+1) = Ax(\ell) + Bu(\ell) \quad \text{(1b)}$$

$$y(\ell) = g(x(\ell)) \quad \text{(1c)}$$

$$y(\ell) \in \mathcal{Y} \quad \text{(1d)}$$

$$u(\ell) \in \mathcal{U}\,, \quad \max|\Delta u(\ell)| \leq 1 \quad \text{(1e)}$$

$$\forall \ell = k, \ldots, k+N_p-1\,, \quad \text{(1f)}$$

with $J^*(x(k))$ denoting the minimum of the objective function J as a function of the state vector $x(k)$ at the current time-instant k. Often, the stator and rotor flux vectors represented in the $\alpha\beta$ reference frame are used as state vector x along with the neutral point potential(s). The motor speed is assumed to be constant within the prediction horizon and is thus not part of the state vector but rather a parameter of the model (1b). The sequence of control inputs $U(k) = [u(k), \ldots, u(k+N_p-1)]$ over the prediction horizon N_p represents the sequence of inverter switch positions the controller has to decide upon. The objective function represents the sum of the switching losses over the prediction horizon divided by the horizon length — it thus approximates the short-term average switching losses. Note that the instantaneous switching (energy) loss E_{sw} at time-instant ℓ is a function of the stator current $i_s(\ell)$, which in turn linearly depends on the state vector $x(\ell)$. The switching loss $E_{\text{sw}}(\ell)$ also depends on the inverter switching transition at time-step ℓ. The latter can be deduced from $u(\ell)$ and $u(\ell-1)$. An indirect (and less effective) way of minimizing the switching losses is to minimize the number of commutations, i.e. the device switching frequency.

The objective function is minimized subject to the dynamical evolution of the drive represented in state-space form with the matrices A and B, which are of appropriate form [2], [11]. The drive's output vector y represents the torque, stator flux magnitude and neutral point potential(s), which are to be kept within their respective bounds given by the set $\mathcal{Y}$. The constraint (1e) limits the control input u to the integer values $\mathcal{U}$ available for the specific inverter topology[1]. Switching in a phase by more than one step up or down is typically not allowed and can be inhibited by the second constraint in (1e) with $\Delta u(\ell) = u(\ell) - u(\ell-1)$. These constraints have to be met at every time-step within the prediction horizon.

[1]For a three-level inverter we have $\mathcal{U} = \{-1, 0, 1\}^3$ and $\mathcal{U} = \{-2, -1, 0, 1, 2\}^3$ for a five-level inverter.

III. MPDTC with Full Enumeration

To solve the closed-form optimization problem (1) is challenging from a computationally point of view even for prediction horizons of modest length. Solving it for reasonably long horizons appears to be impossible[2]. Since this is a combinatorial optimization problem, it is well-known that in the worst case all switching sequences need to be enumerated and evaluated to find the optimum.

A. The Concept of the Switching Horizon

One attractive solution is to consider switching transitions only when the outputs y are close to their respective bounds $\mathcal{Y}$, i.e. when switching is imminently required to keep the outputs within their bounds. When the outputs are well within their bounds the switch positions are frozen and switching is not considered. This is in line with the control objective in (1) and greatly reduces the number of switching sequences to be considered and thus the computational burden.

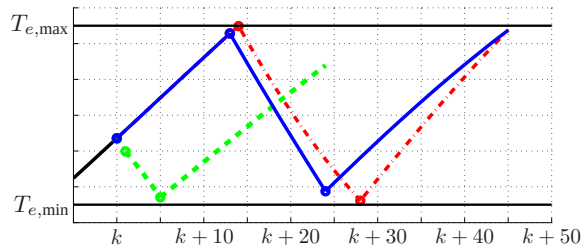
To achieve this, three key concepts were introduced in [2], [3], [6] that characterize Model Predictive Direct Torque Control (MPDTC).

1) The formulation of the optimization problem in an *open form*. For every admissible switching sequence the corresponding output trajectories are computed forward in time.
2) Between the switching events the output trajectories are computed using the model (1b) and (1c), to which we refer as an *extension* step, or they are extrapolated in an approximate manner, which is a so called *extrapolation* step. Typically, quadratic extrapolation is used, even though linear extrapolation is often sufficiently accurate.
3) The set of admissible switching sequences is controlled by the so called *switching horizon*, which is composed of the elements 'S' and 'E' that stand for 'switch' and 'extrapolate' (or more generally 'extend'), respectively.

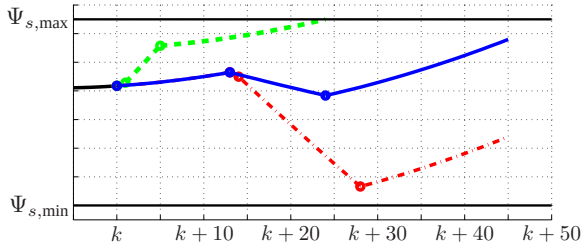
Is important to distinguish between the *switching horizon* (the number of time-steps within the horizon when switching transitions are considered, i.e. the degrees of freedom) and the *prediction horizon* (the number of time-steps MPDTC looks into the future). Between the switching instants the switch positions are frozen and the drive behavior is extrapolated until a hysteresis bound is hit. The concept of extrapolation gives rise to long prediction horizons (typically 50 to 100 time-steps), while the switching horizon is very short (usually one to four switching events are considered).

Example 1: As an example for a switching horizon consider 'SSESE', which stands for switching at time-steps k and $k+1$ and subsequently extending the trajectories until one or more output trajectories cease to be feasible (i.e. within the bounds) and/or pointing in the proper direction. Assume this happens

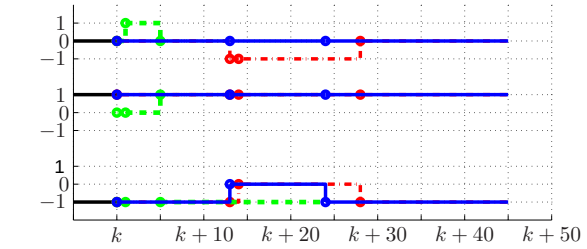
[2]Using a three-level inverter as an example, for a given switch position $u(\ell)$, the number of admissible future switch positions $u(\ell+1)$ is on average 12 and thus less than 27 due to (1e). Nevertheless, for $N_p = 75$ for example, the number of possible switching sequences U amounts to $12^{N_p} = 10^{80}$, which is equal to the estimated number of atoms in the observable universe.



(a) Predicted torque trajectories



(b) Predicted stator flux trajectories



(c) Predicted switching sequences

Fig. 1: Three candidate switching sequences for the switching horizon 'eSSESE' with the associated torque and stator flux trajectories between their respective upper and lower bounds. The neutral point potential is not shown. The prediction horizon here is $N_p = 45$ time-steps

at time-step $k+\ell$ thus triggering the third switching event that is followed by another extension step. We use the task 'e' to add an optional extension leg to the switching horizon. Using 'eSSESE' as an example, three candidate switching sequences are depicted in Fig. 1 along with their output trajectories.

B. Search Tree

The switching horizon induces a search tree as shown in Fig. 2. Each node in the search tree is specified by the 6-tuple $\{U, X, Y, E, N, A\}$ defined as follows.

- A switching sequence U is a sequence of three-phase switch positions u of length N starting at the current time-step k, i.e. $U = [u(k), \ldots, u(k+N-1)]$. An *admissible* switching sequence meets the constraint (1e) at every time-step, as well as other constraints, such as additional switching constraints induced by the inverter topology. Only admissible switching sequences are considered in MPDTC. A *candidate* sequence yields output trajectories that are at every time-step either feasible, or point in the proper direction. Feasibility means that the output variables lie within their corresponding bounds; pointing in the proper direction refers to the case in which an output variable is not necessarily feasible, but the degree of the bounds' violations decreases at every time-step. These conditions must hold componentwise for all output variables[3].
- Associated with a switching sequence is the state sequence $X = [x(k+1), \ldots, x(k+N)]$, which is a sequence of state vectors x that fully describes the evolution of the drive from the initial state $x(k)$ onwards when applying the input sequence U. The state vector typically encompasses the four components of the machine fluxes as well as the inverter's neutral point potential(s).
- Similarly, the evolution of the drive's outputs is described by the output sequence $Y = [y(k+1), \ldots, y(k+N)]$, where y is composed of the torque, the stator flux magnitude and the neutral point potential(s).
- The switching (energy) losses $E = \sum_{\ell=k}^{k+N-1} e(\ell)$ are the sum of the individual switching losses $e(\ell)$ in the switching sequence U. Their unit is Ws. Dividing them by a time interval such as the length of the prediction horizon, leads to the switching (power) losses P_{sw} with the unit W.
- N denotes the lengths of the predicted sequences U, X and Y. It can be interpreted as the resulting prediction horizon of variable length that is induced by the switching horizon.
- A denotes the sequence of actions to be performed on the node with the elements of A being in {'S', 'E'}.

It follows that there is a direct correspondence between switching sequences and nodes. Thus both terms will be used interchangeably in the sequel. Nodes either refer to incomplete or to complete solutions (switching sequences) of the optimization problem. More specifically, we distinguish between the following nodes.

- The *root node* is the initial node at time-step k. It is initialized with $\{\emptyset, \emptyset, \emptyset, 0, 0, \text{'SSESE'}\}$, assuming the switching horizon 'SSESE' and $\emptyset$ denoting an empty sequence. In Fig. 2 it is depicted as a (blue) oval.
- *Bud nodes* are *incomplete candidate* switching sequences with actions left that induce child nodes. The corresponding output sequences fulfill the candidacy requirement (so far). They are also depicted as a (blue) ovals.
- There are two types of *leaf nodes*. (i) *Complete candidate* switching sequence that have been fully computed with no actions remaining and candidacy fulfilled at every time-step. They are shown as a (green) stars. (ii) *Non-candidate* switching sequences, which are not further considered, and denoted by an inverted (red) T.

Pairs of nodes connected by switching transitions are shown as thin (black) lines, while extrapolation or extension steps are depicted as thick (blue) lines.

C. MPDTC Algorithm with Full Enumeration

In its basic form stated above, the MPDTC algorithm enumerates all admissible switching sequences that are candidate sequences and computes their corresponding output

[3]As an example, consider the case where the torque is feasible, the stator flux points in the proper direction and the neutral point potential is feasible.

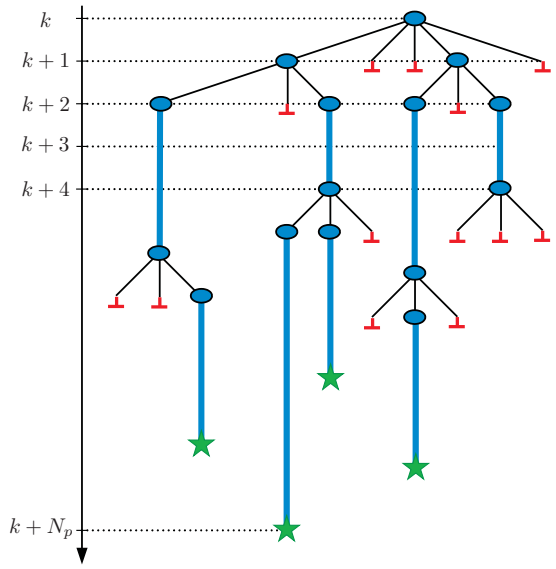


Fig. 2: Example of a search tree induced by the switching horizon 'SSESE' with (blue) ovals denoting the root and bud nodes, (green) stars being complete candidate switching sequences, and inverted (red) Ts marking non-candidate switching sequences. Switching transitions 'S' are shown as thin (black) lines, while extrapolation / extension steps 'E' are thick (blue) lines. The discrete-time axis is shown on the left along with the prediction horizon N_p

trajectories and their cost. Hence all nodes in the search tree are visited that belong to candidate switching sequences. Specifically, at time-step k with the sampling interval T_s, the MPDTC algorithm computes $u(k)$ according to the following procedure, which was presented in [6] and is slightly modified here to facilitate the addition of branch and bound techniques in the next section.

1) Initialize the root node and push it onto the stack.
2a) Take the top node i with a non-empty A_i from the stack.
2b) Read out the first element from A_i and remove it. For 'S', branch on all admissible switching transitions and add the (incremental) switching losses $E_i = E_i + e(\ell)$ caused by switching at the time-step ℓ. For 'E', extend the trajectories either by extrapolation as detailed in [2] or by using the internal controller model [6].
2c) Keep only the nodes that are candidates and push them onto the stack.
2d) Stop if there are no more nodes with non-empty A_i.
3) Compute for each (candidate) leaf node $i \in \mathcal{I}$, with $\mathcal{I}$ being an index set, the associated cost $c_i = E_i/(N_i T_s)$, which targets the switching losses.
4) Choose the leaf node $i = \arg\min_{i\in\mathcal{I}} c_i$ with the minimal cost and read out the associated switching sequence $U^* = U_i(k)$.
5) Apply (only) the first switch position $u(k) = u^*$ of this sequence and execute the above procedure again at the next time-step $k+1$.

If the switching frequency is to be minimized, replace E_i by the number of commutations S_i, add the incremental number of commutations in Step 2b when switching, and use in Step 3 the cost $c_i = S_i/(N_i T_s)$, which approximates the average switching frequency over the length of the prediction horizon.

IV. Computationally Efficient MPDTC

The problem at hand is to devise a modified algorithm that is computationally efficient thus allowing the implementation of MPDTC with long switching horizons on today's available computational hardware. This implies that the worst-case computational effort has to be reduced by at least an order of magnitude. This can be achieved by using tailored optimization techniques that reduce the number of nodes visited in the search tree. These techniques include the so called branch and bound methodology that is used in mathematical programming to solve combinatorial and (mixed) integer programs [9], [12].

A. Bounds on the Cost

First of all, note that the evolution of the cost associated with a switching sequence over time is neither smooth nor monotonically increasing or decreasing. Instead, at time-instants when switching transitions occur the cost is increased in a step-like fashion. As the sequence is extended the cost decreases smoothly.

Example 2: As an example consider the switching horizon 'SESE' and the cost evolution of the switching sequence 1 over time, which is given by the straight (green) line in Fig. 3. The initial cost is zero. At time-steps k and $k+7$ switching transitions occur that carry the switching energy losses e_1 and e_2, respectively. Between the switching events, the trajectories are extended, which reduces the cost (switching power losses) by distributing the switching energy losses over a longer time-interval. In this example, sequence 1a is an incomplete candidate switching sequence with the actions 'SE' remaining. At $k+7$ more than one admissible switching transition is feasible. Another transition with the energy losses e_3 leads to the dashed (blue) sequence 2. Sequences 1 and 2 are complete candidate switching sequences with no actions remaining, whose first parts coincide with sequence 1a.

Before proceeding, some terminology has to be introduced. In this, the index i refers to the i-th switching sequence (node).

- $c_i = E_i/(N_i T_s)$ is the cost associated with a (complete) candidate switching sequence, where E_i is the sum of the switching energy losses.
- c^* is the optimal (minimal) cost of the complete candidate switching sequences.
- $\bar{c}$ denotes the incumbent minimal cost, i.e. the smallest cost found so far for all complete candidate switching sequences. This cost constitutes an upper bound on the optimal cost to be found, i.e. $\bar{c} \geq c^*$.
- N_{max} is an upper bound to the (maximal) length of the prediction horizon, i.e. it is assumed that $N_i \leq N_{\text{max}}$ for all i.
- $\underline{c}_i = E_i/(N_{\text{max}} T_s)$ is a lower bound on the cost of the i-th incomplete switching sequence, where E_i is the sum of the switching energy losses incurred so far for this sequence[4].
- $\underline{c}$ refers to the minimum of all lower bounds $\underline{c}_i$. It holds that $\underline{c} \leq c^*$.

[4]Since E_i increases monotonically as the i-th switching sequence is extended and $N_i \leq N_{\text{max}}$ by definition, it holds that $\underline{c}_i \leq c_i$, i.e. $\underline{c}_i$ always underestimates the cost c_i.

B. The Concept of Branch and Bound

The concept of branch and bound is introduced in an intuitively accessible way by considering again Example 2.

Example 3: In Fig. 3 the cost associated with the complete candidate sequence 1 is $c_1 = (e_1+e_2)/(N_1T_s)$, with $N_1 = 12$ time-steps. The incumbent minimal cost is $\bar{c} = c_1$. Having computed the second switching transition at $k + 7$ with the energy losses e_3, one can try to find a proof *before* extending sequence 2 that this sequence, when completed, will only lead to a suboptimal solution that is inferior to the incumbent optimum. This proof can be found by computing the lower bound on the cost for sequence 2, which is given by $\underline{c}_2 = (e_1 + e_3)/(N_{\max}T_s)$. If $\underline{c}_2$ is equal to or exceeds $\bar{c}$ the remainder of this sequence can be discarded and removed from the search tree. If this is not the case, however, the sequence should be further considered and in this case extended. The same applies to the dash-dotted (red) sequence 3. Having computed the first switching transition with the losses e_4, the whole subtree starting from this node can be discarded if $\underline{c}_4 = e_4/(N_{\max}T_s) \geq \bar{c}$.

More formally, the branch and bound algorithm tailored to the MPDTC problem setup is as follows. Compute the switching sequences and the associated output trajectories and costs iteratively as the tree is explored from its root node to the terminal nodes (leaves). Assume that the bud node i, which corresponds to the incomplete candidate switching sequence U_i, has the minimum cost incurred so far. Discard the bud node and prune the attached unexplored part of the tree if $\underline{c}_i \geq \bar{c}$. If a candidate switching sequence is completed compute its cost c_i and update the incumbent minimal cost if required by setting $\bar{c} = \min(\bar{c}, c_i)$. The algorithm summarized in Sect. III-C can be easily augmented by the branch and bound methodology as will be shown in Sect. IV-D.

Example 4: Consider one instance of the combinatorial optimization problem, arising for example from a three-level inverter and a switching horizon of 'eSSESE'. The induced search tree contains 730 nodes. Using full enumeration, all 730 nodes are explored. As shown in Fig. 4(a) the incumbent minimal cost drops fairly quickly, but the minimal cost of 2.25 kW is only found after having almost fully explored the search tree. The optimal u^*, which is the first element in the optimal switching sequence U^*, is found already after having explored 221 nodes. To obtain a certificate that this is indeed the optimal u^* the search tree has to be fully explored.

In contrast to that, with branch and bound, promising nodes are explored first and clearly inferior parts of the search tree are removed. As a result, the optimal cost c^* and the optimal u^* are found significantly earlier – in this example already after 61 steps. Some additional nodes need to be explored to prove that this is indeed the optimum. This certificate is obtained after a total of 140 nodes visited.

A few remarks about branch and bound. This algorithm does not impact optimality, i.e. the same optimal switching sequence is found as with full enumeration. In general, branch and bound drastically reduces the *average* computation time when compared to full enumeration. Yet, in the worst case, despite branch and bound techniques, a full enumeration of the search tree might be required to find not only the optimum

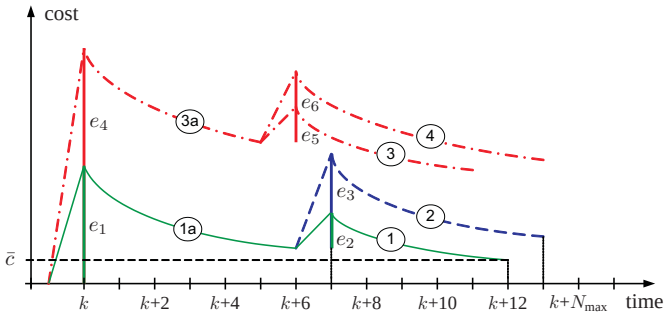


Fig. 3: Cost evolution (W) of switching sequences over time, were the e_i denote switching energy losses (Ws). The incumbent minimal cost $\bar{c} = (e_1 + e_2)/(12Ts)$ refers to sequence 1, and $N_{\max} = 14$ denotes the upper bound on the expected length of the prediction horizon

but also a proof (certificate) that the optimum has indeed been found. Such a certificate is provided when no more bud nodes exist with $\underline{c}_i < \bar{c}$. The optimal switching sequence is usually found a lot quicker. Note that we only require the first element of this sequence, i.e. the optimal u^*, which is found even quicker, as indicated by the arrows in Fig. 4(a).

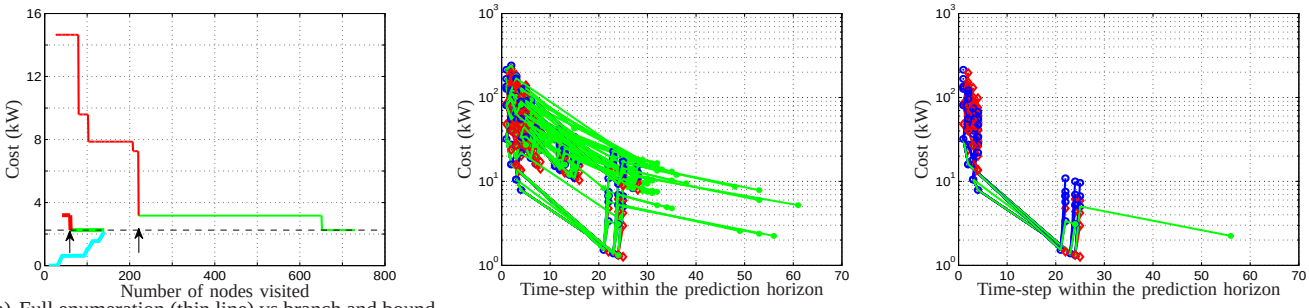
At every step during the optimization procedure the optimal cost is upper and lower bounded by $\underline{c} \leq c^* \leq \bar{c}$. As the optimization proceeds, these bounds are tightened. They provide information on how close the incumbent minimal cost is to the optimum. This can be seen in Fig. 4(a), where the upper thick line refers to $\bar{c}$, while the lower one corresponds to $\underline{c}$. Both lines converge to the optimal cost given by the dashed (black) line.

Branch and bound works best if the upper and lower bounds are tight. A tight upper bound $\bar{c}$ is achieved by finding a close to optimal leaf node with a low cost during an early stage of the optimization. To achieve this, depth-first search techniques can be employed and the optimal switching sequence from the previous time-step $k - 1$ can be used to warm start the optimization. A tight lower bound $\underline{c}$ is the result of a tight upper bound on the maximal length of the prediction horizon. During the optimization process, branching heuristics can help to identify and to focus on the most promising nodes first.

C. Limiting the Maximal Number of Computations

In a practical controller implementation only a certain number of computations can be performed within the sampling interval, which is typically of a fixed length in time. Therefore, it might be necessary to limit the maximal number of computational steps and/or to impose an upper bound on the computation time. Aborting the branch and bound optimization before a certificate of optimality is obtained might lead to suboptimal solutions, i.e. switching sequences that yield a higher cost than the optimal sequence. Yet, as explained in the last section, in most cases the optimum will have been found already.

Since long prediction horizons lead to a better performance (i.e. lower switching losses for the same distortion levels and vice versa), MPDTC's performance can be improved by considering longer switching horizons, by imposing an upper bound on the number of computations and by accepting that the result is in some cases (slightly) suboptimal. This is in contrast to the classic approach of using fairly short switching



(a) Full enumeration (thin line) vs branch and bound (thick line)

(b) Full enumeration

(c) Branch and bound

Fig. 4: Evolution of the optimal cost (kW) when solving one instance of the MPDTC optimization problem. (a) shows the incumbent minimal cost vs the number of nodes visited, with the arrows indicating when the optimal u^* is found. The thick cyan line refers to the minimum $\underline{c}$ of the lower bounds $\underline{c}_i$. (b) and (c) depict for every switching sequence the cost as a function of the time-step within the prediction horizon. Complete candidate switching sequences are green, incomplete (i.e. pruned) candidate sequences are blue and non-candidate sequences are red. Note the logarithmic scaling of the cost in (b) and (c)

horizons, and to ensure that the search tree can be fully enumerated in the time available and that the optimal solution is thus found under all circumstances.

More specifically, one of the following stopping criteria can be added to the MPDTC algorithm.

- An upper bound on the number of nodes to be explored and/or the computation time available is imposed. The optimization is halted if this number or time is exceeded, and the switching sequence with the incumbent minimal cost is accepted as the solution.
- Alternatively, one may run the optimization procedure for as long as possible, e.g. until an interrupt is received to stop it. This allows one to reduce idle processor time and to rather spend this time on improving the incumbent optimal solution.

One might also stop with a guarantee of closeness to optimality. Choose an acceptable distance to optimality in terms of the cost, e.g. 5%, and stop the search when $\underline{c} \geq 0.95\bar{c}$.

D. Computationally Efficient MPDTC Algorithm

In the sequel a computationally efficient version of the MPDTC algorithm is presented that is based on a tailored branch and bound technique to reduce the average computational burden. By adding the upper bound $j_{\max}$ on the number of explored nodes j, the maximal computational burden is limited, too.

1) Initialize the root node and push it onto the stack. Set $\bar{c} = \infty$ and $j = 0$.

2a) Take the node i with a non-empty A_i from the stack that has the minimum cost $\underline{c}_i$.

2b) Read out the first element from A_i and remove it. For 'S', branch on all admissible switching transitions and add the (incremental) switching losses $E_i = E_i + e(\ell)$ caused by switching at the time-step ℓ. For 'E', extend the trajectories either by extrapolation as detailed in [2] or by using the internal controller model [6].

2c) Set $j = j + \#S$ for 'S', where $\#S$ denotes the number of switching transitions, and set $j = j + 1$ for 'E'.

2d) Keep only the nodes that are candidates.

2e) For leaf nodes: compute their costs c_i and update $\bar{c} = \min(\bar{c}, c_i)$. For bud nodes: compute the lower bounds $\underline{c}_i$ on their costs and remove bud nodes with $\underline{c}_i \geq \bar{c}$.

2f) Push the remaining nodes onto the stack.

2g) Stop if there are no more nodes with non-empty A_i on the stack or if $j > j_{\max}$.

3) Compute for each (candidate) leaf node $i \in \mathcal{I}$, with $\mathcal{I}$ being an index set, the associated cost $c_i = E_i/(N_i T_s)$.

4) Choose the leaf node i with $c_i = \bar{c}$ and read out the associated switching sequence $U^* = U_i(k)$.

5) Apply (only) the first switch position $u(k) = u^*$ of this sequence and execute the above procedure again at the next time-step $k + 1$.

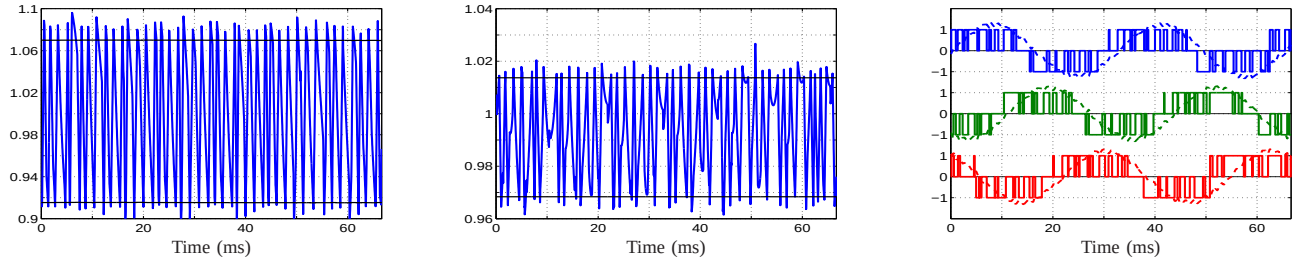
V. Computational Results

As a case study, consider a three-level voltage source inverter driving an electrical machine with a constant mechanical load. A 3.3 kV and 50 Hz squirrel-cage induction motor rated at 2 MVA is used as an example for a commonly used medium-voltage machine. The detailed parameters of the drive can be found in [6]. At 60% speed with a 100% torque setpoint the steady-state performance of DTC was compared with the one of computationally efficient MPDTC for short and long switching horizons.

For this comparison, a very accurate and detailed Matlab/Simulink model of the drive was used, which was provided by ABB to ensure a realistic simulation set-up. This model includes an outer (speed) control loop that adjusts the torque reference and the (time-varying) bounds on the torque accordingly. The induction motor model includes the saturation of the machine's magnetic material and the changes of the rotor resistance due to the skin effect. To run MPDTC, the DTC look-up table was replaced by the computationally efficient version of the MPDTC algorithm. The significance of such simulations is underlined by the very close match between the previous simulation results in [3], which were obtained using the same model, and the experimental results in [8].

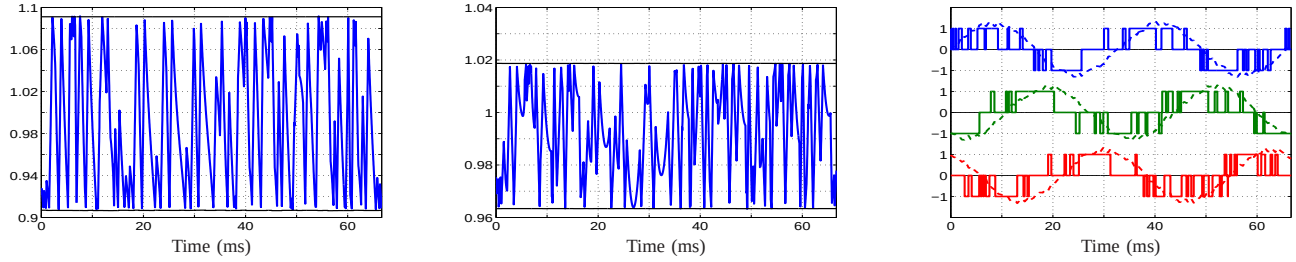
Using pu quantities, the comparisons are shown in Figs. 5 to 7. For MPDTC the torque and flux bounds were widened by 1.5% and 0.5%, respectively, to account for DTC's imminent violations of the bounds. As a result, DTC and MPDTC yield similar current THDs, while for MPDTC with long horizons the torque THD is slightly lower than for DTC, see Table I.

More importantly, with respect to DTC, MPDTC with the fairly short switching horizon 'eSSE' reduces the switching



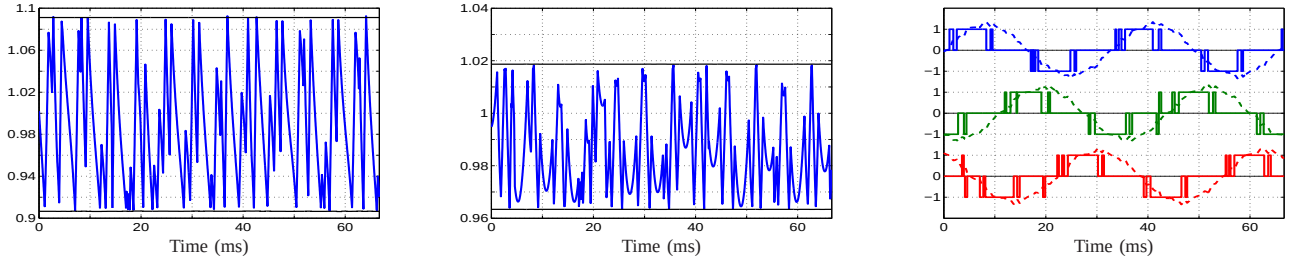
(a) Electromagnetic torque (b) Stator flux magnitude (c) Switch positions with stator currents

Fig. 5: Standard DTC, corresponding to the first row in Table I



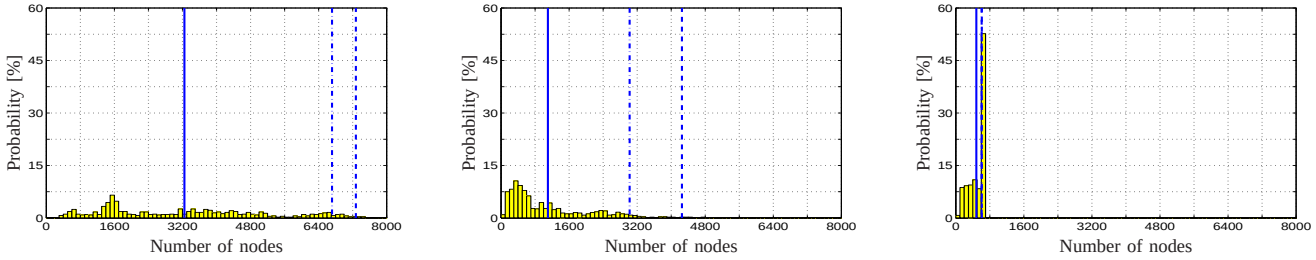
(a) Electromagnetic torque (b) Stator flux magnitude (c) Switch positions with stator currents

Fig. 6: Computationally efficient MPDTC with the switching horizon 'eSSE', corresponding to the fifths row in Table I



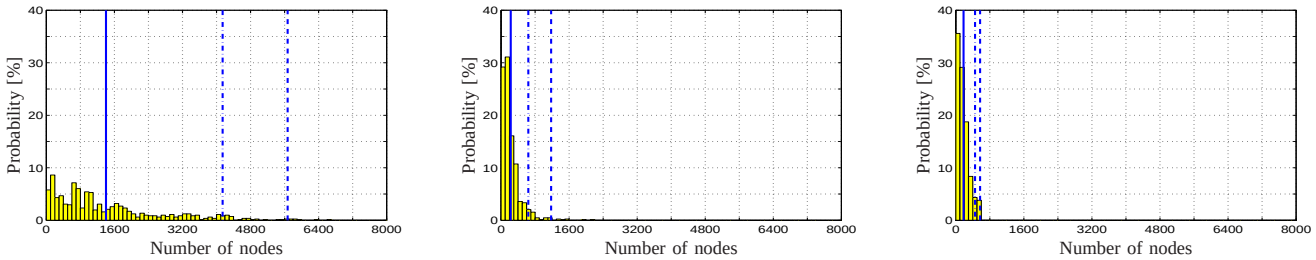
(a) Electromagnetic torque (b) Stator flux magnitude (c) Switch positions with stator currents

Fig. 7: Computationally efficient MPDTC with the switching horizon 'eSSESESE', corresponding to the last row in Table I



(a) Full enumeration (b) Branch and bound with $N_{\max}$ (c) Branch and bound with $N_{\max}$ and $j_{\max}$

Fig. 8: Histogram of the computational burden (number of explored nodes) for MPDTC with the switching horizon 'eSSESESE'



(a) Full enumeration (b) Branch and bound with $N_{\max}$ (c) Branch and bound with $N_{\max}$ and $j_{\max}$

Fig. 9: Histogram of the number of nodes required to be explored to obtain the optimal cost c^* for MPDTC with the switching horizon 'eSSESESE'

Controller settings				Pred. horizon		Nodes explored		u^*	Performance			
Scheme	Sw. horizon	$N_{\max}$	$j_{\max}$	avg.	max.	avg.	max.	found [%]	P_{sw} [%]	f_{sw} [%]	$I_{s,\text{THD}}$ [%]	$T_{e,\text{THD}}$ [%]
DTC	–	–	–	–	–	–	–	–	100	100	100	100
MPDTC	eSSE	–	–	26.6	96	112	277	100	57.3	71.2	103	98.4
MPDTC	eSSE	100	–	25.7	92	86.9	275	100	57.3	71.2	103	98.4
MPDTC	eSSE	50	–	25.6	96	64.3	249	97.4	57.7	73.4	103	102
MPDTC	eSSE	50	50	22.0	97	43.6	50	92.2	58.3	74.1	104	103
MPDTC	eSSESESE	–	–	98.2	150	3246	7693	100	37.9	48.9	97.0	92.0
MPDTC	eSSESESE	150	–	95.2	150	1884	6806	100	37.9	48.9	97.0	92.0
MPDTC	eSSESESE	110	–	92.5	150	1102	4756	96.7	40.9	50.0	99.5	92.2
MPDTC	eSSESESE	110	600	88.0	152	483	600	92.1	38.6	51.4	97.3	94.0

TABLE I: Comparison of DTC with MPDTC and computationally efficient MPDTC with the upper bounds $N_{\max}$ and $j_{\max}$ on the horizon length and the number of nodes explored, respectively. The fifths and sixths columns indicate the average and maximal lengths of the achieved prediction horizon. Columns seven and eight state the number of explored nodes in the search tree, followed by the probability that the optimal $u(k)$ is found at every control cycle. The last four columns relate to the switching losses P_{sw}, switching frequency f_{sw}, current THD $I_{s,\text{THD}}$ and torque THD $T_{e,\text{THD}}$ using DTC as a baseline

losses by 40%. The standard MPDTC algorithm based on the full enumeration of the search tree requires the exploration of up to 277 nodes to achieve this result, as shown in the second row in Table I. Using branch and bound techniques and tightening $N_{\max}$ almost halves the average number of nodes visited. Yet, the maximal number of nodes explored remains high. Limiting the number of computations by setting $j_{\max} = 50$ leads to suboptimal results – in almost 8% of the cases a suboptimal $u(k)$ is computed, but this appears, at least in this particular case, to barely affect the performance. As a result, the maximal computational burden is reduced by 82% from 277 to 50 nodes explored.

For the long switching horizon 'eSSESESE' the switching losses are reduced by another 35% with respect to MPDTC with 'eSSE'. This is achieved, as can be observed from Fig. 7(c), by reducing the switching frequency by another 30% and by carefully redistributing the remaining switching transitions along the time-axis. As a result, about half of the transitions occur when the respective phase current and hence the incurred switching losses are small. Note that the current and torque distortions are not increased by doing this – instead, they tend to get smaller.

However, the computational burden for MPDTC with full enumeration is exorbitant with search trees featuring almost up to 8000 nodes. Branch and bound with a tight $N_{\max}$ cuts down the average computation time by two thirds, but a fairly small upper bound on the number of explored nodes, $j_{\max} = 600$, is required to achieve an overall reduction of more than 90%. As previously, the impact on the performance appears to be small. Figs. 8 and 9 show the histograms of the computational burden and the number of computational steps required to derive the optimal cost, respectively. The (blue) vertical lines denote percentiles, with the straight, dash-dotted and dashed lines referring to the 50%, 95% and 99% percentiles. The effect of branch and bound on the cost is particularly remarkable as shown in Fig. 9(b).

VI. Conclusions

This paper proposed a modified version of the MPDTC algorithm based on branch and bound techniques adopted from mathematical programming. Using upper and lower bounds on the optimal cost, suboptimal parts of the search tree can be identified and pruned thus avoiding the full enumeration of the tree, as it was required with the previous MPDTC algorithm.

Initial simulation results suggest that the worst-case computational effort can be reduced by an order of magnitude. In general, the longer the horizon the more significant is the percentage-wise reduction of the computational burden and thus the saving. This makes this scheme particularly attractive for MPDTC with very long prediction horizons. The techniques presented here are equally applicable to the recently proposed adaptation of MPDTC to the current control problem, Model Predictive Direct Current Control (MPDCC) [11].

It is expected that the branch and bound method will enable the implementation of MPDTC with long switching horizons on today's available computational hardware thus allowing one to take full advantage of MPDTC's performance benefits. Smart branching heuristics are expected to further cut down the computation time.

References

[1] I. Takahashi and T. Noguchi. A new quick response and high efficiency control strategy for the induction motor. *IEEE Trans. Ind. Applicat.*, 22(2):820–827, Sep./Oct. 1986.

[2] T. Geyer. *Low Complexity Model Predictive Control in Power Electronics and Power Systems*. PhD thesis, Automatic Control Laboratory ETH Zurich, 2005.

[3] T. Geyer, G. Papafotiou, and M. Morari. Model predictive direct torque control - part I: Concept, algorithm and analysis. *IEEE Trans. Ind. Electron.*, 56(6):1894–1905, Jun. 2009.

[4] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice – a survey. *Automatica*, 25(3):335–348, Mar. 1989.

[5] J. B. Rawlings and D. Q. Mayne. *Model predictive control: theory and design*. Nob Hill Publ., 2009.

[6] T. Geyer. Generalized model predictive direct torque control: Long prediction horizons and minimization of switching losses. In *Proc. IEEE Conf. on Decis. and Control*, Shanghai, China, Dec. 2009.

[7] T. Geyer. A comparison of control and modulation schemes for medium-voltage drives: emerging predictive control concepts versus field oriented control. In *Proc. IEEE Energy Conv. Congr. and Exp.*, Atlanta, USA, Sep. 2010.

[8] G. Papafotiou, J. Kley, K. G. Papadopoulos, P. Bohren, and M. Morari. Model predictive direct torque control - part II: Implementation and experimental evaluation. *IEEE Trans. Ind. Electron.*, 56(6):1906–1915, Jun. 2009.

[9] E. L. Lawler and D. E. Wood. Branch and bound methods: A survey. *Op. Res.*, 14(4):699–719, Jul./Aug. 1966.

[10] S. Mastellone, G. Papafotiou, and E. Liakos. Model predictive direct torque control for MV drives with LC filters. In *Proc. Eur. Power Electron. Conf.*, Barcelona, Spain, Sep. 2009.

[11] T. Geyer. Model predictive direct current control for multi-level inverters. In *Proc. IEEE Energy Conv. Congr. and Exp.*, Atlanta, USA, Sep. 2010.

[12] J. E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of Applied Optimization, Oxford University Press*, pages 65–77, Jan. 2002.