Flexible Piecewise Function Evaluation Methods With Application to Explicit Model Predictive Control

Farhad Bayat, Tor Arne Johansen, *Senior Member, IEEE*, and Ali Akbar Jalali, *Member, IEEE*

***Abstract*— Efficient evaluation of piecewise functions defined over polyhedral regions is a vital problem in many scientific areas such as control applications. In this paper some efficient methods are proposed to deal with this problem. As an important application, the explicit model predictive control (eMPC) problem is considered which requires a piecewise affine (PWA) control law to be evaluated online. The widely used Binary Search Tree (BST) method is modified to be able to deal with a wider class of problems for which the BST method becomes prohibitive in terms of preprocessing time or memory requirements. The proposed method combines an Orthogonal Truncated Binary Search Tree (OTBST) and lattice representation for PWA functions in a unified structure enjoying the advantages of both approaches. The proposed Lattice-based OTBST (LOTBST) method enables the designer to trade-off between preprocessing time, storage requirement and online computation time. Using several examples it is shown that the proposed LOTBST leads to a considerably less preprocessing time and memory requirement comparing to the pure BST and less online computation time comparing to the pure lattice representation.**

I. INTRODUCTION

Piecewise functions have received significant attentions over the last few years and arise naturally in various applications such as control applications, e.g in the presence of constraints. Also the simplicity of piecewise affine systems make them attractive as an approximation to non-linear systems. In most of the applications efficient evaluation of the piecewise function is a crucial problem and can affect the performance of the whole system especially when there is large number of function pieces. In this paper we address efficient evaluation of a piecewise functions. Although this may seem trivial, but when the function is complex, a straightforward implementation is computationally expensive. This work is mainly motivated, but not limited, by recent developments within explicit solutions of Model Predictive Control (eMPC), in which the solutions are complex piecewise affine (PWA) state feedback laws to be evaluated online. Recently in [5] it was recognized that the solution to the linear MPC problem with quadratic cost can be formulated as a multi-parametric quadratic program (mp-QP) and solved explicitly, resulting in a PWA function of the current state $x(t)$. Similar ideas were developed for linear systems with $1/\infty$ norms in [4], and hybrid and piecewise linear systems in [6]. Also the approximate explicit MPC solution for nonlinear system was treated in [9]. All these approaches lead to a PWA control law defined over several convex polyhedral regions. So the evaluation of such a control law requires to determine the polyhedral region which contains the current state $x(t)$, the so-called point location problem. By exploiting the piecewise affinity of the associated value function for MPC with linear cost function, in [1] it was shown that the point location problem can be solved with no need to store the polyhedral partitions provided that a convex PWA function defined over the polyhedral partition is available. In the case of linear cost-function in [11] the point location problem was posed as a weighted Voronoi diagram, which can be solved using an approximate nearest neighbor algorithm. In [7] combining the concept of interval trees and bounding boxes associated with the polyhedral regions, a particular search tree was proposed to solve the point location problem. Although it can be applied to larger partitions, the online computational performance can be poor and there is no flexibility to trade-off between offline and online complexities in order to guarantee specific requirements. In [21] a Lattice Representation (LR) for the PWA solution of the explicit MPC was obtained based on the multi-parametric programming which can save online calculation and memory. Also in [22] a procedure was introduced to trade-off between the warm-start and online computational effort when a combination of explicit MPC and online computation is used. More recently, in [2] exploiting the concept of hashing theory a two-stage algorithm was proposed which combines the direct search method with an efficient set intersection algorithm together with some extremely efficiently solvable one-dimensional sub-problems (i.e. in the time of order $\mathcal{O}(1)$) to solve the whole problem. The hash-based method enables the designer to trade-off between online and offline complexities while it does not make any assumptions beyond the PWA structure. Therefore it can be applied for a general class of partitions including discontinuous and overlapped ones. Among the existing methods which can be applied (sometimes with some mild modifications) to more general class of partitions including discontinuous and overlapping ones, the Binary Search Tree (BST) [19] is acknowledged for its efficient online computational performance. Unfortunately, the pre-processing time in this method becomes prohibitive when the complexity of the partition increases. On the other hand, in several applications the achieved online computational performance using the BST method is beyond the practical requirements and may lead to idle processor time. This motivates us to modify the BST method by using such idle time to reduce

F. Bayat is with the Department of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran, (e-mail: fbayat@iust.ac.ir).

T.A. Johansen is with the Department of Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway, (e-mail: Tor.Arne.Johansen@itk.ntnu.no).

A.A. Jalali is with the Department of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran, (e-mail: ajalali@iust.ac.ir).

the pre-processing time and storage requirements and thereby enabling not only extended applicability of this method but also rapid prototyping of embedded control systems where a fast design iterations is of vital importance to be able to re-design controllers quickly by taking advantage of extended processing power typical for rapid prototyping environments. To this aim, by constraining the maximum available online computation time, a truncated binary search tree (TBST) is proposed. This truncation leads to extensive reduction in pre-processing time and requires additional search among the pre-computed function pieces remaining at the leaf nodes on the TBST to locate the optimal region. The BST data structure represents a recursive splitting of the domain of the piecewise function by including a new hyper plane at each node in the search tree. In the BST method these hyper planes are selected among the hyper planes defining the polyhedral regions of the piecewise function. As a reasonable alternative, we propose to select these hyper planes among a carefully selected set of axis-orthogonal hyper planes when constructing the BST. It is discussed in the next section that this replacement leads to faster evaluation of each node when traversing the tree. Furthermore, considering the continuous PWA solution of explicit MPC application an efficient alternative to the required direct search in each truncated leaf [3] is proposed by analytical representation of PWA control law [21] corresponding to each truncated leaf. This representation discards the requirement of storing all polyhedral regions for direct search and thereby reduces the storage complexity to a large extent.

The key features of the proposed method is that the exact solution (to the accuracy of numerical round-off errors) can be computed with pre-defined worst case online computation time guarantees, while pre-processing time and memory use is implicitly minimized subject to this constraint. The computations are readily implementable using fixed-point arithmetic on a low cost microcontroller or DSP since there is no recursive accumulation of round-off errors, and the online algorithm is simple with a small footprint suitable for formal verification of correctness of implementation. This is an advantage compared to other methods that rely on more complex online iterative floating-point computations in numerical optimization solvers [22] and [8], although promising new approaches may remove some of these limitations [17]. Unlike the methods [16],[20],[13],[10] and [15] that benefit from dedicated hardware (such as FPGA and ASIC), the present method targets the typical workhorses of embedded systems - low cost microcontrollers and DSPs, and also rapid prototyping environments.

The main contribution of the proposed algorithm is that it allows a flexible tuning and optimization of the trade-off between online and offline complexities. This flexibility provides the user direct control of the use of embedded system resources such as computation time and also online memory use, which is essential both in rapid prototyping design and production implementation.

II. PIECEWISE FUNCTION EVALUATION

Definition 1 ([12]): Let a compact set $X \subset R^n$ be partitioned into a set of N_r convex polyhedral regions $\mathcal{X}_i, i = 1, ..., N_r$ so that $X = \cup_{i=1}^{N_r} \mathcal{X}_i$. A piecewise function $p(x) : X \to R$ is defined as $p(x) = f(x|\phi_i) = f_i(x), \forall x \in \mathcal{X}_i$. Then $p(x)$ is said to be a PWA function if $f_i(x) = [x^T\ 1]\phi_i$ and continuous if $f_i(x) = f_j(x), \forall x \in \mathcal{X}_i \cap \mathcal{X}_j$.

A. Explicit MPC Application

Consider a constrained discrete time LTI system:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t \\ x_t &\in \mathbf{X}_c, u_t \in \mathbf{U}_c \end{aligned} \tag{1}$$

Where $A \in R^{n\times n}, B \in R^{n\times m}$ and $C \in R^{p\times n}$ and $\mathbf{X_c}, \mathbf{U_c}$ are polyhedral sets containing the origin in their interiors. Then the constrained finite time optimal control (CFTOC) problem for system described in (1) is defined as the following optimization problem:

$$\begin{aligned} J^*(x_t) = \min_U\ & \|Q_f x_{t+N}\|_p + \sum_{k=0}^{N-1} \|Qx_{t+k}\|_p + \|Ru_{t+k}\|_p \\ subject\ to\quad & x_{t+k} \in \mathbf{X}_c \quad \forall\, k = 1, ..., N \\ & u_{t+k} \in \mathbf{U}_c \quad \forall\, k = 1, ..., N-1 \\ & x_{t+k+1} = Ax_{t+k} + Bu_{t+k},\ k \geq 0 \end{aligned} \tag{2}$$

where $U = \left[u_t^T, u_{t+1}^T, ..., u_{t+N-1}^T\right]^T$ is the optimization variable, $\|\cdot\|_p$ denotes the standard vector norm when $p \in \{1, \infty\}$ and for $p = 2$, $\|Qx\|_p = x^TQx$. As shown in [5] and [4], for $Q \geq 0$ and $R > 0$ the solution to the CFTOC problem (2) at each time t is a continuous time-invariant PWA function of the current state $x(t)$ in the following form:

$$\begin{aligned} u^*(t) &= K_i x(t) + k_i, \quad if\ x(t) \in \mathcal{X}_i \\ \mathcal{X}_i &= \{x \in R^{n_x} | H_i x \leq h_i\}, \forall i = 1, ..., N_r \end{aligned} \tag{3}$$

Therefore, for the state $x(t)$, the online evaluation of explicit MPC requires to find the optimal polyhedral region $\mathcal{X}_\rangle$ containing the state $x(t)$, i.e. $x(t) \in \mathcal{X}_i$.

B. Binary Search Tree (BST)

In order to support efficient online evaluation of PWA functions, in [19] a binary search tree is constructed on the basis of the HPs defining the convex polyhedral regions $\mathcal{X}_i$. The idea is to use these HPs as separating variables in the nodes of the tree to split the partitioned space in a way that the number of candidate affine functions decreases as much as possible from the current to the next level of the tree. In online operation, and starting from the root node, one needs to identify on which side of the corresponding HP the current state resides and move through the tree until the optimal region is found. This approach lead to an efficient solution in terms of online evaluation complexity which is logarithmic in the number of regions (N_r). When the number of regions increases the cost of construction of the BST increases quickly and becomes prohibitive for large N_r with the algorithm [19]. This is due to the fact that for constructing

such a tree, for each node the BST algorithm in [19] needs to determine on which side of every HP each polyhedral region is located. This requires $2N_rN_H$ linear programs (LPs) to be solved where N_H denotes the number of HPs and is typically much larger than the number of regions. The storage requirement of the BST is in the order of $\mathcal{O}(n_xN_n)$, where N_n denotes the number of nodes of the tree. We emphasis that, less optimal trees can be generated by solving a reduced number of LPs or selecting HPs based on other criteria. The latter will be presented in section 2.D, while section 2.C introduces the idea of combining a truncated BST with direct search. Section 2.F introduces Lattice representation as an efficient alternative to direct search.

C. Truncated Binary Search Tree (TBST)

Consider the set of polyhedral regions $\{\mathcal{X}_1, ..., \mathcal{X}_{N_r}\}$ and the corresponding set of distinct functions $\{f_1, ..., f_K\}$ where $K \leq N_r$, since some regions may have the same function. Let $\{a_j^T x - b_j = 0, j = 1, ..., N_H\}$ denotes the set of all separating HPs defining the polyhedral partition. Let $d_j(x) = a_j^T x - b_j$ and define $\mathcal{J}$ as the index representation of a polyhedron ([19]) which consists of the corresponding HPs determined by d_j and their sign, e.g. $\mathcal{J} = \{2^+, 5^-\}$ means that $d_2(x) \leq 0$ and $d_5(x) \geq 0$. Such a set defines a polyhedron in the state space, i.e. $\mathcal{P}(\mathcal{J}) = \{x | d_2(x) \leq 0,\ d_5(x) \geq 0\}$. Further define the index sets $\mathcal{I}(\mathcal{J}) = \{i | \mathcal{X}_i \cap \mathcal{P}(\mathcal{J}) \text{ is full-dimensional}\}$ and $\mathcal{F}(\mathcal{I}) = \{k |\ f_k \text{ corresponds to } \mathcal{X}_i,\ i \in \mathcal{I}\}$ where $\mathcal{F}(\mathcal{I})$ denotes the index set of all functions corresponding to the index set $\mathcal{I}(\mathcal{J})$. We use the notation $'\pm'$ for statements which should be repeated for both $'+'$ and $'-'$.

Lemma 1 ([19]): If $i \in \mathcal{I}(\mathcal{J}) \cap \mathcal{I}(j^+)$ and $i \notin \mathcal{I}(\mathcal{J} \cup j^+)$, then $\mathcal{X}_i$ is split into two full-dimensional polyhedra by hyper plane j, i.e. $i \in \mathcal{I}(\mathcal{J}) \cap \mathcal{I}(j^+) \cap \mathcal{I}(j^-)$. The same result holds when j^+ is interchanged by j^-.

Lemma 1 provides a computationally efficient approximation of $\mathcal{I}_k^{\pm} = \mathcal{I}(\mathcal{J}_k \cup j^{\pm})$ as $\mathcal{I}(\mathcal{J}_k) \cap \mathcal{I}(j^{\pm})$ when one builds the k-th node of the tree. Based on these definitions and results, the modified algorithm to build the truncated binary search tree is presented in Alg.1.

Let $\mathcal{M}_{clk}$ be the maximum admissible number of clock cycles allocated for online piecewise function evaluation in the processor. We will use $\mathcal{U}$ denoting the list of all indices of those search tree nodes which are currently unexplored. When exploring the k-th node of the tree $\mathcal{N}_k$, let $(\mathcal{I}_k, \mathcal{J}_k)$ be the corresponding index set of regions and hyper planes as already defined. We introduce $\mathcal{C}_k = \mathcal{C}\{\mathcal{I}_k\}$ denoting the maximum number of clock cycles required to find the optimal region through the candidates in the index set $\mathcal{I}_k$ using direct search or other efficient alternatives. Then, one of the following conditions may occur during construction of the tree:

1. $|\mathcal{F}(\mathcal{I}_k)| = 1$, it means the exact solution is obtained and this node is flagged as a leaf node ($\mathcal{L}$).
2. $|\mathcal{F}(\mathcal{I}_k)| > 1,\ \mathcal{C}\{\mathcal{I}_k\} \leq \mathcal{M}_{clk}$ this means the online computational requirement is satisfied and it is possible to directly search through the members of $\mathcal{I}_k$ to find the optimal solution. This node is flagged as a truncated leaf ($\mathcal{TL}$), which is not pursued further in the BST construction.
3. $|\mathcal{F}(\mathcal{I}_k)| > 1,\ \mathcal{C}\{\mathcal{I}_k\} > \mathcal{M}_{clk}$ in this case the child node will be added to the tree and pursued further in the same way as in the BST method.

Algorithm 1: Building Truncated Binary Search Tree

1. Compute $\mathcal{I}(j^{\pm})$ for every $j \in J = \{1, ..., N_H\}$.
2. Initialize as $\mathcal{N}_1 \leftarrow \{(1, ..., N_r), \emptyset\}$ and $\mathcal{U} \leftarrow \{\mathcal{N}_1\}$.
3. **WHILE** $\mathcal{U} \neq \emptyset$ **DO**
   - i. Select any $\mathcal{N}_k \in \mathcal{U}$ and set $\mathcal{U} \leftarrow \mathcal{U} \backslash \mathcal{N}_k$.
   - ii. For all $j \in J$, compute $\mathcal{I}^{\pm} = \mathcal{I}(\mathcal{J}_k) \cap \mathcal{I}(j^{\pm})$ and sort the candidate optimal HPs by $\max_{j \in J} (|\mathcal{F}(\mathcal{I}^+)|, |\mathcal{F}(\mathcal{I}^-)|)$, and collect the index set of all optimal HPs as J_k^*.
   - iii. Compute the exact solutions $\mathcal{I}_k^{\pm} = \mathcal{I}\left(\mathcal{J}_k \cup j_k^{\pm}\right)$ for all $j_k \in J_k^*$. This is done by solving two LPs $\min_{x \in \mathcal{X}_i} \pm d_{j_k}(x)$ for each $i \in \mathcal{I}(\mathcal{J}_k) \cap \mathcal{I}(j_k^+) \cap \mathcal{I}(j_k^-)$ considering Lemma 1. Choose any optimal solution $j_k = \arg\min_{j_k} \max\left(\left|\mathcal{F}(\mathcal{I}_k^+)\right|, \left|\mathcal{F}(\mathcal{I}_k^-)\right|\right)$, $\mathcal{I}_k^{\pm} = \mathcal{I}\left(\mathcal{J}_k \cup j_k^{\pm}\right)$.
   - iv. **IF** $\left|\mathcal{F}(\mathcal{I}_k^{\pm})\right| = 1$ **THEN** Flag the resulting new node $\mathcal{N}^{\pm}$ as a leaf node ($\mathcal{L}$), and put $\mathcal{N}^{\pm} \leftarrow \left\{\mathcal{F}(\mathcal{I}_k^{\pm}), 0\right\}$.
   - v. **ELSE IF** $\mathcal{C}\left\{\mathcal{I}_k^{\pm}\right\} \leq \mathcal{M}_{clk}$ **THEN** Flag node $\mathcal{N}^{\pm}$ as a truncated leaf node ($\mathcal{TL}$), and put $\mathcal{N}^{\pm} \leftarrow \left\{\mathcal{F}(\mathcal{I}_k^{\pm}), -1\right\}$.
   - vi. **ELSE** Put $\mathcal{N}^{\pm} \leftarrow \left\{\mathcal{I}_k^{\pm}, \mathcal{J}_k \cup j_k^{\pm}\right\}$ as left/right child of the current node and add $\mathcal{N}^{\pm}$ to the list $\mathcal{U}$ as new unexplored node(s).
   - vii. **END IF**
4. **END EHILE**

Note that 0 and -1 in steps 3.iv and 3.v denote the leaf and truncated leaf, respectively. The proposed TBST method in Alg.1 has been illustrated and compared to the BST approach in Fig.1.

Remark 1: Observe that when the construction of the TBST in Alg.1 is terminated, storing $(\mathcal{I}_k, \mathcal{J}_k)$ is not required anymore for online application. Rather the corresponding decision HP and the pointers to the left and right children need to be stored for each node if it is not a leaf or truncated leaf. When the current node is leaf (truncated leaf) then index (indices) of the associated function (candidate optimal regions) should be stored rather than the pointers to the children.

The structure of the TBST is shown in Fig. 2. The first n_x+1 numbers, i.e. $\mathcal{TBST}_k\{1 : n_x + 1\}$, stores the information of optimal HP, i.e. (a_k, b_k), the next two numbers point to the left and right children of the current node respectively. $\mathcal{TBST}_k\{n_x + 2 \text{ or } n_x + 3\} < 0$ implies that left or right child is a leaf node and the index of optimal PWA function is $-\mathcal{TBST}_k\{n_x+2 \text{ or } n_x+3\}$, respectively. If $\mathcal{TBST}_k\{n_x+2 \text{ or } n_x + 3\} = 0$, then it means the left or right child is a truncated leaf and the indices of candidate regions are stored

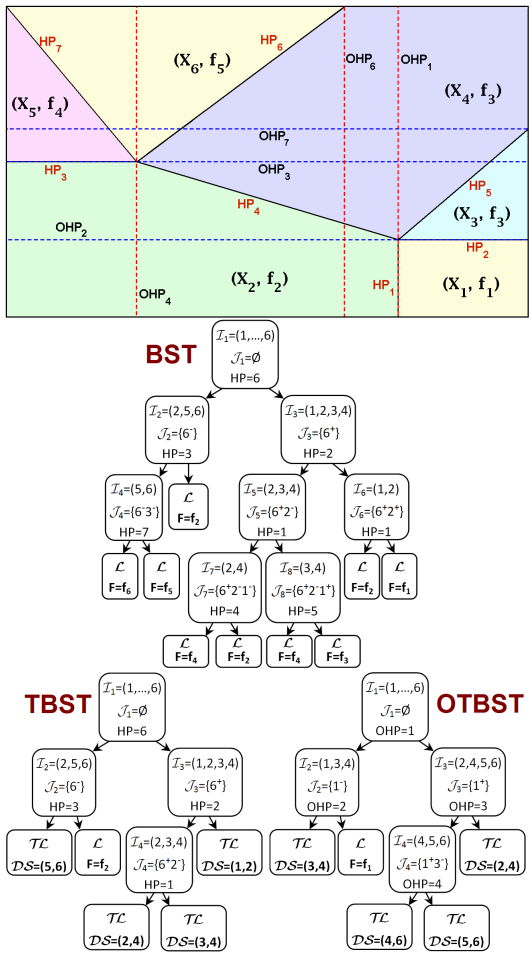

Fig. 1. Illustration of the BST, TBST and OTBST.

$\mathcal{TBST}_k\{1:n_x+1\}$	$\mathcal{TBST}_k\{n_x+2\}$	$\mathcal{TBST}_k\{n_x+3\}$	$\mathcal{TBST}_k\{n_x+4\}$	$\mathcal{TBST}_k\{n_x+5\}$

Fig. 2. The date structure of each node in the TBST method.

in $\mathcal{TBST}_k\{n_x+4 \; or \; n_x+5\}$, respectively.

Remark 2: The maximum required number of clock cycles in a truncated leaf node corresponding to the direct search and traversing the tree is typically calculated as $\mathcal{C}_k = \mathcal{C}_k^{ST} + \mathcal{C}_k^{DS} = \mathcal{H}_k^{ST}\mathcal{M}_H + \mathcal{H}_k^{DS}\mathcal{M}_H$, where $\mathcal{H}_k^{ST}$ denotes the number of traversed nodes until the current node, $\mathcal{M}_H = n_xC_{Mul} + 2C_{Comp} + (n_x-1)C_{Sum} + (n_x+2)C_{Mem} + 2C_{Branch}$ denotes the number of clock cycles required for evaluation of each HP, and $\mathcal{H}_k^{DS}$ denotes the number of all HPs corresponding to the candidate regions in node $\mathcal{TBST}_k$ excluding the region with maximum number of HPs. This is due to the fact that after ordering the $n_{\mathcal{TBST}_k}$ candidate regions in node $\mathcal{TBST}_k$ in terms of number of HPs, one needs to check only first $n_{\mathcal{TBST}_k}-1$ candidates to find the optimal one. The numbers $(C_{Mul}, C_{Sum}, C_{Comp}, C_{Mem}, C_{Branch})$ represent the number of clock cycles required for performing multiplication, summation, comparison, memory access and branching operations, respectively, on the target processor. In those processors for which some of the operations can

$\mathcal{OTBST}_k\{1,2\}$	$\mathcal{OTBST}_k\{3\}$	$\mathcal{OTBST}_k\{4\}$	$\mathcal{OTBST}_k\{5\}$	$\mathcal{OTBST}_k\{6\}$

Fig. 3. The fixed structure of each node in the OTBST.

be executed concurrently or are combined within a single operation, the formula for $\mathcal{C}_k$ should be modified accordingly.

Remark 3: The optimal HP j_k from step 3-(iii) may not be unique. Among the set of optimal HPs which are the same regarding the criterion in step 3-(iii), one can further refine the selection by one of the following secondary criteria:

1. $\min\limits_{j_k} \left(\max\left(\mathcal{C}\{\mathcal{I}_k^-\}, \mathcal{C}\{\mathcal{I}_k^+\}\right)\right)$,
2. $\min\limits_{j_k} \left(\max\left(\left|\mathcal{I}_k^-\right|, \left|\mathcal{I}_k^+\right|\right)\right)$,

Using the first secondary criterion, leads us not only to reduce the number of possible control laws while traversing from one level in the tree to the next, but also the required number of operations in online search and thereby reducing the depth of the truncated tree. The second criterion tries to reduce the number of candidate optimal polyhedral regions from one level of the tree to the next. The analysis of the modified method is similar to the standard BST method except that during the online search in the TBST method one may end up with a truncated leaf instead of a leaf. In this case it is necessary to search through the possible candidate regions to find the optimal solution.

Remark 4: In the case of explicit MPC applications, usually several regions have the same affine function especially when input constraints are included, so the decision criterion used in Alg.1 often reduces the size of the resulting tree to a large extent because when all candidate regions in the current node have the same affine function, then it can be considered as a leaf node. Otherwise, when the function pieces are mostly different, it will be more beneficial to replace the criterion in Alg.1 with $j_k = \arg\min\limits_{j_k} \left(\max\left(\mathcal{C}\{\mathcal{I}_k^-\}, \mathcal{C}\{\mathcal{I}_k^+\}\right)\right)$ as described in the Remark 3.

Algorithm 2: Online Piecewise Function Evaluation
Input: Any feasible query point $x = [x_1, ..., x_{n_x}]^T$.

1. Start from the root node $\mathcal{TBST}_k \leftarrow \mathcal{TBST}_1$.
2. If $\mathcal{TBST}_k\{n_x+2 \; or \; n_x+3\} \leq 0$ (leaf or truncated leaf), go to step 4.
3. Evaluate $d_k(x) = a_k^T x - b_k$ corresponding to the current node $\mathcal{TBST}_k$. If $d_k(x) \leq 0$ then take the right child as the current node, else take the left child as the current node. Return to step 2.
4. If $\mathcal{TBST}_k\{n_x+2 \; or \; n_x+3\} < 0$ (leaf node), then $-\mathcal{TBST}_k\{n_x+2 \; or \; n_x+3\}$ denotes the index of the optimal PWA function, else apply direct search to the set of candidate regions indicated by $\mathcal{TBST}_k\{n_x+4 \; or \; n_x+5\}$ to evaluate the optimal PWA function.

D. Orthogonal Truncated Binary Search Tree (OTBST)

In this section we further modify the BST method to achieve simpler structure of the search tree with considerably less pre-processing time. The main idea is to let the node decision criteria (HPs in TBST method) to be free (not

restricted to the existing HPs). Although this can lead to a better solution, but still it can be expensive in terms of pre-processing time. So among all combinations of decision criteria we are interested in orthogonal hyper-planes. We select some candidate OHPs directly based on the information of the existing polyhedral regions. Such a set of candidate OHPs can be obtained by the following simple LPs for each region $\mathcal{X}_r \in X,\ r = 1, ..., N_r$:

$$\min_{x \in \mathcal{X}_r} \pm e_i^T x, \qquad i = 1, ..., n_x \tag{4}$$

where e_i denotes the $i-th$ basic unit vector. The LPs in (4) provide the Bounding Hyper-Rectangle (BHR) representation of each polyhedral region. So $2n_xN_r$ candidate OHPs are produced as a set of candidate discriminating hyper-planes which is extensively less than its counterpart in the BST and also TBST methods, i.e. $2n_xN_r$. In the next step one can further refine the set of candidate decision variables by removing very close OHPs. This relies on the fact that the present method allows the binary search tree to end up with a truncated leaf instead of leaf which can contain more than one region. Considering the orthogonality of OHPs, it is clear that any two parallel OHPs with small distance comparing to the BHR length of the associated polyhedral regions in the corresponding axis, are likely not to be informative and removing one of them simplifies the pre-processing stage. A rule of thumb is to keep the one crossing more vertices. For example in Fig.1, $OHP_{6,7}$ are close to $OHP_{1,3}$ respectively and could be removed. In Alg.3 the conceptual steps of building the proposed OTBST are presented. The main features of the proposed method can be summarized as follows. (i) The total number of discriminating OHPs is usually much less than the HPs in the BST and TBST methods, leading to a significant reduction in the pre-processing time, (ii) In the first step of building the search tree one has to determine for each node decision criteria (HPs or OHPs), which side every region $\mathcal{X}_r$ lies on. This requires $2N_HN_r$ linear programs which can be replaced by simple comparison when OHPs are used, giving further reduction in the pre-processing demands, (iii) Unlike the BST method, both modified approaches (TBST and OTBST) enable the designer to trade-off between offline and online complexities, and (iv) In the OTBST the online evaluation of each node is more efficient than the BST. Traversing each node in the OTBST requires a single comparison, 3 memory access and 1 branching operations, while it takes n_x multiplications, n_x additions, 1 comparison, $n_x + 2$ memory accesses and 1 branching operation in the BST approach.

Remark 5: The last property (iv) implies that traversing the tree in the OTBST is computationally more efficient than the BST, especially when multiplication requires several clock cycles in a simple embedded (micro) controller. However, the major part of the online evaluation time in the OTBST method usually belongs to the search through the candidate regions recognized by truncated leaves and can be dominant comparing to the online evaluation time of the BST method. To deal with this problem, in the next section considering continuous PWA solution of explicit MPC an alternative is proposed by representing each truncated leaf in a compact analytical form that provides faster evaluation and less memory requirements than direct search used in [3].

Algorithm 3: Building the OTBST

Input: Maximum admissible clock cycles $\mathcal{M}_{clk}$ and the minimum distance thresholds $\delta_i,\ \ i = 1, ..., n_x$.

1. Compute the initial set of candidate OHPs using (4) and store the upper and lower bounds of each region as $\mathcal{W}_r = [L_r\ U_r],\ r = 1, ..., N_r$. Refine the initial set by removing the redundant OHPs which are identical or closer than δ_i to any parallel OHPs.
2. Let $j \in \bar{J} = \{1, ..., \bar{N}_H\}$ be the set of refined OHPs defined by a pair (i_j, h_j) where $d_j = e_{i_j}^T x - h_j$. Then compute the index set $\mathcal{I}(j^{\pm})$ for every OHP by a simple comparison based on the available information from step 1, i.e. $\mathcal{W}_r,\ r = 1, ..., N_r$, as
   - **IF** $L_r^{i_j} \leq h_j$ **THEN** add r to the index set $\mathcal{I}(j^+)$
   - **IF** $U_r^{i_j} \geq h_j$ **THEN** add r to the index set $\mathcal{I}(j^-)$
3. Initialize the root node by setting $\mathcal{N}_1 = \{\mathcal{I}_1, \mathcal{J}_1\} \leftarrow \{(1, ..., N_r), \emptyset\}$ and $\mathcal{U} \leftarrow \{\mathcal{N}_1\}$.
4. The rest of algorithm is similar to the steps 3-4 in Alg.1 replacing the HPs with the current OHPs and taking the notation in Remark 6 into account.

Remark 6: Unlike the BST and TBST methods, the proposed OTBST has a fixed node structure independent of problem dimension n_x. Similar to the Remark 1, the structure of the OTBST is shown in Fig.3. Accordingly, in each node $\mathcal{OTBST}_k\{1, 2\}$ stores the information of optimal OHP, i.e. (i_k, h_k), and $\mathcal{OTBST}_k\{3\ or\ 4\} > 0$ points to the left/right child, while $\mathcal{N}_k\{3\ or\ 4\} < 0$ implies the leaf node with negative sign of corresponding function index, and $\mathcal{N}_k\{3\ or\ 4\} = 0$ indicates the truncated leaf with the indices of candidate optimal regions stored in $\mathcal{OTBST}_k\{5\ or\ 6\}$. For a given feasible query point x, the online procedure of piecewise function evaluation is explained in Alg. 4.

Algorithm 4: Online Evaluation Using OTBST

Input: Any feasible query point $x = [x_1, ...x_{n_x}]^T$.

1. Start from the root node $\mathcal{OTBST}_k \leftarrow \mathcal{OTBST}_1$.
2. If $\mathcal{OTBST}_k\{3\ or\ 4\} \leq 0$ go to step 4. This means the associated child node is a leaf or truncated leaf node.
3. If $x_{\mathcal{OTBST}_k\{1\}} \leq \mathcal{OTBST}_k\{2\}$, then $k \leftarrow \mathcal{OTBST}_k\{4\}$, else $k \leftarrow \mathcal{OTBST}_k\{3\}$. Return to step 2.
4. If $\mathcal{OTBST}_k\{3\ or\ 4\} < 0$ (leaf) then $-\mathcal{OTBST}_k\{3\ or\ 4\}$ denotes the index of the optimal PWA function, respectively, else apply direct search to the set of candidate regions indicated by $\mathcal{OTBST}_k\{5\ or\ 6\}$ to find and evaluate optimal PWA function.

E. Analytical Representation of Truncated Leaves

In this section, assuming continuous PWA function evaluation we introduce an efficient approach as an alternative to the direct search in the proposed algorithms. To this aim we use the modified form of lattice representation model

[21] to represent each PWA restriction of the PWA function associated with each truncated leaf.

1) Modified Lattice Representation: The Lattice representation model has a universal representation capability and describes a PWA function in terms of its local affine functions and the order of the values of all the affine functions in each region [21],[18].

Lemma 2 ([18]): Given any n-dimensional continuous PWA function $p(x)$, there is a lattice PWA function $P(x|\Phi,\Psi)$ such that $p(x)=P(x|\Phi,\Psi)$ holds for all $x\in R^n$ and

$$P(x|\Phi,\Psi)=\min_{1\le i\le N_r}\left\{\max_{\substack{1\le j\le N_r\\ \psi_{ij}=1}}\{f(x|\phi_j)\}\right\} \tag{5}$$

where $\Phi=[\phi_1,...,\phi_{N_r}]^T$, $\Psi=[\psi_{ij}]$ and

$$\psi_{ij}=\begin{cases}1 & if\ \ f_i(x)\ge f_j(x)\\ 0 & else\end{cases} \tag{6}$$

As shown in [21] any continuous PWA function can be fully specified by a parameter matrix Φ and structure matrix Ψ.

Using the properties of min-max representation form of the lattice model and the possibility of having several regions with same affine function, two simplification algorithms were proposed in [21] as row and column vector simplification lemmas. The row vector simplification algorithm is presented in Lemma 3.

Lemma 3 (Row Vector Simplification Lemma [21]): Assume that $P(x|\ \Phi,\Psi)$ is a lattice representation with N_r segments. Let ψ_i, ψ_j be rows of the structure matrix. If the pointwise inequality $\psi_i-\psi_j\le 0$ holds for any $i,j\in 1,...,N_r$, there exists a simplified structure matrix $\tilde{\Psi}\in R^{(N_r-1)\times N_r}$, such that $P(x|\Phi,\Psi)=P(x|\Phi,\tilde{\Psi})$, where $\tilde{\Psi}=[\psi_1^T,...,\psi_{j-1}^T,\psi_{j+1}^T,...,\psi_{N_r}^T]^T$.

The column vector simplification lemma of [21] has been modified in Lemma 4 to prevent misrepresentation which may occur for some PWA functions. To resolve this problem an extra condition $\hat{\psi}_{kl}=0,\ \forall l$ was added in Lemma 4-(i). This problem is originated in this fact that the lemma in [21] has been analyzed and proved for one iteration of simplification for which a piece of PWA function f_i is removed since it can be represented by another piece f_j in its min-max form. When the lemma is used iteratively, the problem arises if the function piece f_j is itself removed by another representative f_k which is not a representative for f_i. The modified lemma is presented in Lemma 4.

Lemma 4 (Column Vector Simplification): Assume that $P(x|\ \Phi,\Psi)$ is a lattice representation with N_r segments. Let $\Psi=[\psi_{ij}]^{N_r\times N_r}$ and $\hat{\Psi}=[\hat{\psi}_{ij}]^{N_r\times N_r}$ denote the primary and dual structure matrices and define $I_i=\{k|f_k(x)\le f_i(x),\forall x\in\mathcal{X}_i\}$. Then the following results hold.

(i). Given any $i,j,k\in\{1,...,N_r\}$, if $k,j\in I_i$ and $\hat{\psi}_{jk}=1$, then set $\psi_{ij}=0$ and $\hat{\psi}_{kl}=0,\ \forall l=1,...,N_r$.

(ii). if $\psi_{ij}=0,\forall i\in\{1,...,N_r\}$, then there exist a simplified structure matrix $\tilde{\Psi}\in R^{(N_r-1)\times N_r}$ and parameter matrix $\tilde{\Phi}\in R^{N_r\times(N_r-1)}$ such that $P(x|\Phi,\Psi)=P(x|\tilde{\Phi},\tilde{\Psi})$, where $\tilde{\Phi}=[\phi_1,...,\phi_{j-1},\phi_{j+1},...,\phi_{N_r}]^T$ and $\tilde{\Psi}$ is the same as defined in Lemma 2.

Proof: The proof terms are the same as given in [21] except that the Eq. 20 in [21] is now hold for all cases considering the extra condition in Lemma 4-(i). ■

Utilizing the compact analytical model in (5)-(6) we aim to combine the OTBST and lattice representation approaches in a unified structure enjoying the advantages of both approaches. We emphasize that the OTBST method benefits from very low preprocessing time comparing to the BST method while the lattice representation approach provides a compact representation for the PWA function comparing to the polyhedral representation which is required in the OTBST for direct search. As a result, it can reduce the storage requirement to a large extent. Furthermore, since in online applications traversing the OTBST is done in a logarithmic time,thus replacing the direct search with the corresponding LR model can also reduce the online computation time taking the advantages of the simplification lemmas into account. The approach of combined lattice-based OTBST (LOTBST) is described in Alg.5. We remark, in the proposed approach the only information which is required to be stored for online application is structure and parameter matrices associated with each orthogonal truncated leaf instead of storing all polyhedral regions.

Algorithm 5: Lattice-OTBST (LOTBST)

Offline Procedure

1. Calculate explicit MPC solution for a given problem.
2. Calculate $\mathcal{OTBST}$ by applying Alg.3 to the PWA control law obtained in 1.
3. Using the Lemma 2 calculate lattice representation (parameter and structure matrices) associated with each truncated leaf ($\mathcal{TL}$) in the $\mathcal{OTBST}$.
4. Simplify the lattice representation using Lemma 3 and 4.

Online Procedure

1. For a given query point $x\in X$ apply Alg.4 to obtain if x belongs to a leaf node or truncated node.
2. If x belongs to a leaf node use the same procedure as in Alg.4, else use Eq.(5) in Lemma 2 to obtain $p(x)$.

Remark 7: Considering Remark 2, the maximum required number of clock cycles in an orthogonal truncated leaf node in LOTBST approach should be modified. Let N_k denotes the number of candidate regions in node $\mathcal{OTBST}_k$. Then, the online evaluation of LR model corresponding to each orthogonal truncated node consists of three parts. At first N_k affine function should be evaluated. In the worst case, this requires n_xN_k multiplications and n_xN_k sums. Then one need to calculate $N_k(N_k-1)$ maximization terms and N_k-1 minimization terms to evaluate LR model, in the worst case. Therefore, this min-max calculation requires N_k^2-1 comparisons totally. Finally, the LR model evaluation requires to load structure and parameter matrices which consist of $N_k^2+N_k(n_x+1)$ data. Therefore, $\mathcal{C}_k^{DS}$ is replaced with $\mathcal{C}_k^{LR}=(n_xN_k)C_{Mul}+(n_xN_k)C_{Sum}+(N_k^2-1)C_{Comp}+$

$(N_k^2 + N_k(n_x+1))C_{Mem}$, denoting the maximum required number of clock cycles corresponding to the lattice model evaluation of the current node. We emphasize that the actual required clock cycles for LR evaluation is considerably less than $\mathcal{C}_k^{LR}$ after applying Lemmas 3 and 4.

Remark 8: Note that the structure matrix contains only 0/1 elements, and therefore it is possible to pack and store these elements as bits of an integer number in the form of byte or word. This reduces the storage complexity as well as on-line calculation corresponding to reloading structure matrix. However it requires some online computation to decode 0/1s from integer values. This provides another flexibility to trade-off between storage and online computation complexities.

Remark 9: We emphasize that in the pure LR approach the storage requirement depends on the size of structure and parameter matrices. On the other hand, when the number of regions increases the storage complexity may increase quickly in the worst case where the simplification lemmas have no major effect, e.g. where most of affine function pieces are different. Hence, another important advantage of combining truncated search tree and LR model in the proposed approach is referred to this fact that each truncated node contains much smaller number of regions and it is more likely that some of those regions have the same affine function for which the simplification lemmas lead to a significant simplification. Furthermore, it is evident from the above explanation that one can easily trade-off between the computation (online/offline) and storage complexities by truncating the tree in different levels. To this aim, the parameter $\mathcal{M}_{clk}$ is a tuning knob which can control not only the mentioned trade-off but also guaranies the prerequisite online computation time according to the hardware on hand. This feature has been demonstrated by several numerical examples in section III (See Tables I-II).

Remark 10: Although the LOTBST method outperforms the proposed direct search based approaches (TBST and OTBST), but it should be emphasized that the TBST and OTBST are global approaches and can be applied to general piecewise nonlinear (PWNL) functions.

III. EXAMPLES

In this section the proposed piecewise function evaluation methods (TBST, OTBST and LOTBST), binary search method (BST), and pure lattice representation method (LR) have been implemented and compared via several examples. The clock cycle specifications in the following simulations are chosen based on the typical embedded (micro) controllers where multiplication, summation, comparison, memory access and branching operations take $(2, 1, 1, 2, 2)$ clock cycle(s) to be carried out, respectively.

Example 1: Double Integrator

Consider the discrete-time version of the double integrator

$$x(t+1) = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} T_s^2 \\ T_s \end{bmatrix} u(t) \qquad (7)$$

where $T_s = 0.05s$ and the system is subject to input constraints, $|u(t)| \leq 1$, and output constraint $|y(t) = x_2| \leq 1$. The CFTOC problem (2) is solved with $p = 2, Q = diag([1, 0]), R = 1$, $Q_f = 0_{2\times 2}$ for different horizon values $N = 2, 4, 8$ and 12. The explicit solutions characteristics and the comparison results of applying different PWA function evaluation methods are presented in Table 1. The pre-processing time in the OTBST and LOTBST methods have been extensively reduced comparing to the BST method with moderate extra online computation time. However the number of required online clock cycles is guaranteed to be admissible while constructing the tree in all approaches using the parameter $\mathcal{M}_{clk}$. The off-line computations were done on a 3GHz Pentium IV computer with some aid of the MPT toolbox for Matlab [14]. Considering the third test case $(N = 8)$, in Fig.4 it is shown that the proposed LOTBST method efficiently parameterizes alternative solutions with online computational performance between the pure LR and BST approaches when the parameter $\mathcal{M}_{clk}$ decreases from its upper to lower value limited by the LR and BST methods. Note that even for small $\mathcal{M}_{clk}$, in the OTBST and LOTBST approaches, most often there are some regions to be directly searched or represented by LR model. That is why it is not possible to exactly achieve the BST performance. This fact can be seen in Fig.4 as a discontinuity. For the same test case $(N = 8)$ and using a low cost processor, e.g. AVR-XMEGA series, with 32 MHz clock frequency, the PWA function evaluation procedure in the BST, TBST, OTBST and LOTBST methods, loads the processor with about 0.02%, 0.08%, 0.07% and 0.07%, respectively, when the sampling time $T_s = 0.05s$ is considered. In this case the direct search approach loads the processor with about 5%. Furthermore, the results in all test cases demonstrate extensive reduction in storage requirements when LOTBST is applied.

TABLE I

THE COMPARISON RESULTS OF EXAMPLE 1 FOR DIFFERENT HORIZON. — DENOTES IT IS NOT APPLIED TO THIS METHOD.

N	Method	N_r	$\mathcal{M}_{clk}$	N_n	Storage (Numbers)	Preprocessing Time(sec)	Online Clock-Cycles Min	Online Clock-Cycles Max
2	BST	83	—	188	752	27	147	187
	TBST		550	22	1096	11	107	513
	OTBST			33	1160	3	62	454
	LOTBST				179	5	136	323
	LR		—	—	256	0.2	16625	
4	BST	219	—	421	1792	160	147	207
	TBST		1000	33	2825	62	107	933
	OTBST			39	2890	6	73	952
	LOTBST				307	11	165	861
	LR		—	—	665	1	54711	
8	BST	627	—	1177	4736	1321	187	247
	TBST		1250	84	8204	514	107	1227
	OTBST			90	8230	40	62	1165
	LOTBST				716	51	95	1174
	LR		—	—	2585	4	317585	
12	BST	1325	—	2181	8704	5769	207	267
	TBST		1700	166	17202	2420	127	1655
	OTBST			131	17095	118	73	1531
	LOTBST				1177	151	95	1696
	LR		—	—	7756	19	1158990	

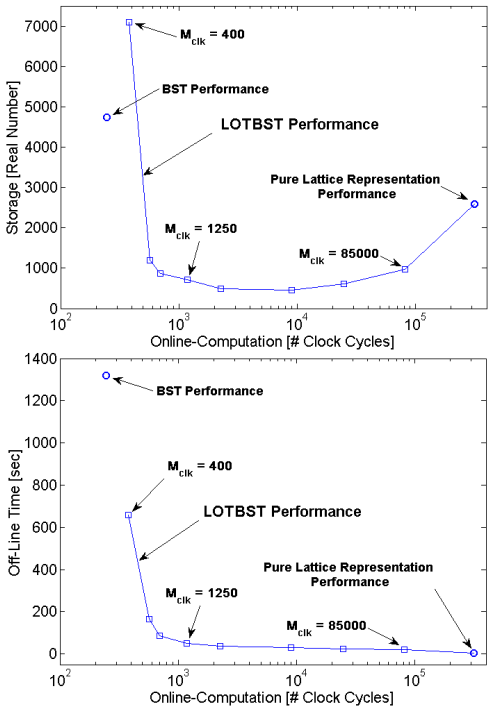

Fig. 4. Trade-off characteristics of the proposed LOTBST method, parameterizing performance of the BST and pure LR approaches.

IV. CONCLUSIONS

The problem of evaluating piecewise functions defined over polyhedral regions was investigated regarding its application in the embedded control systems. To this aim the well known BST approach was modified in a way that by making use of available online computation time, at first a truncated tree was introduced, then an orthogonal truncated tree was established by introducing artificial orthogonal discriminating hyper-planes instead of the HPs resulting from the polyhedral region descriptions giving extensive reduction in the preprocessing time comparing to the BST and TBST. Furthermore, as an important application the explicit MPC problem was considered which requires a continuous PWA function to be evaluated online. In the case of continuous PWA functions, as an alternative to the required direct search in the OTBST we proposed the LOTBST approach which represents control hyper-surfaces in each orthogonal truncated leaf (see Remark 5) in a compact analytical form utilizing the lattice representation model. According to the simulation results, the LOTBST approach is able to improve both online computation time and storage requirement of pure LR approach. Also, it reduces extensively the storage complexity and offline computation time in the BST approach, while guaranteeing the required online computation time. In summary, the simulation results showed that the proposed approaches open up the flexible use of binary search trees to a wider class of problems for which neither the BST nor LR approaches can be applied because of either prohibitive preprocessing time and storage requirement or high online computational complexity, respectively.

V. ACKNOWLEDGMENTS

The authors would like to thank Dr. Chengtao Wen for access to his software, and useful discussions on the lattice representation.

REFERENCES

[1] M. Baotic, F. Borrelli, A. Bemporad, and M. Morari, *Efficient on-line computation of constrained optimal control*, SIAM Journal on Cont. and Opt., 47(5), 2470-2489, 2008.

[2] F. Bayat, T.A. Johansen, A.A. Jalali, *Using hash tables to manage time-storage complexity in point location problem: Application to Explicit MPC*, Automatica, Accepted (2011).

[3] F. Bayat, T.A. Johansen, and A.A. Jalali, *Combining truncated binary search tree and direct search for flexible piecewise function evaluation for explicit MPC in embedded microcontrollers*, In 18th IFAC World Congres, Italy, 2011, Submitted.

[4] A. Bemporad, F. Borrelli, and M. Morari, *Model predictive control based on linear programming - the explicit solution*, IEEE Trans. on Automatic Control, 47(12), 1974-1985, 2002b.

[5] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos, *The explicit linear quadratic regulator for constrained systems*, Automatica, 38(1), 3-20, 2002a.

[6] F. Borrelli, *Constrained optimal control of linear and hybrid systems*, In Lecture Notes in Control and Information Sciences, volume 290, 2003.

[7] F. Christophersen, M. Kvasnica, C.N. Jones, and M. Morari, *Efficient evaluation of piecewise control laws defined over a large number of polyhedra*, In European Control Conference, 2360-2367, 2007.

[8] H. Ferreau, H. Bock, and M. Diehl, *An online active set strategy to overcome the limitations of explicit MPC*, Int. J. Robust and Nonlinear Control, 18, 816-830, 2008.

[9] T. A. Johansen, *On Multi-parametric Nonlinear Programming and Explicit Nonlinear Model Predictive Control*, IEEE Conf. Decision and Control, Las Vegas, NV, Vol. 3, pp. 2768-2773, 2002.

[10] T. A. Johansen, W. Jackson, R. Schreiber, P. Tøndel, *Hardware Synthesis of Explicit Model Predictive Controllers*, IEEE Transactions Control Systems Technology, vol. 15, pp. 191-197, 2007.

[11] C.N. Jones, P. Grieder, and S.V. Rakovic, *A logarithmic-time solution to the point location problem for parametric linear programming*, Automatica, 42(12), 2215-2218, 2006.

[12] C. Kahlert and L.O. Chua, *A generalized canonical piecewise linear representation*, IEEE Transactions on Circuits Systems I, 37(3), 373-382, 1990.

[13] G. Knagge, A. Wills, A. Mills, and B. Ninness, *ASIC and FPGA implementation strategies for model predictive control*, In European Control Conference, Budapest, 2009.

[14] M. Kvasnica, P. Grieder, M. Baotic, and M. Morari, *Multi-parametric toolbox (MPT). Hybrid Systems: Computation and Control*, Springe, 2004.

[15] K. Ling, B. Wu, and J. Maciejowski, *Embedded model predictive control (MPC) using a FPGA*, In IFAC World Congress, Seoul, 2008.

[16] T. Poggi, F. Comaschi, and M. Storace, *Digital circuit realization of piecewise affine functions with non-uniform resolution: Theory and FPGA Implementation*, IEEE Trans. Circuits and Systems - II: Express Briefs, 57, 131-135, 2010.

[17] S. Richter, S. Mariethoz, and M. Morari, *Highspeed online MPC based on a fast gradient method applied to power converter control*, In American Control Conference, Baltimore, 2010.

[18] J.M. Tarela, M.V. Martinez, *Region configurations for realizability of lattice piecewise-linear models*, Mathematical and Computer Modelling, 30(11-12), 17-27, 1999.

[19] P. Tøndel, T. A. Johansen and A. Bemporad, *Evaluation of Piecewise Affine Control via Binary Search Tree*, Automatica, Vol. 39, pp. 743-749, 2003.

[20] P. Vouzis, L. Bleris, M. Arnold, and M. Kothare, *A system-on-a-chip implementation of embedded real-time model predictive control*, IEEE Trans. Control Systems Technology, 17, 1006-1017, 2009.

[21] C. Wen, X. Ma, and B.E. Ydstie, *Analytical expression of explicit MPC solution via lattice piecewise-affine function*, Automatica, 45(4), 910-917, 2009.

[22] M. Zeilinger, C. Jones and M. Morari, *Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization*, In 47th IEEE Conf. on Decision and Control, 4718-4723, 2008.