A software package for MPC design and tuning: MPT+

Juraj Holaza, Lenka Galčíková, Juraj Oravec, Michal Kvasnica

***Abstract*— The industrial implementation of the model predictive control (MPC) is driven by the necessity to design, tune, and validate the constructed control policy. We present a novel software toolbox for advanced model predictive control (MPC) design extending the `Multi-Parametric toolbox` (`MPT`). Particularly, `MPTplus` introduces several advanced MPC controller design methods, including memory-efficient explicit tube MPC design, tube MPC controller with the limited rate of control actions, and a polynomial approximation of the $1-$, $\infty-$norm-based explicit control law. The benefits of the proposed toolbox are demonstrated using both, numerical simulations and laboratory implementation on the device with fast dynamics.**

I. Introduction

Although model predictive control (MPC) [1], [2] provides an optimal control policy under various physical and performance constraints, its industrial implementation is still under significant relevance [3]. Introducing multi-parametric programming into MPC design framework [4] gained its potential by reducing the runtimes and requirements on the advanced software, pushed the MPC implementation towards the industrial embedded hardware, see [5]. Moreover, the construction of the explicit solution map in the form of the piece-wise affine (PWA) function enables rigorous certification of the control law. Although MPC has been intensively studied in the past three decades, there are still challenges worth addressing to spread its industrial implementation. Such a development is highly dependent on tailored validation tools. The MPC design problem is addressed in plenty of well-developed software tools. Most of the packages provide dedicated built-in solvers, and the other delegate the optimization problem to third-party solvers.

Following is a brief review of the MPC design software limited just for the tools offering an interface for the `MATLAB`[1] programming environment. We refer the reader to [6], [7], [8], and the references therein, for the review on the tailored solvers and tools supporting MPC design and its evaluation using also other programming environments, e.g., `Python`[2], `Julia` [3], etc. Such packages include QP-solvers `qpDUNES` [9], `QPALM` [10], `CVXGEN` [11], and tools that introduce distributed optimization `OSQP` [12], `ALADIN-`α [13], to name a few.

Commercial `MPC Toolbox` [14] for `MATLAB` addresses various classes of the MPC design problems, including the construction of the adaptive, explicit, gain-scheduled, and non-linear MPC controllers. This toolbox provides several built-in solvers and also a dedicated user interface `MPC Designer App` for the controller tuning. The non-linear MPC design problems are efficiently solved using the `ACADO Toolkit` [15]. The non-convex optimization problem is solved by the sequential quadratic programming (SQP) approach. `ACADO Toolkit` evaluates the library-dependent `C/C++` code and provides an interface for `MATLAB`. `CasADi` [16] represents another toolbox suitable for the non-linear MPC design. This open-source package also provides interfaces for `MATLAB` and `Python`. The non-linear MPC can be designed also using the open-source toolbox `MATMPC` [17]. `YALMIP` [18] toolbox is a widely-used modeling parser focused on the various classes of optimization problems, including non-convex optimization. `YALMIP` provides also support for the MPC design problems. Similar to `YALMIP` toolbox, the `CVX` [19] is focused on the modeling of various classes of convex optimization problems suitable for MPC controller design.

The `Multi-Parametric Toolbox` (`MPT`) [20] for `MATLAB` represents a widely-used software package for the implicit (non-explicit) and explicit MPC design, multi-parametric optimization, and operations over convex sets. `MPT` integrates many tools enabling efficient construction, tuning, and validation of the advanced MPC controllers.

The industrial implementation of the MPC is limited by the runtime effort and advanced hardware/software requirements, especially in the case of the implicit MPC, and memory consumption, limiting the implementation of explicit MPC on the industrial embedded hardware. Therefore, the complexity reduction methods are of high interest. `MPT` module `LowCom` [21] introduces the set of methods reducing the complexity of the explicit solution maps associated with the MPC controllers to minimize the runtimes and memory footprints. In this framework, the polynomial-based approximation of the explicit solution maps introduced in [22] represents a perspective technique to minimize the memory footprint, however, it is not widely available in some user-friendly way, yet.

In both, implicit and explicit MPC design problems, the tube MPC represents a natural step toward the stochastic and non-linear MPC design. Finally, due to the ability to handle the quantized control variables, tube MPC is also a necessary tool in a highly relevant field of the encrypted MPC design using the cloud-computing services, e.g., see [23] and the related works.

Authors are with Faculty of Chemical and Food Technology, Slovak University of Technology in Bratislava, 812 37 Bratislava, Slovakia, `juraj.holaza@stuba.sk`

[1] `MATLAB`, MathWorks, Inc.: `https://www.mathworks.com`

[2] `Python`, Python Software Foundation: `https://www.python.org`

[3] `Julia`, JuliaLang.org: `https://julialang.org`

This paper presents the `MATLAB` toolbox `MPTplus` [24] augmenting the features of the original `MPT` toolbox by the notable methods introducing: (i) the framework for the memory-efficient explicit tube MPC design, (ii) limits on a rate of control actions for implicit/explicit tube MPC design, and (iii) the polynomial approximation of any explicit MPC controller according to [22]. To demonstrate the benefits of the proposed `MPTplus` for real-world applications, we analyze the results of an extensive case study considering the implementation of designed MPC controllers on the laboratory device having a fast-dynamics called Flexy2 [25].

II. Theoretical backgrounds

This section briefly reviews the main theoretical backgrounds of the currently implemented methods: (i) original (rigid) tube MPC design approach proposed in [26] and its formulation considering multi-parametric optimization, (ii) limited rates on the control actions, and (iii) polynomial approximation of explicit control law.

A. Tube MPC design

Given an uncertain linear time-invariant (LTI) system defined as follows:

$$x(t+T_{\mathrm{s}}) = Ax(t) + Bu(t) + Ed(t), \tag{1}$$

where t is the time sample of the discrete-time domain defined using a sampling time T_{s}. $A \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{x}}}$ is system matrix, $B \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{u}}}$ is input matrix, such that the matrix pair (A, B) is stabilizable. $E \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{d}}}$ is disturbance matrix, $x \in \mathbb{R}^{n_{\mathrm{x}}}$ is the vector of the system states, $u \in \mathbb{R}^{n_{\mathrm{u}}}$ is control action, $d \in \mathbb{D} \subset \mathbb{R}^{n_{\mathrm{d}}}$ is bounded additive disturbance such that $\mathbb{D}$ is compact set containing the origin. For the disturbance in (1) holds:

$$w = E\,d,\ w \in \mathbb{W},\ \mathbb{W} = \{w \in \mathbb{R}^{n_{\mathrm{x}}} : \|w\|_\infty \leq w_{\max}\}, \tag{2}$$

where $w_{\max} = \|E\,d\|_\infty$ is an upper bound for $\forall d \in \mathbb{D}$ satisfying $\mathbb{W} \supseteq E\,\mathbb{D}$ is the minimum volume hyper-box such that $\|w\|_\infty = w_{\max}$.

Consider, the uncertain LTI system in (1) is constrained by

$$x(t) \in \mathbb{X}, \quad u(t) \in \mathbb{U}, \quad \forall t \geq 0, \tag{3}$$

where $\mathbb{X} \in \mathbb{R}^{n_{\mathrm{x}}}$ and $\mathbb{U} \in \mathbb{R}^{n_{\mathrm{u}}}$ are polytopes containing origin in their strict interiors.

The convex set denoted as a "tube" $\mathbb{T} \subset \mathbb{R}^{n_{\mathrm{x}}}$ is constructed as an invariant approximation of the minimal robust positively invariant set using an algorithm described in [27].

The conventional (rigid) tube MPC design problem has the form [26]:

$$\min_{\hat{u}_0,\ldots,\hat{u}_{N-1},\hat{x}_0,\ldots,\hat{x}_N} \ \|\hat{x}_N\|_P^2 + \sum_{k=0}^{N-1} \left(\|\hat{x}_k\|_Q^2 + \|\hat{u}_k\|_R^2\right) \tag{4a}$$

$$\text{s.t.:} \quad x(t) - \hat{x}_0 \in \mathbb{T}, \tag{4b}$$

$$\hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k, \tag{4c}$$

$$\hat{x}_k \in \mathbb{X} \ominus \mathbb{T}, \tag{4d}$$

$$\hat{x}_N \in \mathbb{X}_{\mathrm{N}}, \tag{4e}$$

$$\hat{u}_k \in \mathbb{U} \ominus K\,\mathbb{T}, \tag{4f}$$

for $\forall k = 0, \ldots, N-1$, $\forall l = 1, \ldots, N-1$, and for prediction horizon N. Vectors $\hat{u}_k$ and $\hat{x}_k$ are decision variables optimized w.r.t. the nominal LTI system in (4c), without any perturbations of the disturbance w. The squared $2-$norm objective function in (4a) is minimized for the symmetric positive definite penalty factors $Q \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{x}}}$, $R \in \mathbb{R}^{n_{\mathrm{u}} \times n_{\mathrm{u}}}$, $P \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{x}}}$. K represents the LQR gain. The stability and recursive feasibility of the tube MPC design problem in (4) has been proved in [26].

Finally, the control action $u(t)$ to be implemented to the uncertain LTI system in (1) is given by the control law $\kappa : \mathbb{X}_{\mathrm{F}} \to \mathbb{U}$

$$\kappa(x(t)) = \hat{u}_0^\star + K\left(x(t) - \hat{x}_0^\star\right), \tag{5}$$

where symbol $\star$ in the optimal solution of the MPC design problem in (4) and the feasibility set $\mathbb{X}_{\mathrm{F}} \subseteq \mathbb{R}^{n_{\mathrm{x}}}$ of the optimized initial condition $\hat{x}_0$ of (4) represents its domain.

The actuators of the physical systems are often under limited rates on the control actions formulated by:

$$\Delta u(t) = u(t) - u(t_-) \in \mathbb{U}_\Delta, \tag{6}$$

where $\Delta u(t) = u(t) - u(t_-)$ is the rate of control action and $\mathbb{U}_\Delta \in \mathbb{R}^{n_{\mathrm{u}}}$ is the corresponding convex set bounding the rates. Then, the tube MPC design problem in (4) is extended by the linear constraint having the form:

$$\hat{u}_k + K(x(t) - \hat{x}_0) - u(t_-) \in \mathbb{U}_\Delta, \tag{7}$$

leading to the augmented vector of parameters $[x(t)^\top, u(t_-)^\top]^\top \in \mathbb{R}^{n_{\mathrm{x}}+n_{\mathrm{u}}}$, in case of the multi-parametric solution of (4).

B. Explicit tube MPC

The main benefit of explicit MPC is its ability to significantly speed up the real-time evaluation of the optimal control action by pre-computing the optimization problem in advance, see [4]. Simultaneously, the explicit solution provides the possibility for rigorous analysis, verification, and certification of the designed control law.

The QP in (4) can be reformulated into the explicit MPC design framework by evaluating the initial condition in (4b) exploring the set of admissible initial conditions: $\Theta = \mathbb{X} \ominus \mathbb{T}$. Then, the solution of corresponding multi-parametric QP (mpQP) returns also the convex set

of feasible system states $\mathbb{X}_\mathrm{F} \subseteq \Theta$ defined as the explicit solution map:

$$\mathbb{X}_\mathrm{F} = \bigcup_{i}^{n_\mathrm{r}} \mathcal{P}_i, \tag{8}$$

where $\mathcal{P}_i$ are polytopes, i.e., critical regions, and n_r is the total number of critical regions.

As the consequence, for $x(t) \in \mathcal{P}_i$ denoting ith critical region, holds control law in (5) given in a form of piece-wise affine (PWA) functions corresponding to the particular decision variables:

$$\hat{x}^\star(x(t)) = F_{\mathrm{x},i}\, x(t) + g_{\mathrm{x},i} \quad \text{if} \quad x(t) \in \mathcal{P}_i, \tag{9a}$$
$$\hat{u}^\star(x(t)) = F_{\mathrm{u},i}\, x(t) + g_{\mathrm{u},i} \quad \text{if} \quad x(t) \in \mathcal{P}_i, \tag{9b}$$

where $F_{\mathrm{x},i}$, $F_{\mathrm{u},i}$, $g_{\mathrm{x},i}$, $g_{\mathrm{u},i}$ have appropriate dimensions. For further technical details see [4]. Analogous to [28], we re-formulate the control law of the explicit tube MPC in (9) into the compact form:

$$\kappa(x(k)) = \underbrace{(K - KF_{\mathrm{x},i} + F_{\mathrm{u},i})}_{F_i} x(k) + \underbrace{(g_{\mathrm{u},i} - Kg_{\mathrm{x},i})}_{g_i}. \tag{10}$$

We point out, that such a compact formulation significantly reduces the associated memory footprint necessary to store the PWA functions compared to the conventional explicit formulation in (9). To be specific, the feedback law in (10) saves $3\,n_\mathrm{x}$ double precision numbers per each critical region compared to the original form (9a).

C. Polynomial approximation of explicit MPC

This section briefly recalls the theoretical backgrounds of the polynomial approximation of explicit MPC [29]. This approach provides a technique suitable for control applications where low computational effort is necessary while stabilizing the system and satisfying the constraints. On the other hand, the polynomial approximation of the control law is suboptimal.

Let us consider the MPC optimization problem in the following form:

$$\min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \left(\|x_k\|_Q^p + \|u_k\|_R^p \right) \tag{11a}$$
$$\text{s.t.:} \quad x_0 = x(t), \tag{11b}$$
$$x_{k+1} = Ax_k + Bu_k, \tag{11c}$$
$$x_k \in \mathbb{X}, \tag{11d}$$
$$u_k \in \mathbb{U}, \tag{11e}$$

where the properties of the parameters are adopted from the optimization problem in (4). The norm is considered as $p \in \{1, \infty\}$, i.e., the problem in (11) can be formulated as a Linear Program (LP). Then, by multi-parametric solving for all feasible $x_0 \in \mathbb{X}_\mathrm{F}$, the PWA explicit feedback control law is obtained in the form:

$$\kappa(x(t)) = F_i\, x(t) + g_i \quad \text{if} \quad x(t) \in \mathcal{P}_i. \tag{12}$$

The aim of this complexity reduction approach is to approximate the PWA control law in (12) with a polynomial:

$$\widetilde{\kappa}(x(t)) = a_1\, x(t) + a_2 x(t)^2 + \cdots + a_d x(t)^n, \tag{13}$$

where $a = \{a_1, \ldots, a_d\}$ are the coefficients of the polynomial, and n is the polynomial degree which is fixed and given in advance by the control engineer. Note, the polynomial degree n directly dictates the memory footprint of the approximated control law in (13) and at the same time affects the performance loss. Therefore, n is the main tuning parameter of the method that trades off the performance and the complexity reduction of (13).

Let us presume that there exists a PWA Lyapunov function L for the considered closed-loop system f_CL, i.e. controlled system (1) subject to the control law in (12), such that $L(f_\mathrm{CL}) \leq \gamma L(x)$, $L(0) = 0$ for $\gamma \in [0, 1)$ and x is within the feasible set $\mathbb{X}_\mathrm{F}$, i.e., $x \in \mathbb{X}_\mathrm{F}$. Considering the controlled LTI system, Lyapunov function L and fixed γ, the stabilizing "tube" can be constructed according to [30]:

$$\mathcal{T}(L, \gamma) := \left\{ \begin{matrix} [x^\top, u^\top]^\top \,:\, u \in \mathbb{U}, x \in \mathbb{X}_\mathrm{F}, Ax + Bu \in \mathbb{X}_\mathrm{F} \\ L(f_\mathrm{CL}) \leq \gamma L(x) \end{matrix} \right\},$$

consisting of $i = 1, \ldots, n_\mathrm{r}$ polytopes:

$$\mathcal{T}_i := \left\{ [x^\top, u^\top]^\top \,:\, [T_i^\mathrm{x}, T_i^\mathrm{u}][x^\top, u^\top]^\top \leq T_i^0 \right\}. \tag{14}$$

If such a stabilizing Lyapunov function exists and a stability "tube" is constructed, then the polynomial coefficients a are obtained by solving a Polya-filter-based linear program minimizing the suboptimality level of (13) from its optimal counterpart (12), i.e., $\min(\|\widetilde{\kappa}(x(t)) - \kappa(x(t))\|^p)$. For a more detailed overview of the approximation procedure see [29].

III. The software package MPT+

The `MPTplus` toolbox extends the functionality of the `MPT` toolbox [20]. In this section, we introduce the novel features, including (i) efficient explicit tube MPC controller construction, (ii) introducing incremental constraints into the tube MPC design, and (iii) the general polynomial approximation of the PWA control law.

A. Installation

The `MPTplus` toolbox is freely available on `GitHub`[4]. We recommend to install the package via `tbxManager`[5] by typing:

```
tbxmanager install mptplus
```

Alternatively, install the `MPTplus` by setting the corresponding path in `MATLAB`.

[4] `MPTplus: https://github.com/holaza/mptplus`
[5] `tbxManager: https://www.tbxmanager.com`

B. Explicit tube MPC design

In this section, we will show how one can easily derive an explicit tube MPC policy. For demonstrative reasons, we will assume a double integrator model adopted from [26]:

$$x(t+T_{\mathrm{s}}) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} d(t), \quad (15)$$

that is subjected to state and control input constraints as follows:

$$[-200, -2]^\top \preceq x(t) \preceq [200, 2]^\top, \quad (16a)$$

$$-1 \leq u(t) \leq 1, \quad (16b)$$

and the disturbance is constrained in a following way:

$$[-0.1, -0.1]^\top \preceq x(t) \preceq [0.1, 0.1]^\top. \quad (17)$$

First, assume the implicit tube MPC design. We defined "`model`", i.e., the object that contains prediction model (15) as in (4c), constraints (16) transformed into (4d)-(4f), penalty matrices, the appertain LQR terminal penalty and terminal set as in (4a) and (4e), respectively. Finally, using the `MPTplus`, we may simply construct and evaluate the implicit tube MPC controller by typing:

```
model=ULTISystem('A', [1, 1; 0, 1],...
'B', [0.5; 1],...
'E', [1, 0; 0, 1])
model.d.min = [-0.1; -0.1]
model.d.max = [ 0.1;  0.1]
model.x.min = [-200; -2]
model.x.max = [ 200;  2]
model.u.min = [-1]
model.u.max = [ 1]
% Penalty functions
model.x.penalty=QuadFunction([1,0;0,1])
model.u.penalty=QuadFunction(0.01)
% Prediction horizon
N = 9;
option={'solType',1,'LQRstability',1}
TMPC = TMPCController(model,N,option)
% TMPC evaluation
x0 = [-5; 2]
u = TMPC.evaluate(x0)
```

leading to the optimal control action $u_0^\star = 1$ for the given initial state $x(t) = [-5, 2]^\top$. It is worth noting that the terminal set and terminal penalty are enforced through the options parameter `LQRstability`, while the second parameter `solType` $= 1$ specifies the type of the returned optimized variables to be as in (5), i.e., that are fed directly to the controlled system in (1).

To design the associated explicit MPC policy we can evoke:

```
ETMPC = TMPC.toExplicit
```

what constructs the explicit feedback law `ETMPC` as in (10) that is defined over $n_{\mathrm{r}} = 520$ critical regions in $n_{\mathrm{x}} = 2$ dimensional space.

The evaluation of the exact decision variables $\hat{u}_0^\star = 0.7026$, $\hat{x}_0^\star = [-5.0516, -1.7500]^\top$ in (5), i.e., the optimizers $\left[\hat{u}_0^{\star\top}, \hat{x}_0^{\star\top}\right]^\top$ in (4), we need to change the `solType` parameter to zero and revoke the MPC syntheses via:

```
option={'LQRstability',1, 'solType',0}
TMPC = TMPCController(model,N,option)
ux = TMPC.optimizer(x0)
```

The resulting MPC policy `TMPC` preserves all of the computed/employed information inside of its structure. To respectively access and depict them, if possible, $K = [-0.6609, -1.3261]$ in (5), $\mathbb{T}$ employed in (4), $\mathbb{X} \ominus \mathbb{T}$ in (4d), $\mathbb{U} \ominus K\,\mathbb{T}$ in (4e) use:

```
K = TMPC.TMPCparams.K
Tube = TMPC.TMPCparams.Tube
figure, Tube.plot()
XT = TMPC.TMPCparams.Xset
figure, XT.plot()
UKT = TMPC.TMPCparams.Uset
figure, UKT.plot()
```

To graphically analyze our explicit controller, we can type:

```
figure; ETMPC.partition.plot()
figure; ETMPC.feedback.fplot()
figure; ETMPC.cost.fplot()
```

to plot the polytopic partition (8), the PWA feedback law (10), and the PWQ value function as defined in (4a), respectively.[6]

To perform a closed-loop simulation of a computed explicit tube MPC controller one can define an object:

```
loop = ClosedLoop(ETMPC,model)
```

where `model` can contain even a modified version of (15), i.e., the controlled system can be different from the prediction model used to design the explicit policy `ETMPC`. Subsequently, we can execute the closed-loop simulation by:

```
Nsim = 10
data = loop.simulate(x0, Nsim)
```

with `x0` denoting the initial condition and `Nsim` number of the closed-loop simulation steps. The generated `data` structure contains all important information, e.g., the closed-loop profiles of control inputs `data.U`, applied disturbances `data.D`, states `data.X`, and the cost of (4a) at each time step, to name a few. Alternatively, one can create a customized loop, where the control input of `ETMPC` can be obtained by typing:

[6]We note that while both, the PWA feedback law and the PWQ value function, are given as solution to (4) the solution (9) was posterior transformed into (10), however, the value function preserves its form of (4a).

```
u = ETMPC.evaluate(x)
```

for any given feasible state vector $x \in \mathbb{X}_F$.

The complexity reduction using the polynomial approximation significantly reduces the memory footprint by replacing the whole explicit solution map by coefficients of the polynomials [22]. The real-time runtimes are also decreased as the point location problem is replaced by the polynomial evaluation, see Section II-C. This approach is called, depicted, and evaluated by:

```
EMPCsim = PolynomialMPC(EMPC,options)
EMPCsim.plot
u = EMPCsim.evaluate(x0)
```

where parameter `options` provides the user additional options for the approximation such as the degree of the polynomial n, decay of the Lyapunov function γ, and many more.[7]

Finally, if only a pure visual analysis of the closed-loop simulation is required then we encourage users to use

```
ETMPC.clicksim()
```

where initial state conditions are defined by the mouse.

C. Tube MPC design for rate constraints

Another contribution of this paper is the inclusion of the so-called rate constraints that are defined in (6).

Implementation of these constraints in `MPTplus` is straightforward. User has to include

```
model.u.with('deltaMin')
model.u.with('deltaMax')
model.u.deltaMin = -0.5
model.u.deltaMax =  0.5
```

to the original LTI system definition, see section III-B. Then we can evoke the construction of the implicit tube MPC policy via:

```
N = 2
option={'solType',1,'LQRstability',1}
TMPC = TMPCController(model,N,option)
```

to design `TMPC` instance that considers (6) in (4).

Subsequently, the construction of the explicit MPC feedback law can be triggered by typing:

```
ETMPC = TMPC.toExplicit;
```

that, for our specific case considered in this section, is defined over a polyhedral partition consisting of 520 critical regions. Finally, evaluation of these controllers for an initial state $x(t) = [-1, 1]^\top$ can be done by executing following commands:

```
x0 = [1; -1]
u_prev = 0;
[u_0, feasible] = TMPC.evaluate(x0,...
'u.previous', u_prev)
[u_0, feasible] = ETMPC.evaluate(x0,...
'u.previous', u_prev)
```

that lead to the same closed-loop control action $u_0 = [0.3678]$. Notice that the evaluation requires an additional parameter `u_prev`, i.e. control action from the previous sampling period $u(t_-)$ as in (6).

[7]For the complete list of available options, call `help PolynomialMPC` in `MATLAB`. We note that if a parameter is not provided, then the algorithm uses its default value.

IV. Case Studies

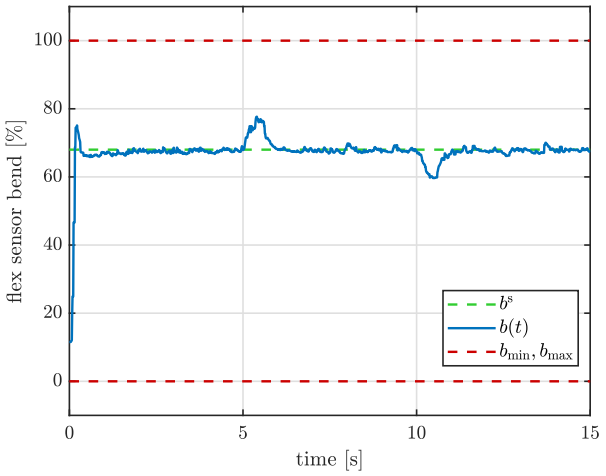

Fig. 1. Tube MPC with rate constraints: trajectory of the controlled variable.

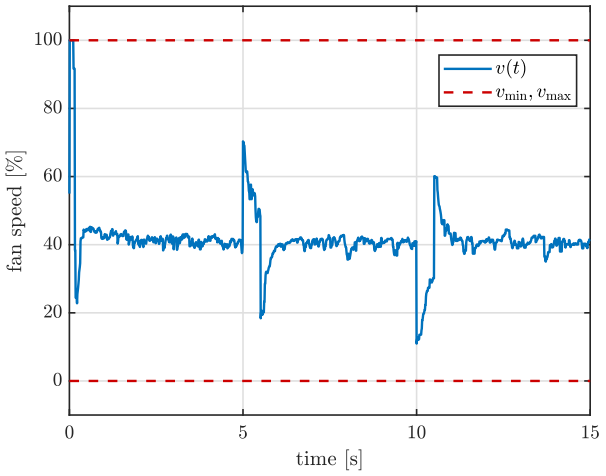

Fig. 2. Tube MPC with rate constraints: trajectory of the manipulated variable.

This section addresses the experimental implementation of the two presented control methods, i.e., the tube explicit MPC with rate constraints, and the polynomial approximation of an explicit MPC controller. To provide experimental results presenting the proposed methods, the case study was realized on a dynamical SISO device Flexy2 [31], see Figure 3. The actuator is a fan that propels an air column in an upward vertical direction.

Fig. 3. Flexy2 device [31].

The power of the airflow is measured by a flexible sensor placed in the air column. The sensor changes its electrical resistance according to the bend caused by the push of the air. Therefore, the flex sensor bend $b(t)$ in percentage is assumed as the controlled variable, and the fan speed $v(t)$ in percentage is a manipulated variable.

Flexy2 is a system with non-linear dynamics, as the sensitivity of the flex bend decreases when increasing fan speed. Moreover, the oscillations are minor at higher fan speeds. Lastly, the measurement noise is also present. These challenges make this device a suitable candidate for the implementation of tube MPC. As the system is naturally very fast and requires low sampling time with a fast evaluation of the control inputs, the explicit solution of tube MPC is considered. In this paper, the goal was to control the flex sensor bend $b(t)$ to a steady-state value b^{s} and reject the effect of a disturbance.

The model of the system was obtained through experimental identification based on several step responses. The matrices of the nominal state-space system, transformed into the discrete-time domain using sampling time $T_{\mathrm{s}} = 0.01\,\mathrm{s}$ are:

$$A = \begin{bmatrix} 0.966 \end{bmatrix}, \quad B = \begin{bmatrix} 0.101 \end{bmatrix}. \tag{18}$$

As the controlled as well as the manipulated variable were set in percentage, their values were constrained from $b_{\min} = v_{\min} = 0\%$ to $b_{\max} = v_{\max} = 100\,\%$. Considering the steady-state values where the model was linearized, i.e., $v^{\mathrm{s}} = 40\%$ and $b^{\mathrm{s}} = 68\%$, the constraints were set in both presented control methods as follows:

$$-40\,\% \leq u(t) \leq 60\,\%, \quad -68\,\% \leq x(t) \leq 32\,\%. \tag{19}$$

The rest of the two explicit MPC setups are described in the follow-up sections, according to the specific control approach, as the controllers differ in their structures and tuning parameters. Both explicit MPC feedback laws (10) were constructed in `MATLAB 2020b` programming environment, using toolboxes `MPT 3.2.1`, `MPTplus`, `YALMIP R20210331`, and solver `Gurobi 9.1.1`. The explicit MPCs were executed on CPU `AMD Ryzen 7 PRO 4750U`, 1.7 GHz with 16 GB RAM. We would like to note that, due to the paper limitations, all `MPTplus` commands are here omitted.[8]

A. Tube explicit MPC with rate constraints

In this section, the implementation of tube explicit MPC with rate constraints is described. Let us consider the system model in (18) and constraints stated in (21). As the uncertainties in the model were considered, the additive disturbance was considered in the MPC design as well and was constrained as:

$$-1\,\% \leq w(t) \leq 1\,\%. \tag{20}$$

Moreover, the change of the input variable was set to validate the control method implemented in the toolbox:

$$-55\,\% \leq \Delta u \leq 55\,\%. \tag{21}$$

By systematic tuning, the penalty matrices Q, R of the optimization problem in (4), and the terminal penalty P given by the LQR penalty, were respectively assigned as:

$$Q = 10, \quad R = 1, \quad P = 95.3852\,. \tag{22}$$

Furthermore, the terminal set was constructed as the LQR set and was defined by the following inequality:

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} x \leq \begin{bmatrix} 31.8694 \\ 21.2463 \end{bmatrix}. \tag{23}$$

The prediction horizon N was set to 30 steps. After the construction of the tube explicit model predictive controller, the disturbance rejection control problem was investigated. The control results can be seen in Figure 1 for the controlled variable, i.e., the flex sensor bend $b(t)$. In Figure 2, the corresponding trajectory of manipulated variable is depicted, i.e., the trajectory of fan speed $v(t)$. The aim was to drive the flex sensor bend to the steady state $b^{\mathrm{s}} = 68\,\%$, while rejecting the effect of the two disturbances occurring at times 5 s and 10 s.

When observing Figures 1, 2, it can be seen that the goal of the control was achieved. The effects of both disturbances were rejected, with respect to the constraints on the manipulated and controlled variables. Moreover, the constraints on the change of the manipulated variable were satisfied as well.

From the viewpoint of computational complexity, the average time to evaluate the control input was 1 milisecond, which is suitable for the considered sampling time $T_{\mathrm{s}} = 0.01\,\mathrm{s}$.

B. Polynomial approximation of explicit MPC

The following case study focuses on the second contribution of this paper, i.e., the implementation of the explicit control law approximated by a polynomial function as in (13). In this section we have assumed MPC setup as in (11) with one-norm $p = 1$, prediction horizon $N = 10$, model (18), state and input constraints (19). The weighting matrices were set to $Q = 10$ and $R = 1$.

[8]Interested readers are referred to `https://github.com/holaza/mptplus/wiki` for more detailed guidance on how to use `MPTplus` framework.

The resulting explicit feedback law (10) was defined over a polyhedral partition consisting of 17 critical regions.

Next, the polynomial controller (13) was created based on the optimal one. The degree of the polynomial was set to $n = 3$ and all other optional parameters were kept to their default values. The approximation of the optimal control law can be seen in Figure 4.

The comparison of the control results can be seen in Figure 5 for the controlled variable and in Figure 6 for the manipulated input. In both graphs, two trajectories are depicted - for the original optimal controller with subscript "o" and its polynomial approximation with subscript "p". Analogously to the first case study, the aim was to drive the flex sensor bend to the steady state $b^{\mathrm{s}} = 68\,\%$, while rejecting the effect of the two disturbances occurring at times 5 s and 10 s.

It can be seen in Figure 5, that both disturbances were successfully rejected. Compared to the optimal controller, the approximated one was less oscillating in terms of manipulated variable, see Figure 6. With more cautious control inputs, the effect of the disturbance on the trajectory of the controlled variable is more significant. The reason for less oscillating control inputs can be seen in Figure 4. The original PWA control law is steeper around the origin, while the polynomial approximation is quite moderate. The approximation can be further tuned by setting the polynomial order, which, obviously, directly affects the control performance.

From the viewpoint of computational complexity, the average time to evaluate the optimal control input was 9×10^{-4} s, which is suitable for the considered sampling time $T_{\mathrm{s}} = 0.01$ s. The average control input evaluation corresponding to the polynomial controller took even less, i.e., 3×10^{-5} s.

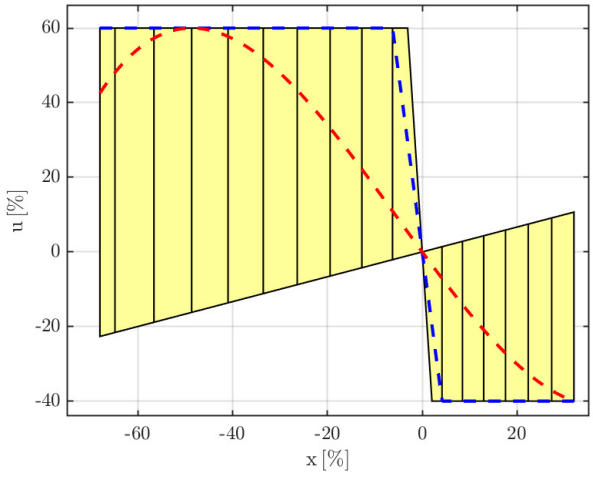

Fig. 4. Approximation of the optimal explicit MPC. The blue line represents the optimal control law, the yellow area is the stability tube, and the red line represents the polynomial approximation of the optimal control law.

V. Conclusions

The paper presented a detailed tutorial for package `MPTplus` extending the original `MPT` toolbox. Among other relevant features, the release introduces the full framework for the memory-efficient explicit tube MPC design taking into account also the limited rate of control action. `MPT` toolbox also enables the construction of light-weight polynomial-approximation-based controllers. The possibilities of construction, tuning, and validation of the various MPC controllers were analyzed by the laboratory implementation on the device with fast dynamics. The measured control profiles confirmed the ease of MPC tuning and the benefits of the user-friendly environment of `MPTplus` pushing the designed MPC controllers towards their industrial implementation.

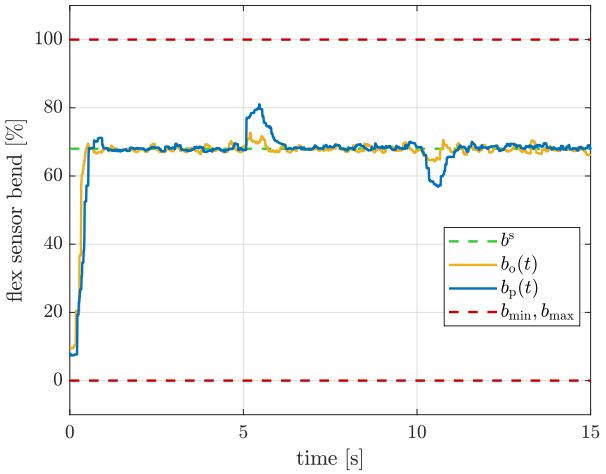

Fig. 5. Comparison of optimal explicit MPC and its polynomial approximation: trajectory of the controlled variable.

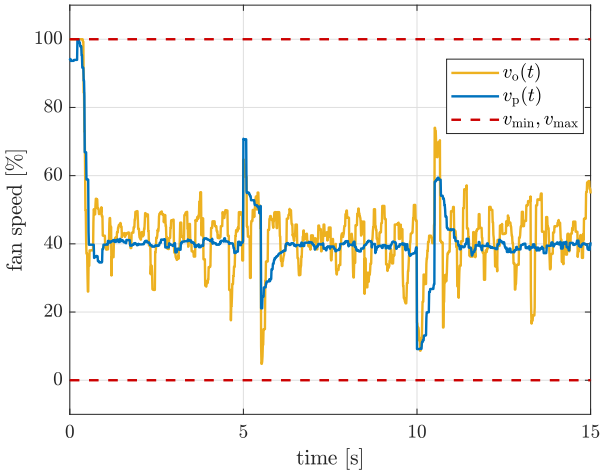

Fig. 6. Comparison of optimal explicit MPC and its polynomial approximation: trajectory of the manipulated variable.

Acknowledgemnts

This research is funded by the European Commission under the grant no. 101079342 (Fostering Opportunities Towards Slovak Excellence in Advanced Control for Smart Industries). The authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants 1/0545/20, 1/0297/22, and the Slovak Research and Development Agency under the project APVV-20-0261. L. Galčíková was also supported by an internal STU grant.

References

[1] J. Maciejowski, *Predictive Control with Constraints*. London: Prentice Hall, 2000.

[2] F. Borrelli, *Constrained Optimal Control of Linear and Hybrid Systems*. Springer Berlin Heidelberg, 2017.

[3] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[4] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, pp. 3–20, 2002.

[5] I. Pappas, D. Kenefake, B. Burnak, S. Avraamidou, H. S. Ganesh, J. Katz, N. A. Diangelakis, and E. N. Pistikopoulos, "Multiparametric programming in process systems engineering: Recent developments and path forward," *Frontiers in Chemical Engineering*, vol. 2, 1 2021.

[6] D. Kouzoupis, G. Frison, A. Zanelli, and M. Diehl, "Recent advances in quadratic programming algorithms for nonlinear model predictive control," *Vietnam J. Math*, vol. 46, pp. 863–882, 2018.

[7] S. Lucia, A. Tätulea-Codrean, C. Schoppmeyer, and S. Engell, "Rapid development of modular and sustainable nonlinear model predictive control solutions," *Control Engineering Practice*, vol. 60, pp. 51–62, 2017.

[8] B. Houska and J. Shi, "Distributed MPC with ALADIN–a tutorial," in *2022 American Control Conference (ACC)*, 2022, pp. 358–363.

[9] J. Frasch, S. Sager, and M. Diehl, "A parallel quadratic programming method for dynamic optimization problems," *Mathematical Programming Computation*, vol. 7, pp. 289–329, 2015.

[10] B. Hermans, A. Themelis, and P. Patrinos, "QPALM: A newton-type proximal augmented lagrangian method for quadratic programs," in *2019 IEEE Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 4325–4330.

[11] J. Mattingley and S. P. Boyd, "CVXGEN: a code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, pp. 1–27, 2012.

[12] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[13] A. Engelmann, Y. Jiang, H. Benner, R. Ou, B. Houska, and T. Faulwasser, "ALADIN-α–an open-source MATLAB toolbox for distributed non-convex optimization," *Optimal Control Applications and Methods*, vol. 43, no. 1, pp. 4–22, 2021.

[14] MathWorks, "MPC Toolbox," 2021, https://www.mathworks.com/products/model-predictive-control.html.

[15] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – an open source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[16] J. Andersson, J. Gillis, G. Horn, J. Rawlings, and M. Diehl, "CasADi – a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[17] Y. Chen, M. Bruschetta, E. Picotti, and A. Beghi, "MATMPC - A MATLAB based toolbox for real-time nonlinear model predictive control," *CoRR*, vol. abs/1811.08761, 2018. [Online]. Available: http://arxiv.org/abs/1811.08761

[18] J. Löfberg, "YALMIP : A Toolbox for Modeling and Optimization in MATLAB," in *Proc. of the CACSD Conf.*, Taipei, Taiwan, 2004. [Online]. Available: http://users.isy.liu.se/johanl/yalmip

[19] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds. Springer-Verlag Limited, 2008, pp. 95–110.

[20] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *2013 European Control Conference*, 2013, pp. 502–510.

[21] M. Kvasnica, J. Holaza, B. Takács, and D. Ingole, "Design and verification of low-complexity explicit MPC controllers in MPT3," in *European Control Conference 2015*, Linz, Austria, 2015, pp. 2600–2605.

[22] M. Kvasnica, J. Löfberg, and M. Fikar, "Stabilizing polynomial approximation of explicit MPC," *Automatica*, vol. 47, no. 10, pp. 2292–2297, 2011.

[23] M. Schulze Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo, "Towards encrypted MPC for linear constrained systems," *IEEE Control Systems Letters*, vol. 2, no. 2, pp. 195–200, 2018.

[24] J. Holaza, K. Kvasnicová, E. Pavlovičová, and J. Oravec, "Tube MPC extension of MPT: Experimental analysis (accepted)," in *Proceedings of the 24th International Conference on Process Control*. Štrbské Pleso, Slovakia: Slovak University of Technology in Bratislava, June 6-9, 2023 2023.

[25] Ľ. Čirka, M. Kalúz, D. Dzurková, and R. Valo, "Educational device Flexy2 in the teaching of experimental identication," in *Proceedings of the 22nd International Conference on Process Control*, M. Fikar and M. Kvasnica, Eds., Slovak University of Technology in Bratislava. Štrbské Pleso, Slovakia: Slovak Chemical Library, June 11-14, 2019 2019, pp. 239–244.

[26] D. Mayne, M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[27] S. Rakovic, E. Kerrigan, K. Kouramas, and D. Mayne, "Invariant approximations of the minimal robust positively invariant set," *IEEE Transactions on Automatic Control*, vol. 50, no. 3, pp. 406–410, 2005.

[28] M. Zhao and X. Tang, "Robust tube-based MPC with piecewise affine control laws," *Abstract and Applied Analysis*, vol. 2014, pp. 1–14, 03 2014.

[29] M. Kvasnica, J. Löfberg, and M. Fikar, "Stabilizing polynomial approximation of explicit mpc," *Automatica*, vol. 47, no. 10, pp. 2292–2297, 2011.

[30] F. Christophersen, *Optimal Control of Constrained Piecewise Affine Systems*. Springer Verlag, 2007.

[31] M. Kalúz, M. Klaučo, L. Čirka, and M. Fikar, "Flexy2: A portable laboratory device for control engineering education," *IFAC-PapersOnLine*, vol. 52, no. 9, pp. 42–47, 2019, 12th IFAC Symposium on Advances in Control Education.