Computationally Efficient Model Predictive Direct Torque Control

Tobias Geyer, *Senior 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, enabling MPDTC with long prediction horizons to be executed on today's available hardware.**

***Index Terms*—AC motor drives, model predictive control, optimal control, optimization methods, branch and bound**

I. Introduction

Direct Torque Control (DTC) [28] is ABB's method of choice for controlling the motor's torque and flux in medium-voltage drive applications [6], [30], with a typical product 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 [3], [4]. The switching losses, however, which represent a major part of the overall losses of the drive, can be substantial in DTC.

As shown in [11], [12], [17], 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) [10], [27]. As can be seen in Fig. 1, similar to DTC, MPDTC directly controls the torque and flux by manipulating the inverter switch positions. A modulator is not required. MPDTC is part of the family of emerging predictive control schemes for electrical drives, some of which are highlighted in [7]. Recent publications in this field include [29] and [20].

The initial version of MPDTC is available in two forms: with switching horizons of one or two steps [11], [17] and with prediction horizons in the range of a few dozen steps. MPDTC was generalized in [12], 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.

Initial results suggest that—with respect to state of the art DTC—MPDTC reduces the switching losses by up to 60%, while the Total Harmonic Distortion (THD) of the torque and the current is kept constant. Alternatively, the THDs can be reduced by the same amount, while keeping the switching losses constant [12]. These results are based on a medium-voltage three-phase Neutral Point Clamped (NPC) inverter driving an induction machine. Similar results [13] are obtained with respect to Field Oriented Control (FOC) with Pulse Width or Space Vector Modulation (PWM/SVM) [18].

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 THD are further lowered [13]. 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 [26].

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 [21]. 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

T. Geyer is currently with the Department of Electrical and Computer Engineering, The University of Auckland, Private Bag 92019, Auckland 1142, New Zealand; tel. number: +64 (9) 373 7599 extension 89634; e-mail: t.geyer@ieee.org

A preliminary version of this paper was published and presented at the ECCE 2010 in Atlanta, USA.

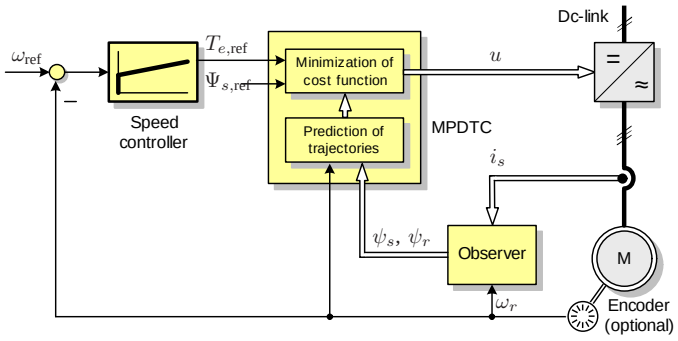


Fig. 1: Control block diagram including model predictive direct torque control (MPDTC), a speed controller and a flux observer

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 significance of such simulations is underlined by the very close match between previous simulations and experimental results using the same model. The simulation results in [17] predicted the experimental results in [26] accurately to within a few percent.

For one-step predictive drive control, methods to reduce the computational effort include [8], in which the search space of the considered voltage vectors is restricted to the ones neighboring the currently applied vector. Instead of ruling out voltage vectors *a priori*, the branch and bound technique presented in this paper removes voltage vectors dynamically *during* the optimization stage, basing the line of reasoning on the cost function rather than on the voltage vectors. The proposed method thus appears to be more elegant and less restrictive than previously reported approaches.

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, the implications of the proposed MPDTC algorithm are discussed in Sect. VI before conclusions are drawn in Sect. VII. Appendix A details the internal model used for the predictions, and Appendix B provides an introduction to the general concept of branch and bound.

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 to minimize the (short-term) average switching frequency [11], [17] or to directly minimize the switching losses [12], [22].

The MPC controller is endowed with a discrete-time model of the drive, which is derived and summarized in Appendix A. This internal model enables the controller 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. Typically, the sampling interval is very short with $T_s = 25\,\mu$s, while the prediction horizon entails up to 160 steps and is thus up to 4 ms long.

It is important to understand that, even though a sequence of switch positions is planned over a long prediction horizon, only the first switch position is implemented. The predictions are recomputed at the next sampling interval using new measurements and a shifted—and if necessary revised—sequence of switch positions is derived. This is referred to as the receding horizon policy, which provides feedback and makes MPDTC robust to parameter uncertainties in the underlying prediction model. As a result, the internal model used for the predictions is not required to be overly accurate.

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 is 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, see Appendix A-C. 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 ℓT_s within the prediction horizon, with $\ell = k, \ldots, k + N_p - 1$.

III. MPDTC with Full Enumeration

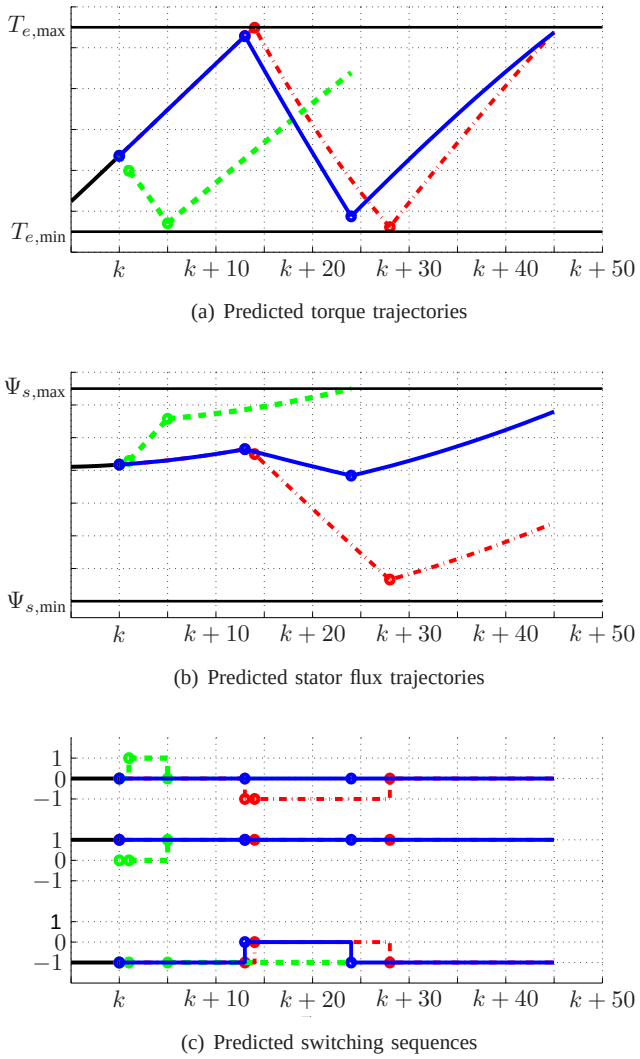
To solve the closed-form optimization problem (1) is challenging from a computationally point of view even for prediction horizons of modest length. Solving it for reasonably long horizons appears to be impossible[2]. Since this is a combinatorial optimization problem, it is well-known that, in the worst 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 [11], [12], [17] 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.



(a) Predicted torque trajectories

(b) Predicted stator flux trajectories

(c) Predicted switching sequences

Fig. 2: 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

It 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 trajectory ceases to be feasible, i.e. violates a bound, and/or ceases to point in the proper direction. Assume

[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.

[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.

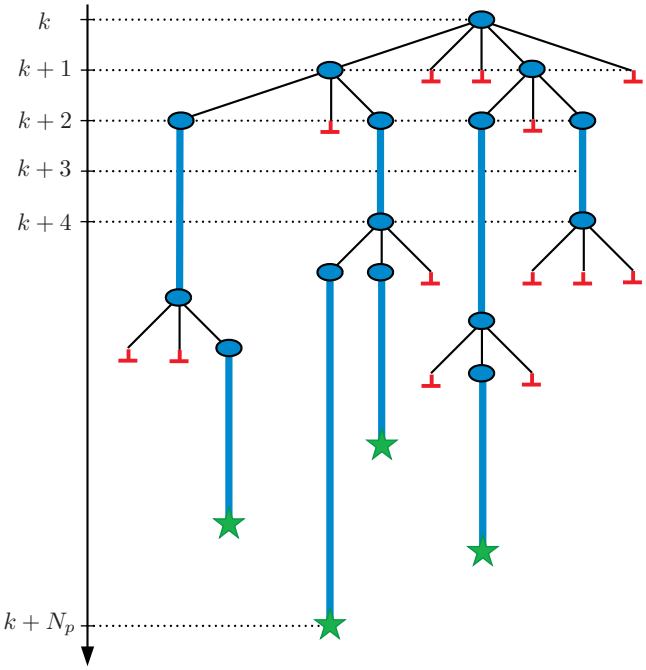


Fig. 3: Example of a search tree, induced by the switching horizon 'SSESE' with (blue) ovals denoting the root and bud nodes, (green) stars being complete candidate switching sequences, and inverted (red) Ts marking non-candidate switching sequences. Switching transitions 'S' are shown as thin (black) 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

this happens 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. 2 along with their output trajectories.

B. Search Tree

The switching horizon induces a search tree, as shown in Fig. 3. 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 using $\emptyset$ to denote an empty sequence. In Fig. 3, 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, are 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 trajectories and their cost. Hence all nodes in the search tree are visited

[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.

that belong to candidate switching sequences. Specifically, at time-step k, the MPDTC algorithm computes $u(k)$ according to the following procedure, which was presented in [12] 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 [11] and [31], or by using the internal controller model [12].
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.

The Steps 2a to 2c are executed until all nodes of the search tree have been enumerated. Subsequently, Steps 3 and 4 are run on a subset of the nodes, the candidate leaf nodes. Therefore, the computational burden of the MPDTC algorithm with full enumeration is proportional to the total number of nodes in the search tree.

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 discrete programs, such as combinatorial and (mixed) integer programs [21], [23].

A. Cost Evolution of a Switching Sequence

First of all, note that the evolution of the cost over time, associated with a switching sequence, 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. 4. 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.

B. Terminology

Before proceeding, some terminology needs to be introduced [25]. 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 $N_i \leq N_{\max}$ for all i.
- $\underline{c}_i = E_i/(N_{\max} T_s)$ 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 $\underline{c} \leq c^*$.

C. The Concept of Branch and Bound Tailored to MPDTC

The general concept of branch and bound is summarized in Appendix B. Hereafter, a modified version of branch and bound, which is tailored to the needs of MPDTC, is introduced in an intuitively accessible way. For this consider again Example 2.

Example 3: In Fig. 4, the cost associated with the complete candidate sequence 1 is $c_1 = (e_1+e_2)/(N_1 T_s)$, 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,

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

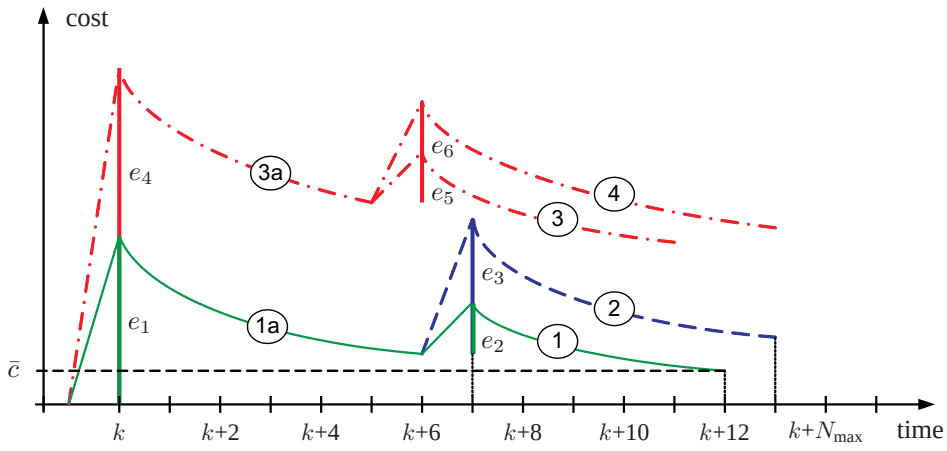


Fig. 4: Cost evolution (W) of switching sequences over time, were the e_i denote switching energy losses (Ws). The incumbent minimal cost $\bar{c} = (e_1 + e_2)/(12Ts)$ refers to sequence 1, and $N_{\max} = 14$ denotes the upper bound on the expected length of the prediction horizon

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 at 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-F.

D. Properties of Branch and Bound

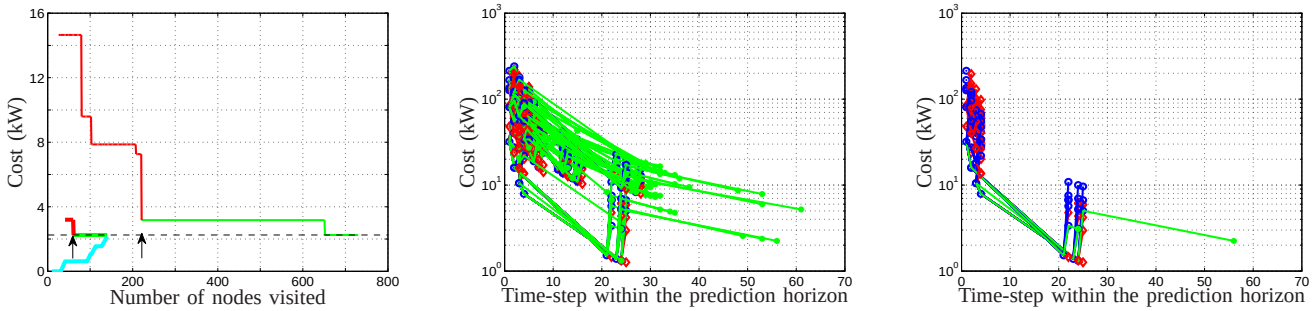
Example 4: Consider one instance of the combinatorial optimization problem, arising for example from a three-level inverter and a switching horizon of 'eSSESE'. The induced search tree contains 730 nodes. Using full enumeration, all 730 nodes are explored. As shown in Fig. 5(a), the incumbent minimal 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 concerning branch and bound [21] are in order. 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, 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. 5(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. The bounds provide information on how close the incumbent minimal cost is to the optimum. This can be seen in Fig. 5(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, which is 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.



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

(b) Full enumeration

(c) Branch and bound

Fig. 5: Evolution of the optimal cost (kW) while 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)

E. 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 horizons, so as 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 $\underline{c} \geq 0.95\bar{c}$.

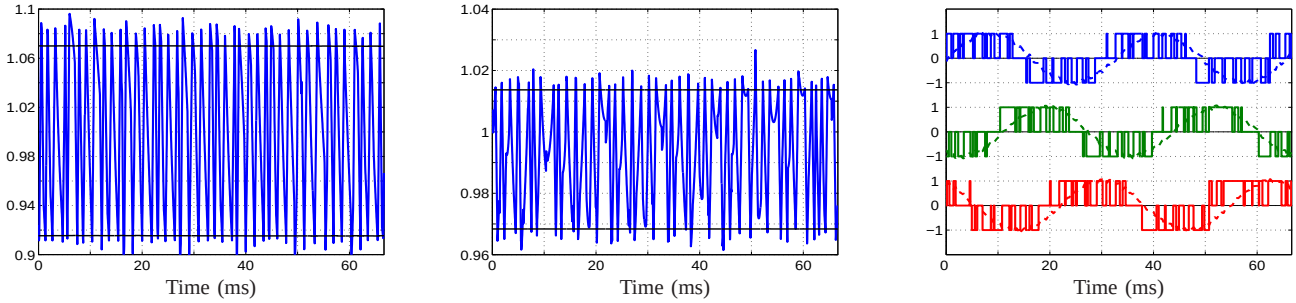
F. The Computationally Efficient MPDTC Algorithm

In the sequel, a computationally efficient version of the MPDTC algorithm is presented that is based on a tailored branch and bound technique that reduces 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 [11] and [31], or by using the internal controller model [12].
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 $c_i = E_i/(N_i T_s)$.

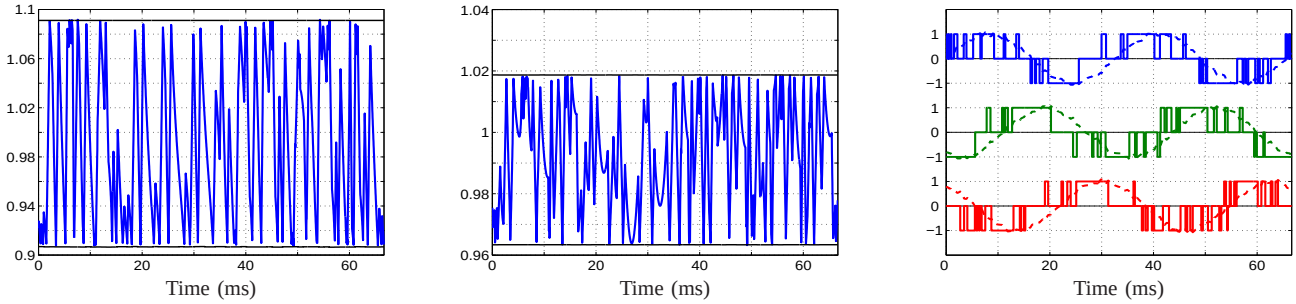
Induction Motor			
Voltage	3300 V	r_s	0.0108 p.u.
Current	356 A	r_r	0.0091 p.u.
Real power	1.587 MW	x_{ls}	0.1493 p.u.
Apparent power	2.035 MVA	x_{lr}	0.1104 p.u.
Frequency	50 Hz	x_m	2.3489 p.u.
Rotational speed	596 rpm		
Inverter			
Dc-link voltage	4294 V	V_{dc}	1.5937 p.u.
		x_c	11.769 p.u.

TABLE I: Rated values (left) and parameters (right) of the drive



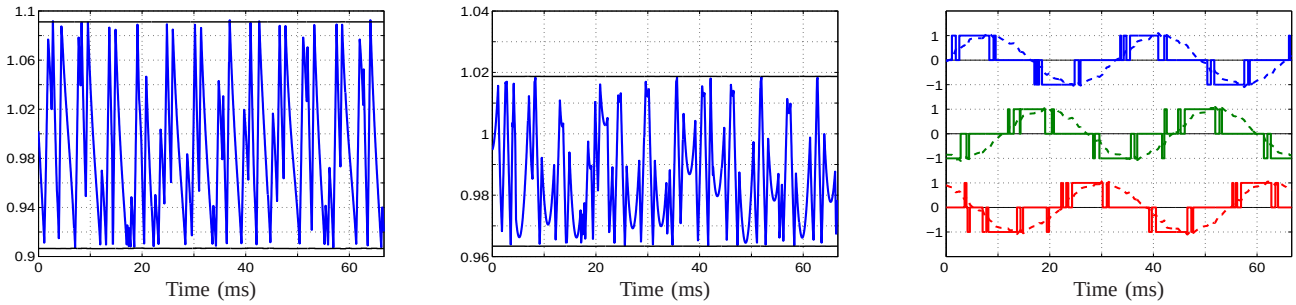
(a) Electromagnetic torque (b) Stator flux magnitude (c) Switch positions with stator currents

Fig. 6: Standard DTC, corresponding to the first row in Table II



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

Fig. 7: Computationally efficient MPDTC with the switching horizon 'eSSE', corresponding to the fifths row in Table II



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

Fig. 8: Computationally efficient MPDTC with the switching horizon 'eSSESESE', corresponding to the last row in Table II

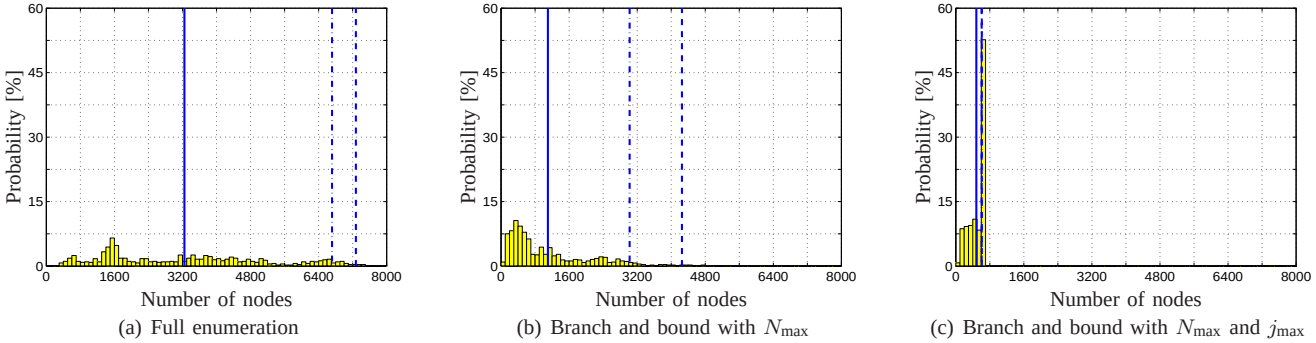
4) Choose the leaf node i with $c_i = \bar{c}$ and read out the associated switching sequence $U^* = U_i(k)$.
5) Apply (only) the first switch position $u(k) = u^*$ of this sequence, and execute the above procedure again at the next time-step $k + 1$.

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 Table I. 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.

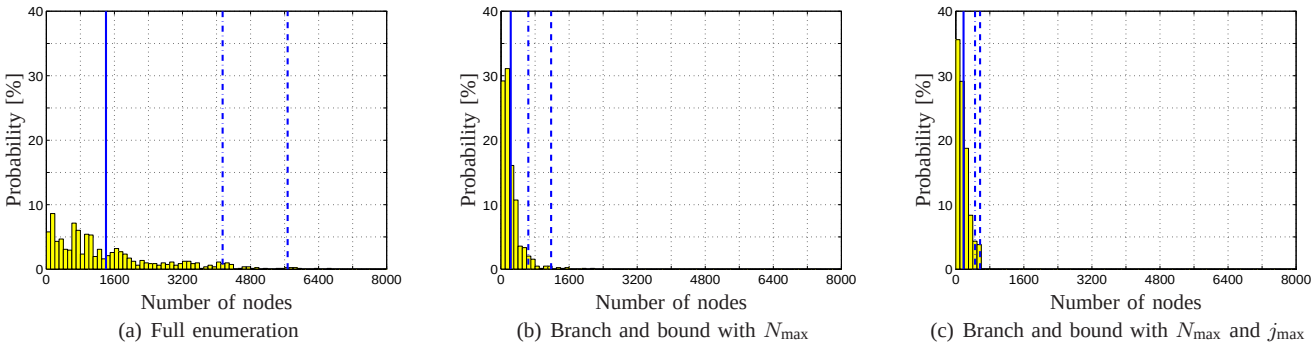
For this comparison, a very accurate and detailed Matlab/Simulink model of the drive system was used that represents the physical drive system[5]. This drive model was provided by ABB, to ensure a simulation set-up as realistic as possible. The model includes an outer (speed) control loop that adjusts the torque reference and the (time-varying) bounds on the torque accordingly, see also Fig. 1. The induction motor model includes the saturation of the machine's magnetic material, the changes of the rotor resistance due to the skin effect, and speed variations due to load torque changes. Furthermore, the inverter switching behavior, the inverter dead-time, minimum on- and off-times and fluctuations of the dc-link voltage are modelled. To run MPDTC, the DTC look-up table was replaced by the computationally efficient version of

[5]This model of the physical drive system is not to be confused with the controller's internal model used for the predictions.



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

the MPDTC algorithm. The significance of such simulations is underlined by the very close match between the previous simulation results in [17], which were obtained using the same model, and the experimental results in [26].

Using pu quantities, the comparisons are shown in Figs. 6 to 8. 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 II.

More importantly, with respect to DTC, MPDTC with the fairly short switching horizon 'eSSE' reduces the switching 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 II. 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. 8(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. 9 and 10 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. 10(b).

VI. Discussion

A. Minimization of the Switching Frequency versus Minimization of the Switching Losses

In general, as evidenced in Table II, MPDTC achieves a greater reduction in the switching losses than in the switching frequency. At a first glance, this might appear to be counter–intuitive. Recall that the control objective in Sect. II was

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 II: 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

to minimize the switching losses rather than the switching frequency. Accordingly, the cost function (1a) captures the switching losses. This is also reflected in Step 3 of the MPDTC algorithms in Sects. III-C and IV-F.

As a result, MPDTC arranges the switching pattern such that a significant number of the switch transitions takes place when the corresponding phase current is small. This can be observed in Fig. 8(c) and to a lesser extent in Fig. 7(c). In contrast to this, as shown in Fig. 6(c), DTC switches regardless of the phase current, similar to PWM and SVM.

Therefore, MPDTC not only switches less often—thus reducing the switching frequency—but it also aims at placing the switch transitions at times when the phase current is small, thus further reducing the switching losses. This is a key feature of MPDTC that was also highlighted in [12]. Alternatively, the switching *frequency* may be targeted in (1a) and in Step 3 of the algorithms, as proposed in [11], [17]. This tends to lead to slightly lower switching frequencies at the expense of higher switching losses, as shown in [12].

B. Practical Implications of the Proposed Algorithm

Recall that the computational burden of MPDTC is proportional to the number of nodes explored. Therefore, by reducing this number by an order of magnitude, the computational burden—and thus the computation time—can be reduced accordingly by an order of magnitude, when compared to the original algorithm based on full enumeration of the search tree.

So far, MPDTC has only been implemented and experimentally verified on a medium-voltage drive setup with a very short switching horizon, using the so called $N = 1$ approach [26]. To facilitate this, an additional FPGA was added to the DSP to perform the bulk of the MPDTC computations. With respect to this, MPDTC based on full enumeration and the switching horizon 'eSSE' improves the performance (lowers the switching losses) by about 10%, whilst increasing the computational burden fivefold[6]. Table II shows that the computationally efficient version of MPDTC can mitigate this fivefold increase, with very little impact on the performance (in the range of 1%). This implies that the proposed MPDTC scheme with 'eSSE' can be implemented on today's commonly used drive control hardware.

As shown in Table II, MPDTC with the switching horizon 'eSSE' reduces the switching losses by 40% with respect to DTC, while the distortion levels remain effectively unchanged. In the medium-voltage arena, the switching losses typically dominate over the conduction losses. When the thermal cooling capability is the limiting factor, lower switching losses enable one to increase the current accordingly. As a result, the power rating of the inverter hardware can be increased, i.e. a 10 MVA inverter, for example, can be uprated to 12 MVA and sold accordingly at a higher price. The standard drive control hardware, augmented with an FPGA, appears to suffice for this.

The switching losses can be reduced by another 35% and the inverter can be uprated by another 15 to 20%, when extending the switching horizon from 'eSSE' to 'eSSESESE', see Table II. At the same time, however, the computational burden is increased by a factor of 12, necessitating a significantly more powerful control platform. In light of the high price tag of medium-voltage drive equipment, this might be an attractive option—particularly in view of the ever increasing processing power of the computational hardware available.

C. Memory Requirement of the Search Tree

Another important aspect is the memory required to store the search tree. The memory requirement is equal to the maximal number of nodes considered at any time during the branch and bound optimization multiplied with the memory requirement of each node. Recall that each node in the search tree is given by the 6-tuple $\{U, X, Y, E, N, A\}$. The memory requirement of this 6-tuple is estimated to be 19 bytes (B)[7].

[6]MPDTC with the switching horizon 'eSSE' is conceptually very similar to the $N = 2$ approach, despite it minimizing the switching frequency. With respect to $N = 1$, $N = 2$ reduces the switching frequency by another 10%, but its computational effort is five times higher. For more details, refer to the analysis of the algorithms and the simulation results in [11], [17] and [26].

[7]The estimated 19 B result from the following reasoning. Out of the switching sequence U, only the first and the last switch position $u \in \{-1, 0, 1\}^3$ need to be stored, requiring 1 byte (B). It suffices to store the states and outputs at the end of the switching sequence rather than the complete sequences X and Y. Assume that the five states (four components of the machine fluxes and the neutral point potential) and the two outputs (torque and stator flux magnitude) are stored with 16-bit accuracy. Thus 14 B are required. The sum of the switching energy losses E is also stored with 16-bit accuracy and hence requires 2 B. The length of the switching sequence N is shorter than 256 and thus requires 1 B. The sequence of actions A to be performed can be captured by 1 B, if the switching horizon has at most eight elements. Therefore, each node requires 19 B.

The number of nodes considered by the algorithm at any time is upper bounded by the total number of nodes explored, $j_{\max}$. For the long switching horizon 'eSSESESE' and $j_{\max} = 600$, the memory required to store the search tree is thus upper bounded by $600 \cdot 19$ B, which is equal to 11.1 kB and thus very low.

VII. Conclusion

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. MPDTC with the switching horizon 'eSSE' can thus be implemented on today's standard drive control hardware. 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, offering the opportunity to implement MPDTC with such very long horizons, so as to allow one to take full advantage of MPDTC's performance benefits. Smart branching heuristics are expected to further cut down the computation time.

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) [14], as well as to other inverter topologies, such as five-level topologies [16] and synchronous machines [15].

Appendix A: Internal Prediction Model

This appendix outlines the derivation of MPDTC's internal prediction model. Starting with the continuous-time model of the inverter's neutral point potential and of the electrical machine, a discrete-time state-space model is derived. This was previously shown in [11], [17] and [12].

We use normalized quantities and a normalized time-axis. All variables $\xi_{abc} = [\xi_a \ \xi_b \ \xi_c]^T$ in the three-phase system (abc) are transformed to $\xi_{\alpha\beta 0} = [\xi_\alpha \ \xi_\beta \ \xi_0]^T$ in the orthogonal $\alpha\beta 0$ stationary reference frame through

$$\xi_{\alpha\beta 0} = P\,\xi_{abc} \tag{2}$$

with

$$P = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}. \tag{3}$$

A. Continuous-Time Inverter Model

Consider a neutral point clamped (three-level) inverter connected to a three-phase induction machine. The total dc-link voltage V_{dc} over the two dc-link capacitors x_c is assumed to be constant, while the neutral point potential υ_n between the two capacitors floats. Let the integer variables u_a, u_b, $u_c \in \{-1, 0, 1\}$ denote the switch positions in each phase leg. The voltage applied to the machine terminals is given by $v_{\alpha\beta 0} = 0.5 V_{\text{dc}} P\, u_{abc}$, with $v_{\alpha\beta 0} = [v_\alpha \ v_\beta \ v_0]^T$ and $u_{abc} = [u_a \ u_b \ u_c]^T$.

The neutral point potential υ_n depends on the state of charge of the two dc-link capacitors and is only affected when current is drawn directly from it, i.e. when one of the switch positions is zero. This yields

$$\frac{d\upsilon_n}{dt} = -\frac{1}{2x_c}\Big((1-|u_a|)i_{sa}+(1-|u_b|)i_{sb}+(1-|u_c|)i_{sc}\Big) \tag{4}$$

with the stator phase currents i_{sa}, i_{sb}, i_{sc} and one of the two symmetric capacitors x_c of the dc-link.

In the inverter considered here—due to the fact that only one di/dt snubber is available in the upper and the lower half, respectively—not all switch transitions are possible. Specifically, each phase leg can switch only by at most one step, at most two phase legs can switch at the same time, and if so, switching needs to occur in the opposite halves of the inverter. For example, from $[1\ 1\ 1]^T$, switching is only admissible to $[0\ 1\ 1]^T$, $[1\ 0\ 1]^T$ or $[1\ 1\ 0]^T$, and not to any of the other 23 switch positions.

The switching losses depend on the applied voltage, the commutated current and the semiconductor characteristics. Considering Integrated Gate Commutated Thyristors (IGCTs), with the GCT being the semiconductor switch, the switch-on and switch-off losses can be well approximated to be linear in the dc-link voltage and the phase current. For a diode, the switch-on losses are effectively zero, while the turn-off losses—the reverse recovery losses—are again linear in the voltage, but nonlinear in the commutated phase current. For a derivation of the switching losses, the reader is referred to [12].

B. Continuous-Time Machine Model

The squirrel-cage induction motor is modelled in the $\alpha\beta$ reference frame using the α- and β-components of the stator and the rotor flux linkages per second, $\psi_{s\alpha}$, $\psi_{s\beta}$, $\psi_{r\alpha}$ and $\psi_{r\beta}$, respectively, as state variables. The rotor speed dynamic is neglected and the rotor's rotational speed ω_r is assumed to remain constant within the prediction horizon. This allows us to treat the speed as a model parameter rather than as a state. The other model parameters are the base angular velocity ω_b, the stator and rotor resistances r_s and r_r, and the stator, rotor and mutual reactances x_{ls}, x_{lr} and x_m, respectively. The state equations are [19]

$$\frac{d\psi_{s\alpha}}{dt} = -r_s\frac{x_{rr}}{D}\psi_{s\alpha} + r_s\frac{x_m}{D}\psi_{r\alpha} + \upsilon_\alpha \tag{5a}$$

$$\frac{d\psi_{s\beta}}{dt} = -r_s\frac{x_{rr}}{D}\psi_{s\beta} + r_s\frac{x_m}{D}\psi_{r\beta} + \upsilon_\beta \tag{5b}$$

$$\frac{d\psi_{r\alpha}}{dt} = r_r\frac{x_m}{D}\psi_{s\alpha} - r_r\frac{x_{ss}}{D}\psi_{r\alpha} - \omega_r\psi_{r\beta} \tag{5c}$$

$$\frac{d\psi_{r\beta}}{dt} = r_r\frac{x_m}{D}\psi_{s\beta} + \omega_r\psi_{r\alpha} - r_r\frac{x_{ss}}{D}\psi_{r\beta} \tag{5d}$$

with $x_{ss} = x_{ls} + x_m$, $x_{rr} = x_{lr} + x_m$ and $D = x_{ss}x_{rr} - x_m^2$.

The electromagnetic torque is given by

$$T_e = \frac{x_m}{D}(\psi_{s\beta}\psi_{r\alpha} - \psi_{s\alpha}\psi_{r\beta}) \tag{6}$$

and the length of the stator flux vector is

$$\Psi_s = \sqrt{\psi_{s\alpha}^2 + \psi_{s\beta}^2}\,. \tag{7}$$

For more details, the reader is referred to [11], [17] and [12].

C. Discrete-Time Inverter and Machine Model

Combining the motor model (5)–(7) with the inverter model (4), taking advantage of the fact that the α- and β-components of the stator current $i_{s,\alpha\beta 0}$ are linear combinations of the stator and rotor flux components (see e.g. [19] for details), i.e.

$$i_{s,\alpha\beta 0} = \frac{1}{D}\begin{bmatrix} x_{rr}\psi_{s\alpha} - x_m\psi_{r\alpha} & x_{rr}\psi_{s\beta} - x_m\psi_{r\beta} & 0 \end{bmatrix}^T\,,$$

and using the Euler formula, a discrete-time model of the drive can be derived. As in standard DTC, a sampling interval of $T_s = 25\,\mu$s is used. The discrete-time model is omitted here due to space limitations, but it can be found in [11] and [17].

In summary, the internal prediction model includes the inverter switching behavior, restrictions on the switching transitions, the switching losses, the dynamics of the neutral point potential, and the standard dynamical model of an induction machine with four states, where the saturation of the machine's magnetic material, the changes of the rotor resistance due to the skin effect, and the temperature changes of the stator resistance are neglected. Moreover, the inverter dead-time is neglected. If required, variations on the dc-link voltage can be taken into account as well as changes of the machine's rotational speed.

Appendix B: General Concept of Branch and Bound

The branch and bound concept was developed in the 1960s. It has since become paramount in solving discrete optimization problems, such as optimization problems with boolean or integer variables. The solution space of such problems is discrete. Consider the problem of finding the minimum of an objective function J that is a function of the vector z of discrete decision variables, with z being restricted to the feasible set Ω. This can be formally written as

$$J^* = \min_z J(z) \tag{8a}$$

$$\text{s.t. } z \in \Omega\,. \tag{8b}$$

Rather than enumerating all possible solutions in the search tree, branch and bound seeks to reduce the number of investigated solutions by applying bounds. These bounds allow one to remove uninvestigated solutions from further consideration by proving that these solutions would be suboptimal. As a result, in general, only a small part of the set of solutions—and thus of the search tree—needs to be enumerated to find the optimal solution.

Specifically, as the name indicates, branch and bound consists of the following two operations.

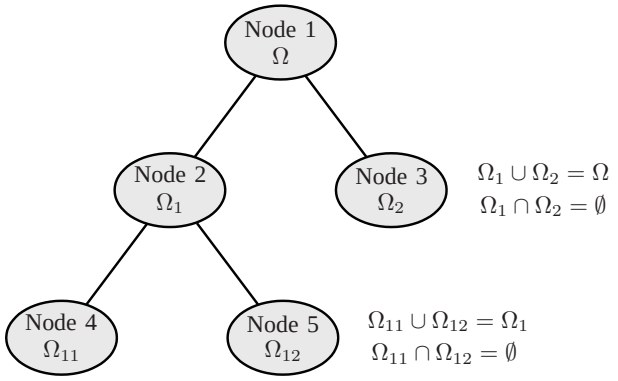


Fig. 11: Branch and bound concept. The set of feasible solutions Ω is recursively split into disjoint subsets. Upper and lower bounds on the objective function are applied to identify and to remove subproblems that contain only suboptimal solutions.

1) Branching: The optimization problem is recursively split into subproblems by dividing the set of feasible solutions Ω into two or more disjoint subsets, e.g. into Ω_1 and Ω_2, such that $\Omega_1 \cup \Omega_2 = \Omega$ and $\Omega_1 \cap \Omega_2 = \emptyset$.
2) Bounding: An upper bound is kept, which is the best solution found so far, i.e. the solution that yields the smallest value of the objective function. Lower bounds on the subsets' optimal solutions are usually provided by relaxations to ensure that they are quick to compute. If the upper bound is smaller than the lower bound for a subset, then the optimal solution cannot be part of this subset and the corresponding subproblem can be removed from further consideration.

Branch and bound is a universal concept that is highlighted in Fig. 11, which was reproduced from [2]. A good introduction and summary of the branch and bound methodology is provided in [2] and [5, Chaps. 12 and 13]. A more mathematical account is presented in [24] and [9, Chap. 8], while [21] provides a survey on branch and bound methods.

Acknowledgment

The author would like to thank Andrew D. Paice and Georgios Papafotiou of ABB Corporate Research, Baden-Dättwil, Switzerland for the particularly fruitful collaboration. A research grant from ABB Corporate Research, Baden-Dättwil, Switzerland is gratefully acknowledged.

References

[1] ABB Asea Brown Boveri Ltd. Product webpage of ACS 6000. Online document. www.abb.com/motors&drives.
[2] N. Agin. Optimum seeking with branch and bound. *Manage. Sci.*, 13(4):176–185, Dec. 1966.
[3] G. S. Buja and M. P. Kazmierkowski. Direct torque control of PWM inverter-fed AC motors – a survey. *IEEE Trans. Ind. Electron.*, 51(4):744–757, Aug. 2004.
[4] D. Casadei, F. Profumo, G. Serra, and A. Tanni. FOC and DTC: Two viable schemes for induction motors torque control. *IEEE Trans. Power Electron.*, 17(5):779–787, Sep. 2002.
[5] J. W. Chinneck. *Practical optimization: A gentle introduction*. Available online at www.sce.carleton.ca/faculty/chinneck/po.html. Draft, 2004.
[6] K. H. J. Chong and R.-D. Klug. High power medium voltage drives. In *Proc. IEEE Int. Conf. on Power Sys. Techn.*, pages 658–664, Singapore, Nov. 2004.

[7] P. Cortés, M. P. Kazmierkowski, R. M. Kennel, D. E. Quevedo, and J. Rodríguez. Predictive control in power electronics and drives. *IEEE Trans. Ind. Electron.*, 55(12):4312–4324, Dec. 2008.

[8] M. J. Duran, J. Prieto, F. Barrero, and S. Toral. Predictive current control of dual three-phase drives using restrained search techniques. *IEEE Trans. Ind. Electron.*, to appear 2011.

[9] W. Forst and D. Hoffmann. *Optimization — Theory and Practice*. Springer, 2010.

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

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

[12] T. Geyer. Generalized model predictive direct torque control: Long prediction horizons and minimization of switching losses. In *Proc. IEEE Conf. on Decis. and Control*, pages 6799–6804, Shanghai, China, Dec. 2009.

[13] 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 Convers. Congr. Expo.*, pages 2836–2843, Atlanta, USA, Sep. 2010.

[14] T. Geyer. Model predictive direct current control for multi-level inverters. In *Proc. IEEE Energy Convers. Congr. Expo.*, pages 4305–4312, Atlanta, USA, Sep. 2010.

[15] T. Geyer, G. A. Beccuti, G. Papafotiou, and M. Morari. Model predictive direct torque control of permanent magnet synchronous motors. In *Proc. IEEE Energy Convers. Congr. Expo.*, pages 199–206, Atlanta, USA, Sep. 2010.

[16] T. Geyer and G. Papafotiou. Model predictive direct torque control of a variable speed drive with a five-level inverter. In *Proc. IEEE Ind. Electron.*, pages 1203–1208, Porto, Portugal, Nov. 2009.

[17] 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.

[18] J. Holtz. Pulsewidth modulation – a survey. *IEEE Trans. Ind. Electron.*, 32(5):410–420, Dec. 1992.

[19] P. C. Krause, O. Wasynczuk, and S. D. Sudhoff. *Analysis of Electric Machinery and Drive Systems*. Intersci. Publ. John Wiley & Sons Inc., 2nd edition, 2002.

[20] T. Laczynski and A. Mertens. Predictive stator current control for medium voltage drives with LC filters. *IEEE Trans. Power Electron.*, 24(11):2427–2435, Nov. 2009.

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

[22] S. Mastellone, G. Papafotiou, and E. Liakos. Model predictive direct torque control for MV drives with LC filters. In *Proc. Eur. Power Electron. Conf.*, pages 1–10, Barcelona, Spain, Sep. 2009.

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

[24] L. G. Mitten. Branch–and–bound methods: General formulation and properties. *Op. Res.*, 18(1):24–34, Jan./Feb. 1970.

[25] I. Nowak. *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*. Birkhäuser Verlag, 2000.

[26] 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.

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

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

[29] R. Vargas, U. Ammann, B. Hudoffsky, J. Rodriguez, and P. Wheeler. Predictive torque control of an induction machine fed by a matrix converter with reactive input power control. *IEEE Trans. Power Electron.*, 25(6):1426–1438, Jun. 2010.

[30] B. Wu. *High-Power Converters and AC Drives*. Intersci. Publ. John Wiley & Sons Inc., 2006.

[31] Y. Zeinaly, T. Geyer, and B. Egardt. Trajectory extension methods for model predictive direct torque control. In *Proc. App. Power Electron. Conf. and Expo.*, to appear 2011.