A Novel Integrated Solution Algorithm for Explicit Model Predictive Control in Embedded Applications

Jamal Arezoo[1], and Karim Salahshoor[2]*

[1] Department of Automation and Instrumentation, Tehran Faculty of Petroleum, Petroleum University of Technology, Iran (e-mail: Jamalarezoo@gmail.com).

[2] Department of Automation and Instrumentation, Tehran Faculty of Petroleum, Petroleum University of Technology, Iran (e-mail: Salahshoor@put.ac.ir).

*Corresponding Author

Received 8 Sep 2021 Received in revised form 23 Feb 2022 Accepted 14 Mar 2022

Type of Article: Research paper

Abstract— **The paper addresses complexity of explicit model predictive control (MPC) in terms of online evaluation and memory requirement. Explicit MPC defines a piecewise affine (PWA) function over different regions of system state-space. An efficient approach is presented to integrate both complexity reduction schemes via 1) a separator function to remove regions defined over control actions which attain saturated values and 2) elimination of regions which have symmetry. The proposed method reduces the conventional necessity of explicit MPC, online evaluation, and storage requirement, by removing the regions corresponding to the saturated optimal control inputs using a simpler replacement function. Moreover, the method incorporates the concept of symmetries in the context of MPC quadratic programming problem to eliminate redundant symmetric regions, leading to devising a novel solution algorithm with much less complexities for embedded applications. The presented method also simplifies the symmetry identification process because the symmetry search algorithm is performed only for regions on which control action is unsaturated rather than the whole original regions. Various simulation tests are conducted to comparatively demonstrate effectiveness of the proposed algorithm for an inexpensive implementation of large-order systems in terms of the required storage and number of floating point operations.**

***Keywords*: Explicit MPC, quadratic programing, symmetric regions, and saturated regions.**

I. INTRODUCTION

MODEL predictive control is a dynamic optimization policy which is able to optimize performance in constrained multivariable systems [1], [2]. The optimization problem can be transformed to the form of a multi-parametric quadratic programming (mp-QP) problem. The problem is treated as a receding horizon fashion in which the mp-QP problem is solved online and only the first control input is then applied to the controlled system at each sampling instance [3], [4].

The ability to solve the mp-QP problem in embedded applications often poses a crucial issue where sampling period is small and storage limitation exists. In [5], an explicit solution of the mp-QP problem has been provided in offline manner in terms of a piecewise affine (PWA) function defined over a set of polytopic regions. The solution maps the current state measurement to the optimal control input. Therefore, online solution of the mp-QP problem is simplified to a PWA function evaluation which is fast and simple for embedded applications. However, the predefined explicit solution can be prohibitive as the number of polyhedral regions of $u^*(\mathrm{x})$ might grow exponentially with the number of constraints arising in the MPC optimization problem [6], [7].

Implementation of explicit MPC consisting of large number of linear control actions and their associated regions might be too expensive due to probable necessity of increasing the storage requirement [7], [8], [9]. Therefore, it is important to keep the number of control regions as low as possible. For this purpose, numerous studies have shown that simplifying the representation of the PWA function can be carried out to approximate explicit MPC by a simpler function [6], [7], [8], [9], [10]. For example, artificial neural networks [6], [7] are used to represent the explicit solution of the MPC problem. Nevertheless, the use of the techniques is challenging because stability guarantees and maximal performance have to be taken into account for all feasible states x [6], [7], [10], [11]. Hence, one way to avoid the challenges is to look for approaches which offer equivalent representations of the PWA function with no implication on feedback optimality rather than the approximate representations. An efficient method has been proposed in [12], [13] where an equivalent simple affine function has been employed to remove the regions whose associated control laws attain a saturated value, leading to significant reduction in the storage requirement. In another proposed approach [14], [15], symmetries of the MPC problem have been computed as a mathematical problem in order to reduce inherent complexity of explicit MPC with no change on optimality of the original representation. Identification of the set of all controller symmetries plays an important role in efficiency of the method of eliminating the symmetric regions [14]. To identify these controller symmetries, however, [15], [16] has shown the fact that the symmetries corresponding to control regions can be identified by using graph theory. For this objective, it is shown that the problem of finding controller symmetries is converted to graph automorphism problem which can easily be solved by the standard graph automorphism software packages [17], [19].

In this paper, an equivalent function $\tilde{u}^*(\mathrm{x})$ is introduced to avoid storage of many regions, called saturated regions, where their optimal control actions attain a saturated value. This interesting feature together with the concept of symmetries will be integrated in an MPC problem context to devise a novel solution algorithm for embedded applications. As the saturated regions are eliminated, the symmetric identification process is conducted only for regions whose control laws are not saturated so that precomputing the explicit MPC symmetry becomes less expensive. Moreover, the proposed algorithm offers an equivalent simpler function which guarantees the stability and control performance remain unchanged because the replacement function has no implication on optimality of the original function. The resulted algorithm simplifies the inherent complexities rooted in i) identification of controller symmetries and ii) the required look-up for all control actions and their corresponding regions, leading to an efficient explicit MPC implementation with much less complexity for large systems in which explicit MPC solution has a large number of regions. The main merit of this new algorithm can clearly be observed in extensive computational case studies, which shows the proposed approach is more efficient and general than the existing methods.

II. THEORETICAL BACKGROUND

A. Explicit Model Predictive Control

Throughout the paper, a class of discrete-time linear time-invariant (LTI) systems with the following structure will be considered for open-loop processes:

$$\begin{aligned} x^+ &= Ax + Bu \\ y &= Cx \end{aligned} \tag{1}$$

in which $x \in R^n$ indicates the state vector, $u \in R^m$ is the vector of system inputs, and x^+ denotes the state at the next sampling instant. The overall goal is to construct a simple state feedback controller in the context of MPC controller in order to solve the following constrained optimal control problem:

$$J(x) = \min_U \; x_N^T Q_N x_N + \sum_{k=0}^{N-1} x_k^T Q x_k + \sum_{k=1}^{N_c} u_k^T R u_k \tag{2a}$$

subject to
$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ x_{min} \le x_k &\le x_{max} \\ u_{min} \le u_k &\le u_{max} \\ \Delta u_{min} \le \Delta u_k &\le \Delta u_{max} \\ y_{min} \le y_k &\le y_{max} \end{aligned} \tag{2b}$$

where x_k and u_k denote, respectively, state trajectory and input sequence over finite horizons N and Nc, given the initial state x_0. It is assumed that Q, Q_N and R represent the penalty matrices which are, respectively, positive semi-definite, positive semi-definite and positive definite. Note that the preceding assumption implies that (2) is strictly a convex QP problem. The receding horizon MPC feedback then becomes $u^*(x_0) = [I\,0\ldots0]U^*$, where the optimal vector $U^* := \left[u_0^T, \ldots, u_{N-1}^T\right]^T$ can be found by solving (2) as a QP problem for a given value of the initial condition x_0. It should be mentioned that penalty can be imposed on Δu in (2a). In this case, (2) can be still translated into a QP problem [20].

One of the drawbacks of the MPC controller is the need to solve the QP problem in (2) online, which has traditionally made MPC as a control methodology for slow processes. As discussed in [5], an alternative approach to solve the QP problem (2) online with the initial state x_0, is to conduct an offline computation of (2) as a parametric quadratic program (pQP) and then take U^* explicitly as a piecewise affine and continuous function with the initial state in the following form:

$$u^*(x) = \begin{cases} F_1 x + G_1 & \text{if } x \in R_1 \\ \cdot & \\ \cdot & \\ \cdot & \\ F_n x + G_n & \text{if } x \in R_{n_r} \end{cases} \tag{3}$$

where $F_i \in R^{m \times n}$ and $G_i \in R^m$ are, respectively, real matrices and vectors of the MPC controller. R_i, $i = 1, \ldots, n_r$ are critical regions, and n_r denotes the total number of critical region. The triple $(F_i; G_i; R_i)$ is called the i-th controller piece. In this way, the explicit representation provides a simple implementation which includes a mere evaluation of piecewise affine function for a given x_0 at each sampling time.

B. Complexity Reduction via Elimination of the Saturated Regions

The complexity of an explicit MPC problem solution is depended on the number of regions to be stored. As already mentioned, in any typical explicit representation of (3), there exists many regions corresponding to the control actions which can attain either u_{min} or u_{max} for (2b). Therefore, (3) can be rewritten as follows [13]:

$$u^*(x) = \begin{cases} F_i x + G_i & \text{if } x \in R_{unsat} \\ u_{max} & \text{if } x \in \overline{R}_{sat} \\ u_{min} & \text{if } x \in \underline{R}_{sat} . \end{cases} \tag{4}$$

As a consequence, the regions R_i, $i = 1, \ldots, n_r$ can be sorted into three different types of R_{unsat}, $\overline{R}_{sat}$ and $\underline{R}_{sat}$, denoting the regions in which their corresponding control inputs attain, respectively, unsaturated, maximal and minimal saturated values. That is, if $x \in R_{unsat}$ then F_i and G_i can be nonzero and:

$$\text{if } x \in \overline{R}_{sat} \text{ then } F_i = 0 \textbf{\textit{ and }} G_i = u_{max} ,$$
$$\text{if } x \in \underline{R}_{sat} \text{ then } F_i = 0 \textbf{\textit{ and }} G_i = u_{min} .$$

The triple $(F_i; G_i; R_i \in R_{unsat})$ is called the unsaturated controller piece.

Theorem 1 [12]**:** Given a function $\tilde{u}^*$ as:

$$\tilde{u}^*(x) = \begin{cases} F_i x + G_i & \text{if } x \in R_{unsat} \\ u_{max} & \text{if } x \notin R_{unsat} , \beta(x) > 0 , \\ u_{min} & \text{if } x \notin R_{unsat} , \beta(x) < 0 \end{cases} \tag{5}$$

such that $\tilde{u}^*(x) = u^*(x)$ for all x belonging to the domain of $u^*(x)$, i.e. $x \in dom(u^*)$, where $\beta : R^n \to R$ is a function defined as:

$$\begin{aligned} \beta(x) > 0 , \quad & \text{if } x \in \overline{R}_{sat} , \\ \beta(x) < 0 , \quad & \text{if } x \in \underline{R}_{sat} . \end{aligned} \tag{6}$$

Proof: Ref. [12] has shown how to find the separator $\beta(x)$ such that (6) is satisfied.

Therefore, to find the regions on which query state x_0 lies and subsequently their corresponding control inputs, sequential search is first done through the regions of R_{unsat}. Then, if $x \in R_i \subset R_{unsat}$, the control input will be $F_i x + G_i$, otherwise the control input is determined by sign of the separator function β which will constantly be equal to u_{max} or u_{min}. Thus, instead of storing all regions, only unsaturated regions and separator function are needed, leading to complexity reductions in online computation burden and required memory. It should be mentioned that the original function (3) and the replacement function (5) are definitely equivalent such that the two functions result in the same control action for the same initial conditions. For this reason, the stability and control performance remain unchanged.

C. Complexity Reduction Via Elimination Of The Saturated Regions

In the previous subsection, many regions with identical properties, corresponding to the saturated regions ($\overline{R}_{sat}$ *and* $\underline{R}_{sat}$), are removed in order to reduce the complexity. However, the number of remaining unsaturated regions (R_{unsat}) and its ratio to the total number of regions should be considered as an important issue to be investigated as well.

This subsection addresses the existence of symmetry in the unsaturated regions and their corresponding control actions. Then, the complexity reduction on which elimination of the symmetric regions has ultimate impacts are investigated.

Definition 1 [14]**:** The pair of invertible matrices (Θ, Ω) is a symmetry of the piecewise affine control law in (3) if there exists i and j such that:

$$\begin{aligned} \Omega F_i &= F_j \Theta \\ \Omega G_i &= G_j \\ \Theta R_i &= R_j . \end{aligned} \tag{7}$$

Therefore, such transformation matrices (Θ, Ω) map the i-th region R_i and the corresponding control law F_i and G_i, respectively, to the j-th region R_j and the corresponding F_j and G_j.

Definition 2 [14]: If there exists a pair of invertible matrices (Θ, Ω) such that (1) and (2) are preserved:

$$\begin{aligned}&\Theta^T Q_N \Theta = Q_N\\&\Theta^T Q\Theta = Q\\&\Omega^T R\Omega = R\\&\Theta A = A\Theta\\&\Theta B = B\Theta\end{aligned} \tag{8}$$

and the transformation does not violate the following constraints:

$$\begin{aligned}&x_{min} \le \Theta x_k \le x_{max}\\&u_{min} \le \Omega u_k \le u_{max}\\&y_{min} \le C\Theta x_k \le y_{max}\,.\end{aligned} \tag{9}$$

the set of all pairs (Θ, Ω) denoted by Aut(u^*) is called a group.

Definition 3 [21]: A permutation of a finite set S is a bijection from S to itself, i.e. bijection $\alpha : S \to S$.

To identify the controller symmetries, (9) is first transformed to the following form:

$$\begin{aligned}&G_x x \le 1\\&G_u u \le 1\end{aligned} \tag{10}$$

where G_x and G_u denote the normalized half-space matrices.

Theorem 2 [14]: Given the following equations which imply the transformation definition, have being introduced in (8):

$$\begin{aligned}&P_x G_x A\left(G_x{}^T G_x\right)^{-1} G_x{}^T = G_x A\left(G_x{}^T G_x\right)^{-1} G_x{}^T P_x\\&P_x G_x B\left(G_u{}^T G_u\right)^{-1} G_u{}^T = G_x B\left(G_x{}^T G_x\right)^{-1} G_u{}^T P_u\\&P_x G_x \left(G_x{}^T G_x\right)^{-1} Q_N \left(G_x\left(G_x{}^T G_x\right)^{-1}\right)^T\\&= G_x \left(G_x{}^T G_x\right)^{-1} Q_N \left(G_x\left(G_x{}^T G_x\right)^{-1}\right)^T P_x\\&P_x G_x \left(G_x{}^T G_x\right)^{-1} Q \left(G_x\left(G_x{}^T G_x\right)^{-1}\right)^T\\&= G_x \left(G_x{}^T G_x\right)^{-1} Q \left(G_x\left(G_x{}^T G_x\right)^{-1}\right)^T P_x\\&P_u G_u \left(G_u{}^T G_u\right)^{-1} R \left(G_u\left(G_u{}^T G_u\right)^{-1}\right)^T\\&= G_u \left(G_u{}^T G_u\right)^{-1} R \left(G_u\left(G_u{}^T G_u\right)^{-1}\right)^T P_u\end{aligned} \tag{11}$$

and if there exists the set of permutation matrices $\left\{\left(P_x^1, P_u^1\right),\ldots,\left(P_x^r, P_u^r\right)\right\}$ such that the relations in (11) are satisfied using $\left(P_x^i, P_u^i\right)$, $i = 1, \ldots, r$, the symmetry group $\left\{\left(\Theta_1, \Omega_1\right),\ldots,\left(\Theta_r, \Omega_r\right)\right\} \in$ Aut(MPC) can then be attained as follows:

$$\begin{aligned}&\Theta_i = \left(G_x{}^T G_x\right)^{-1} G_x{}^T P_x{}^i G_x\\&\Omega_i = \left(G_u{}^T G_u\right)^{-1} G_u{}^T P_u{}^i G_u\,.\end{aligned} \tag{12}$$

Proof: See [14].

The relations in (11), being taken as a graph automorphism problem, can be solved by the standard graph automorphism software packages [19] and identification of the symmetries can then be performed efficiently using (12). However, the explicit solution of (3) has almost more symmetries than the MPC problem in (2) [14]. Therefore, the following theorem can be considered for this purpose.

Theorem 3 [14]: Let $\left\{x_j\right\}_{j=1}^J$ be the extreme points of the partitions $\left\{R_i\right\}_{i=1}^{n_r}$. At the extreme points, $\left\{u^*{}_j\right\}_j^J$ is calculated. Assume that $\left\{u^*{}_j\right\}_j^J$ has at least a subset which is linearly independent. The linear transformation matrices $\left(\Theta, \Omega\right)$ is then obtained as follows:

$$\begin{aligned}&\Theta = XPX^T\left(XX^T\right)^{-1}\\&\Omega = UPU^T\left(UU^T\right)^{-1},\end{aligned} \tag{13}$$

where P satisfies the following equations:

$$\begin{aligned}&PX^T\left(XX^T\right)^{-1} X = X^T\left(XX^T\right)^{-1} XP\\&PU^T\left(UU^T\right)^{-1} U = U^T\left(UU^T\right)^{-1} UP\,,\end{aligned} \tag{14}$$

where $X = \left[x_1, \ldots, x_J\right] \in R^{n\times J}$ and $U = \left[u_1, \ldots, u_J\right] \in R^{m\times J}$. The set of matrices $\left\{\left(\Theta_1, \Omega_1\right),\ldots,\left(\Theta_r, \Omega_r\right)\right\}$ is assumed to be isomorphic to the set of all permutation matrices $P = \left\{P_1, \ldots, P_r\right\}$.

Proof: See [14].

The equation (14) can also be considered as a graph automorphism problem solved by the standard graph automorphism software packages. This will result in determination of the set of the transformation matrices $\{(\Theta_1, \Omega_1), \ldots, (\Theta_r, \Omega_r)\} \in \text{Aut}\left(\text{MPC}\right)$ given in (13). A procedure for identifying the symmetries of the MPC problem is organized as Algorithm 1 which summarizes the above steps.

Algorithm 1: Identifying the symmetries of MPC problem

```
1. Calculation of the set of all permutation matrices $P=\{P_1,\ldots,P_r\}$ using the standard graph automorphism software packages to solve either (14) or (11).
2. Calculation of the set of transformation pairs $\{(\Theta_1,\Omega_1),\ldots,(\Theta_r,\Omega_r)\}\in \mathrm{Aut}(\mathrm{MPC})$ using either (13) or (12).
```

On the other point, removing the saturated regions can significantly reduce the number of the extreme points and the subsequent computational complexity of (14). Then, the symmetry identification problem in (14) is simplified to only the extreme points corresponding to the unsaturated regions R_{unsat} which are usually less than that of $\{R_i\}_{i=1}^{n_r}$.

The property of symmetry implies that only one of i-th or j-th region and their corresponding control laws, as controller pieces of i, $j\in I$, is adequate to be stored and hence other regions and their corresponding control laws could be recovered using transformation matrices (Θ_i,Ω_i) in (7). On the basis of Definitions 2 and 3, permutation $\alpha: I\to I$, $I=\{1,\ldots,\mathrm{n}_r\}$, can be considered as a function which corresponds to the transformations being performed using the set of matrices $\{(\Theta_1,\Omega_1),\ldots,(\Theta_r,\Omega_r)\}$. Therefore, to define a function that is able to map each transformation pair of the set of matrices $\{(\Theta_1,\Omega_1),\ldots,(\Theta_r,\Omega_r)\}$ to the corresponding permutation of controller pieces, the following definition is given:

Definition 4 [21]: consider $\partial: Aut(\mathrm{MPC})\to I$ that maps each transformation pair of the set of matrices $\{(\Theta_1,\Omega_1),\ldots,(\Theta_r,\Omega_r)\}$ to corresponding permutation $\alpha=\partial(Aut(\mathrm{MPC})): I\to I$ of controller pieces I where $j=\alpha(i)$ stands for i and j satisfying (7). Therefore, in the view of symmetry concept, two pieces i, $j\in I$ are called equivalent if:

$\exists(\Theta,\Omega)\in f\subseteq Aut(\mathrm{MPC}): j=\alpha(i)$,

where controller piece i is mapped to controller piece $j=\alpha(i)$ for $\alpha\in\partial(f)$. The set of all controller pieces, equivalent to the i-th piece, is called a controller orbit [14], being described by:

$f(i)=\{\alpha(i):\alpha\in\partial(f)\}$.

Algorithm 2: Construction of controller orbit

```
1. Calculation of explicit solution of (3).
2. Identifying the symmetry group $f\subseteq Aut(\mathrm{MPC})$ of (3) using Algorithm 1.
3. Calculation of the controller orbits $f(i)$ of controller pieces for (3) by identifying the equivalent pieces.
4. Selecting only the representative controller pieces $(F_i,G_i,R_i)$, $i\in f(i)$ from each controller orbit $f(i)$.
```

Then, the controller pieces $I=\{1,\ldots,\mathrm{n}_r\}$ of the explicit solution of (3) are organized into the set of controller orbits $l=\{f(i_1),\ldots,f(i_t)\}$, where $\{i_1,\ldots,i_t\}$ includes one representative controller piece i_j from each orbit $f(i_j)$ and t denotes the number of the representative controller orbits belonging to (3). Therefore, implementation of the controller orbit requires to select one representative piece from each orbit, leading to recovering control law of j-th controller piece by identifying the corresponding controller orbit and subsequently transformation pair(Θ_i,Ω_i) of the orbit. According to (7), control law of j-th controller piece will be as follows:

$$u^*=\Omega_i F_i\Theta_i^{-1}x+\Omega_i G_i \qquad (15)$$

where controller pieces i, $j\in I$ create a controller orbit which its representative controller piece is the i-th controller piece. Similarly, in (5), R_{unsat} can also be classified into two constituent subsets of R_{Isymm} and R_{Jsymm} such that $R_{Isymm}\cup R_{Jsymm}=R_{unsat}$ where R_{Isymm} denotes the regions of representative controller pieces and R_{Jsymm} indicates the pieces which can be recovered by $(\Theta_i,\Omega_i)\in f\subseteq \mathrm{Aut}(u^*)$ of corresponding orbit i.e. if $x\in R_{Jsymm}$ then $\Theta_i x\in R_{Isymm}$ where $(\Theta_i,\Omega_i)\in f\subseteq \mathrm{Aut}(u^*)$. Therefore, only regions corresponding to R_{Isymm} are adequate to be stored. Algorithm 2 summarizes the process of creating orbit controller.

D. Complexity Reduction Via Elimination of the Symmetric and Saturated Regions

Removing saturated regions is not enough to be used as a general efficient approach to tackle the considered complexity reduction issue. Therefore, it is more efficient to accompany it with the symmetric approach, proposing

Algorithm 3: Searching through the regions R_{Isymm}

```
INPUT:
$\beta(x), R_{Isymm}, \{(\Theta_1,\Omega_1),\ldots,(\Theta_r,\Omega_r)\} \in \mathrm{Aut}(u^*), u_{max}$,
$u_{min}$, $F_i, G_i$ which correspond to $R_{Isymm}$
OUTPUT: $\tilde{u}^*(x)$

1:  if $x \in R_{Isymm=i}$
2:      return $i$
3:  elseif $\Theta_j x \in R_{Isymm=i}$
4:      return $i$ and $j$
5:      $u^*(x) = \Omega_j F_i \Theta_j^{-1} x + \Omega_j G_i$
6:  else
7:      if $\beta(x) > 0$
8:          $u^* = u_{\max}$
9:      else $\beta(x) < 0$
10:         $u^* = u_{\min}$
11:     end
12. end
```

a new integrated method due to tendency of many typical mathematical QP problems to exhibit large symmetry groups. On the other hand, a mere deployment of the symmetric approach is obviously not as effective as applying the developed integrated approaches for any candidate QP problem. Then, (5) can be rewritten as:

$$\tilde{u}^*_{symm}(x) = \begin{cases} F_i x + G_i & \text{if } x \in R_{Isymm} \\ \Omega_i F_i \Theta_i^{-1} x + \Omega_i G_i & \text{if } \Theta_i x \in R_{Isymm} \\ u_{max} & \text{if } x \notin R_{unsat}, \ \beta(x) > 0, \\ u_{min} & \text{if } x \notin R_{unsat}, \ \beta(x) < 0, \end{cases} \tag{16}$$

where the separator function is first applied to remove the saturated regions and then the unsaturated regions are classified into R_{Isymm} and R_{Jsymm} under the symmetry concept as performed for (3) in Algorithm 2, implying that only regions R_{Isymm} are needed to recover all of the unsaturated regions.

To find $\tilde{u}^*_{symm}(\mathrm{x})$, sequential search is first done through the regions of R_{Isymm}. Therefore if $x \in R_i \subset R_{\mathrm{Isymm}}$ then $\tilde{u}^*_{symm}(x) = F_i x + G_i$, if $\Theta_i\, x \in R_i \subset R_{\mathrm{Isymm}}$ then $\tilde{u}^*_{symm}(x) = \Omega_i F_i \Theta_i^{-1} x + \Omega_i G_i$, otherwise, the control input takes u_{max} or u_{min} based on sign of the separator function β. Based on the combined approaches introduced in Subsections 2.2, 2.3, and 2.4, Algorithm 3 is proposed to efficiently reduce the memory requirements and complexity for practical applications.

III. COMPLEXITY ANALYSIS

A. Offline computation

The preprocessing computation consists of two steps. The first step is to determine the closed-form solution of $\beta(x)$, discussed in [12]. The next step is then to identify $\{(\Theta_1,\Omega_1),\ldots,(\Theta_r,\Omega_r)\} \in \mathrm{Aut}(u^*)$ that is very difficult to give a closed-form solution of its preprocessing complexity. However, to obtain matrices $\{(\Theta_1,\Omega_1),\ldots,(\Theta_r,\Omega_r)\}$, it is required first to calculate the extreme points, i.e. $\{x_j\}_{j=1}^{J}$ and $\{u^*{}_j\}_j^J$ at these points, and then to determine $\{(\Theta_1,\Omega_1),\ldots,(\Theta_r,\Omega_r)\} \in \mathrm{Aut}(u^*)$ by the standard graph automorphism software packages. It should be noted that a large portion of the computation time to calculate the extreme points $x_j \in \bar{R}_{sat}$ and $\underline{R}_{sat}$ is already done to determine the separator function and thus the computationally demanding of $\mathrm{Aut}(u^*)$ can significantly be reduced.

B. Online Computation

In the worst case, finding a region that includes state x_0 consists of two steps. The first step is to find $(\Theta_i,\Omega_i) \in \mathrm{Aut}(u^*)$ that requires an order of $\mathcal{O}(rn\,|R_{unsat}|)$ time duration where r denotes the number of unique pair of invertible matrices $\{(\Theta_1,\Omega_1),\ldots,(\Theta_r,\Omega_r)\}$ regarding unsaturated controller pieces and $|R_{unsat}|$ is the sum of lengths of all unsaturated controller orbits, i.e. $|R_{unsat}| = \sum_{i=1}^{|R_{Isymm}|} |f(i)|$ where $|f(i)|$ is orbit size and $|R_{Isymm}|$ denotes the number of representative controller pieces of the unsaturated regions. The second step is to evaluate the separator function to find its sign which requires much less computational effort than the first step. Therefore, the online computation burden is approximately $\mathcal{O}(rn\,|\,R_{unsat}\,|)$ that is not considerably increased by the evaluation of the separator function. Indeed, removing many saturated regions comes at the negligible price of evaluation of polynomial $\beta(x)$.

C. Storage Requirement

According to Algorithm 3, the input data must be stored. Regarding to very low required memory to store $\beta(x)$, u_{max}, u_{min} only R_{Isymm} and $\text{Aut}(u^*)$ is investigated in this study that is far more than the other input data to be stored. To compare with the other approaches, analysis of F_i and G_i is neglected. Then, storing R_{Isymm} would require $\sum_{i=1}^{|R_{Isymm}|} h_i(n+1)$ real numbers (h_i is the number of half-spaces forming the i-th region). Also, to store $\{(\Theta_1,\Omega_1),\ldots,(\Theta_r,\Omega_r)\}\in\text{Aut}(u^*)$, the storage of $r(n^2+m^2)$ real number is required. Therefore, the total requirement storage is $r(n^2+m^2)+\sum_{i=1}^{|R_{Isymm}|} h_i(n+1)$. The proposed approach contributes to efficiently reduction of memory requirement since many regions are removed.

D. Comparison With the Other Algorithms

Many algorithms have been presented in the literature to reduce the complexity of an explicit MPC problem solution by reducing the number of regions to be stored, leading to the subsequent reduction in the required memory and online complexity computation. The function $u^*(x)$ in (3) has n_r regions to be stored and require at most $\mathcal{O}(n_r)$ floating point operations (FLOPS) to be computed online. In [22], M nodes with their associated pointers are stored and $\mathcal{O}(\log_2 n_r)$ FLOPS are required to be computed online. The approach based on the elimination of saturated regions scheme [12-13], however, requires storing $\sum_{i=1}^{|R_{unsat}|} h_i(n+1)$ real numbers and has time complexity of $\mathcal{O}(|R_{unsat}|)$ FLOPS. The online complexity and storage requirement of the approach being introduced in [14], are $\mathcal{O}(pnn_r)$ FLOPS and $\sum_{i=1}^{t} h_i(n+1)+p(n^2+m^2)$, respectively, where p denotes the number of unique pair of invertible matrices $\{(\Theta_1,\Omega_1),\ldots,(\Theta_p,\Omega_p)\}$ of all controller pieces which can be found in (3), respectively. As $|R_{Isymm}| \ll l,\ p \ll r$ and $|R_{Isymm}| \ll |R_{unsat}| < n_r$, therefore, although our proposed approach can reduce online complexity, the storage complexity is significantly reduced.

TABLE I
COMPLEXITY OF ALGORITHM 3 VS. OTHER APPROACHES (SYMMETRIC STATE-SPACE)

	Sequential search	Algorithm 3	Algorithm from [12]	Algorithm from [14]
FLOPS (worst case)	4474	628	574	4698
Storage (real numbers)	3357	219	423	1671

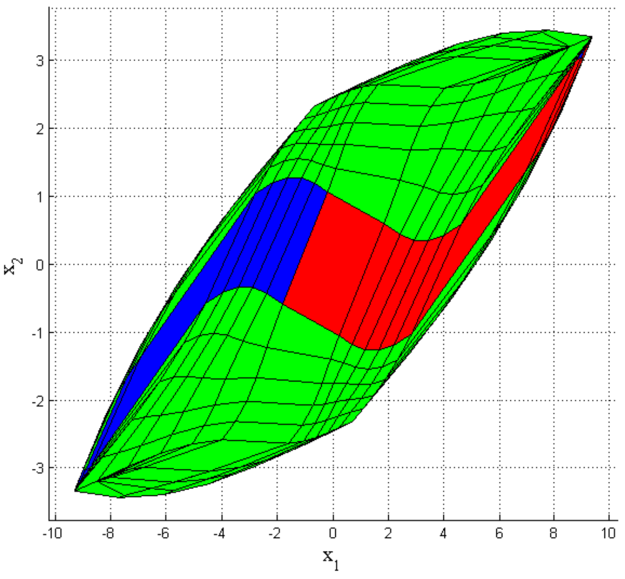

Fig. 1. Symmetric state space partition: $\overline{\boldsymbol{R}}_{sat}$, $\underline{\boldsymbol{R}}_{sat}$(green), $\boldsymbol{R}_{unsat}$(blue and red), $\boldsymbol{R}_{Jsymm}$ (blue), $\boldsymbol{R}_{Isymm}$(red)

IV. RESULTS

In this section, we evaluate the performance of the proposed algorithm.

A. Example 1

Consider the linear system [12]:

$$x(k+1)=\begin{bmatrix}0.755 & 0.680\\ 0.651 & -0.902\end{bmatrix}x(k)+\begin{bmatrix}0.825\\ -0.139\end{bmatrix}u(k)$$

subject to $|x_i|\le 10, i=1,2$ and $|u|\le 1$.

TABLE II
COMPLEXITY OF ALGORITHM 3 VS. OTHER APPROACHES (NON-SYMMETRIC STATE-SPACE)

	Sequential search	Algorithm 3	Algorithm from [12]	Algorithm from [14]
FLOPS (worst case)	3635	444	420	3771
Storage (real numbers)	2727	194	313	2061

The explicit solution (3) of QP problem (2) with $N = 10$, $N_c = 10$, $Q = Q_N = \mathrm{I}_{2\times 2}$ and $R = 1$ is obtained using the MPT Toolbox [23]. The explicit solution (4) consists of 225 regions where $|R_{unsat}| = 29$, $|\overline{R}_{sat}| = 98$ and $|\underline{R}_{sat}| = 98$. The separator function is $\beta(x) := -x_1 - x_2 - 0.0011x_1^3 - 0.254x_2^3$. The saturated regions $\underline{R}_{sat}$ and $\overline{R}_{sat}$ are removed using $\beta(x)$ and moreover many regions of R_{unsat} are removed using the symmetric property, being demonstrated graphically in Fig 1. Furthermore, to identify controller symmetries, it is sufficient to evaluate only the extreme points of the unsaturated regions. The symmetry group $\mathrm{Aut}(u^*)$ consists of reflection about origin of coordinate system as follows:

$$(\Theta_1, \Omega_1) = \left(\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, -1 \right).$$

According to algorithm 2, $|R_{unsat}| = 29$ pieces of the unsaturated regions are organized into $t = 15$ orbits: one orbit of size $|f(i)| = 1$ and 14 orbits of size $|f(i)| = 2$. Therefore, representative controller pieces R_{Isymm} are required to be stored which are depicted in Fig. 1.

Therefore, 14 regions of R_{unsat} have been removed that are redundant under the symmetry group $(\Theta_1, \Omega_1) = \mathrm{Aut}(u^*)$. Then, the required total memory is 1.75 kB, while the original function (3) requires 27 kB and hence the proposed approach results in a reduction of storage space by a factor of 15.2. In the worst case, the online complexity of the original function (3) and the integrated approach, introduced as algorithm 3, are, respectively, 4474 FLOPS and 628 FLOPS to attain $u^*(x)$. Table I presents the comparison between algorithm 3 and the counterpart methods in terms of the required storage and FLOPS in order to compute the control input.

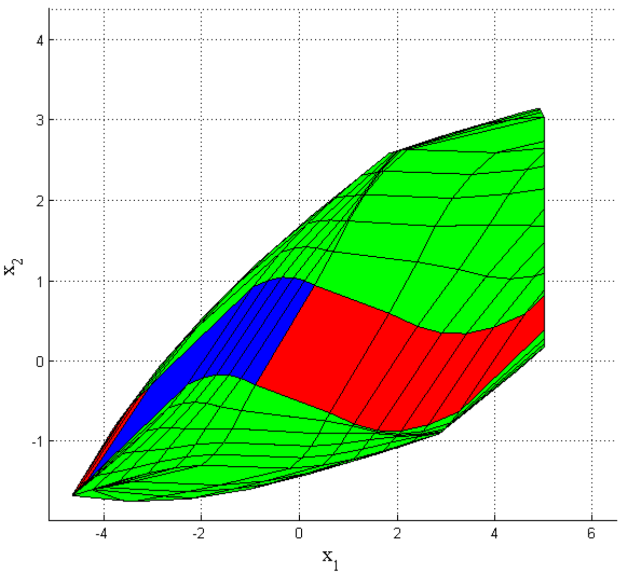

Fig. 2. Non-symmetric state space partition: polytopic sets $\overline{\boldsymbol{R}}_{sat}$, $\underline{\boldsymbol{R}}_{sat}$(green), $\boldsymbol{R}_{unsat}$(blue and red), $\boldsymbol{R}_{Jsymm}$ (blue), $\boldsymbol{R}_{Isymm}$(red)

Next, we assess the Algorithm 3 where the state and input constraints are not symmetric. The Algorithm 3 is again applied to the above linear system except for the constraints which are non-symmetric as $-1 \le u \le 0.5$ and $-10 \le x_i \le 5$. The explicit solution (3) is computed using MPT Toolbox [23]. The solution has a PWA function with 182 partitions classified by the separator function $\beta(x) := -x_1 - x_2 - 0.0011x_1^3 - 0.254x_2^3$ as $|R_{unsat}| = 21$, $|\overline{R}_{sat}| = 93$ and $|\underline{R}_{sat}| = 68$ shown in Fig 2. The symmetry group $\mathrm{Aut}(u^*)$ is as follows:

$$(\Theta_1, \Omega_1) = \left(\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, -1 \right).$$

According to algorithm 2, $|R_{unsat}| = 21$ pieces are resulted in $t = 16$ orbits: 11 orbits of size $|f(i)| = 1$ and 5 orbits of size $|f(i)| = 2$. Finally, $|R_{Isymm}| = 16$ regions, shown in Fig. 2, need to be stored.

As reported in Table II, the results show that the proposed algorithm outperforms the sequential algorithm and algorithms from [12] and [14] regarding processing time and memory requirements. Therefore, the Algorithm 3 remains efficient even where all constraints are not symmetric.

B. Example 2

In this section we apply the Algorithm 3 to the Cessna Citation 500 aircraft model presented in [20], [24]. It has the linearized continuous-time state space model as follows:

$$A = \begin{bmatrix} -1.2822 & 0 & 0.98 & 0 \\ 0 & 0 & 1 & 0 \\ -5.4293 & 0 & -1.8366 & 0 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} -0.3 \\ 0 \\ -17 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix},$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

The only input represents the elevator angle (rad), and the pitch angle (rad), altitude (m) and altitude rate (m/s) are outputs. Due to the equipment design limitations and passenger comfort, the elevator angle and the pitch angle are limited to $\pm 15^{\circ}$ $(\pm 0.262\ rad)$ and $\pm 20^{\circ} (\pm 0.349\ rad)$, respectively. Considering the actual condition for actuators, the constraint $|\Delta u| \leq 0.542$ is also included.

An MPC controller is intended with a sampling interval of $0.5s$, the horizons of $N = 10$ and $N_c = 3$. (The eq.2 and others shall be edited) Our objective is to track the reference $[0; 400m; 0]$ for the outputs.

The QP problem (2.39) with $Q = I_{3\times 3}$ and $R = 1$ is explicitly solved using MPT Toolbox [23]. The explicit solution (3.7) consists of a PWA function with 117 regions where $|R_{unsat}| = 80$, $|\overline{R}_{sat}| = 19$ and $|\underline{R}_{sat}| = 18$. The separator function

$$\beta(x) := [-42.74 * 10^4, 10^6, 26.57 * 10^4, -3.6 * 10^{-11}, 1.67 * 10^{-9}] * x + 1.309 * 10^{-8}$$

is obtained to separate the regions over which control input attains a saturated value. Then, the sign of the separator function $\beta(x)$ is used so that the evaluation of the saturated regions is no longer required to obtain the control action. Therefore, there is no need to save the saturated regions, and the regions can be eliminated.

Next, the extreme points of the unsaturated regions are evaluated to find symmetries in the regions. The evaluation leads to the symmetry group $Aut(u^*)$ consisting of a reflection about origin of coordinate system as follows:

$$(\Theta_1, \Omega_1) = \left(\begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}, 1 \right).$$

Due to the reflection, 40 regions of R_{unsat} are removed because the regions can be recovered by the symmetry

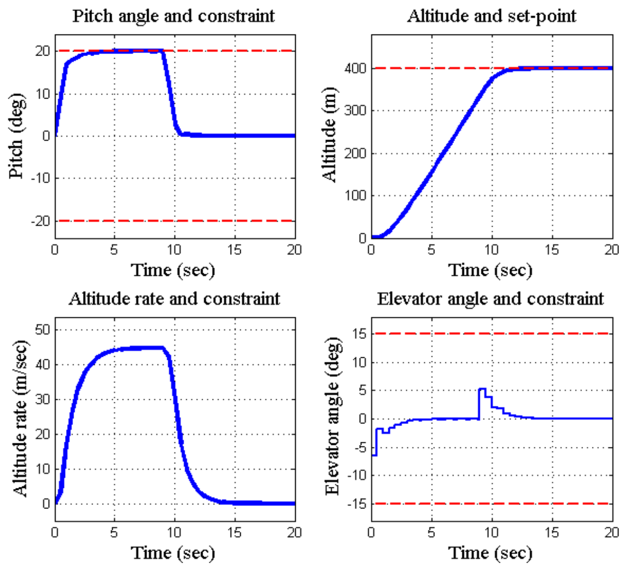

Fig. 3. Responses to 400m step change in altitude set-point.

group Aut(u^*). The comparison between the proposed approach, sequential search method and the algorithms from [12] and [14] are reported in Table III in terms of memory footprint and worst-case computation effort (FLOPS). The table shows that the proposed method needs much less resources to obtain the control actions.

Fig. 3 shows the responses to the output set point of $[0; 400m; 0]$ processed by the Algorithm 3 and the traditional explicit MPC in (3). As it can be seen, the responses are completely identical such that the simplification not only ensures the system remains stable, but it also leads to no changes on the set-point tracking performance. Moreover, Algorithm 3 allows an inexpensive implementation for the high-order system.

The online computations to track the output reference was performed using Matlab 8.3 on a 2.4 GHz Core i5. Table IV presents the comparison between Algorithm 3, the original function and the algorithm PDIP proposed in [20], [24] in terms of average worst case CPU time per sample.

V. CONCLUSION

An efficient integrated approach has been presented in this paper to reduce storage requirement and solution complexity of the QP problems in explicit MPC implementation for embedded applications based on offline and online computations, utilizing two main steps. The first step is elimination of the saturated regions which attain saturated value using a simpler replacement function. In the second step, the redundant symmetric regions are efficiently identified and removed to simplify the inherent complexity on the basis of QP self-similarity patterns. The proposed approach hence combines the interesting merits of both individual schemes to achieve a simpler control law for which its implementation needs less hardware resources than of the original function. It

TABLE III
ONLINE COMPUTATION AND MEMORY COMPLEXITY OF ALGORITHM 3 VS. OTHER APPROACHES

	Sequential search	Algorithm 3	Algorithm from [12]	Algorithm from [14]
FLOPS (worst case)	8925	6374	6174	9515
Storage (real numbers)	8222	2832	5655	4116

TABLE IV
ONLINE COMPLEXITY OF ALGORITHM 3 VS. OTHER APPROACHES

	Sequential search	Algorithm 3	Algorithm PDIP from [20], [24]
Worst case CPU time	4.5 ms	3.5 ms	13.5 ms

was observed that the storage requirement can be reduced by a factor of the total number of regions over the number of regions which are unsaturated and non-redundant. Finally, the effectiveness of the proposed method was experimentally supported by comparative conducted tests using alternative approaches.

REFERENCES

[1] M. Ławryńczuk. “Introduction to Model Predictive Control” In: Nonlinear Predictive Control Using Wiener Models. Studies in Systems, Decision and Control, 1st ed, vol 389. Springer, Cham, 2022, pp. 3-40.

[2] R. Oberdieck, N.A. Diangelakis, EN Pistikopoulos. "Explicit model predictive control: a connected-graph approach." *Automatica*, vol. 76, pp. 103-112, Feb. 2017.

[3] A. Bemporad. "A multiparametric quadratic programming algorithm with polyhedral computations based on nonnegative least squares." *IEEE Transactions on Automatic Control*, vol. 60, pp. 2892-2903, Mar. 2015.

[4] R. Sheikhbahaei, A. Alasty, and G.Vossoughi. “Robust fault tolerant explicit model predictive control.” Automatica, vol. 97, pp.248-253, Nov. 2018.

[5] A. Bemporad, M. Morari, V. Dua, E.N. Pistikopoulos. "The explicit linear quadratic regulator for constrained systems." *Automatica*, vol. 38.1, pp. 3-20, Jan. 2002.

[6] S.W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari. “Large scale model predictive control with neural networks and primal active sets.” Automatica, vol. 135, p.109947, Jan. 2022.

[7] Karg, Benjamin, and Sergio Lucia. "Efficient representation and approximation of model predictive control laws via deep learning." IEEE Transactions on Cybernetics 50.9 (2020): 3866-3878.

[8] A. Bemporad, A. Oliveri, T. Poggi, M. Storace. "Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations." *IEEE Transactions on Automatic Control*, vol. 12, pp. 2883-2897, Apr. 2011.

[9] A. Alessio, A. Bemporad. "A survey on explicit model predictive control." *Nonlinear model predictive control*. Springer Berlin Heidelberg, pp. 345-369, 2009.

[10] M. Schwenzer, M. Ay, T. Bergs, D. Abel. “Review on model predictive control: an engineering perspective.” The International Journal of Advanced Manufacturing Technology, vol. 117(5), pp. 1327-49, Nov. 2021.

[11] F.J. Christophersen, M.N. Zeilinger, C. N. Jones, M. Morari. "Controller complexity reduction for piecewise affine systems through safe region elimination." *Decision and Control. 46th IEEE Conference on*. IEEE. 2007, pp. 4773-4778.

[12] M. Kvasnica, J. Hledík, I. Rauová, M. Fikar. "Complexity reduction of explicit model predictive control via separation." *Automatica*, vol. 6, pp. 1776-1781, Jun. 2013.

[13] M. Kvasnica, M. Fikar. "Clipping-based complexity reduction in explicit MPC." *IEEE Transactions on Automatic Control*, vol. 57.7, pp. 1878-1883, Dec. 2011.

[14] C. Danielson, F. Borrelli. "Symmetric linear model predictive control." *IEEE Transactions on Automatic Control*, vol. 60.5, pp. 1244-1259, Nov. 2014.

[15] C. Danielson. "An alternating direction method of multipliers algorithm for symmetric model predictive control." Optimal Control Applications and Methods vol. 42.1, pp. 236-260, Jan. 2021.

[16] D. Bremner, M.D. Sikiric, A Schürmann. "Polyhedral representation conversion up to symmetries." *CRM proceedings*. Vol. 48, pp. 45-72, 2009.

[17] P.T. Darga, K.A. Sakallah, I.L. Markov. "Faster symmetry discovery using sparsity of symmetries." *Design Automation Conference, DAC. 45th ACM/IEEE*. IEEE. 2008, pp. 149-154.

[18] T. Junttila, P. Kaski. "Engineering an efficient canonical labeling tool for large and sparse graphs." *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*. Society for Industrial and Applied Mathematics, 2007, pp. 135-149.

[19] B.D. McKay. "Practical graph isomorphism." 1981, pp. 45-87.

[20] A. Del Rio Ruiz, and K. Basterretxea. "Towards the automatic implementation of reduced-size and high throughput mpc on fpgas," 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT). IEEE, 2019, pp. 1768-1773.

[21] D.F. Holt, B. Eick, E.A. O'Brien. "Handbook of computational group theory (Discrete Mathematics and Its Applications)".Chapman and Hall/CRC Press, 2005.

[22] P. Tøndel, T.A. Johansen, A. Bemporad. "Evaluation of piecewise affine control via binary search tree." *Automatica*, vol. 39.5, pp. 945-950, May. 2003.

[23] M. Herceg, M. Kvasnica, C.N. Jones, M. Morari. “Multi-parametric toolbox 3.0." *European control conference (ECC)*, IEEE, 2013, pp. 502-510.

[24] K.V. Ling, S.P. Yue, J.M.Maciejowski. “A FPGA implementation of model predictive control.” In American control conference, Minnesota Minneapolis, 2006, p. 6.