FrontSeat Summer School on Optimization-Based Embedded Control Systems
Ruhr-Universität Bochum, September 1 - 5, 2025

Multi-Parametric Toolbox Workshop

Juraj Holaza

Short introduction to MPT: MPT (Multi-Parametric Toolbox) is a an open-source, Matlab-based toolbox for parametric optimization, computational geometry and model predictive control. Further information at MPT official webpage.

Installation of MPT: There are two options how to install the MPT. The preferred option is to simply run the automatic script that performs all necessary steps. The second option is to do these steps manually.

Information about this exercise: This exercise is divided into several modules. It is not necessary to complete all of them. Instead, students should work through the modules at their own pace while keeping in mind the skills being practiced. Due to the time restrictions of this workshop, it is expected that student will be able to solve first two modules $\{1, 2\}$ and one of his/her choice from $\{3, 4, 5\}$. To help you start, here are pre-prepared templates to each module:

1. Module_1_Basics
2. Module_2_CodeGen
3. Module_3_Stability
4. Module_4_Tracking
5. Module_5_AdvancedSetup

MPT support: In case of any questions/support, you can reach me at juraj.holaza@stuba.sk.

Module 1: Basics

In this module, we will cover the fundamental formulation of Model Predictive Control (MPC) in Multi-parametric-Toolbox (MPT). For these tasks we will consider following MPC setup:

$$u^\star(x_0) = \arg\min_u \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k) \tag{1a}$$

$$\text{s.t. } x_{k+1} = A x_k + B u_k,\ k = 0, \dots, N-1, \tag{1b}$$

$$y_k = C x_k + D u_k,\ k = 0, \dots, N-1, \tag{1c}$$

$$u_{\min} \le u_k \le u_{\max},\ k = 0, \dots, N-1, \tag{1d}$$

$$x_{\min} \le x_k \le x_{\max},\ k = 0, \dots, N-1, \tag{1e}$$

with

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \end{bmatrix},$$

$$u_{\min} = -1, \quad u_{\max} = 1, \quad x_{\min} = \begin{bmatrix} -5 \\ -5 \end{bmatrix}, \quad x_{\max} = \begin{bmatrix} 5 \\ 5 \end{bmatrix},$$

$$N = 3, \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 1.$$

1.) Formulate the MPC problem in the MPT framework using the provided template. Construct the MPC policy and evaluate it for a given initial condition $x_0 = [2\ 0]^\intercal$. Plot the open-loop sequence of states, inputs, and outputs. Create a closed-loop system composed of the online MPC controller and the prediction model. Simulate the evolution of the closed-loop system starting from $x_0 = [2\ 0]^\intercal$ for 20 steps. Plot the closed-loop profiles of states, outputs, and control inputs.

   Q1: Are state and input constraints respected?

   Q2: Why, in the open-loop response, the vector of states is longer than vector of outputs/inputs? What does the last state represent?

2.) Construct the explicit solution $u_0^\star(x_0)$ as a piecewise affine function of the initial condition x_0. Obtain the value of $u_0^\star$ for $x_0 = [2\ 0]^\intercal$ by evaluating the explicit solution. Measure the evaluation time and compare it to how long it takes to solve this particular problem instance numerically.

   Q1: Why evaluation time of explicit MPC is (generally) faster?

3.) Plot the critical regions of the explicit solution along with the function $u_0^\star(x_0)$. Plot also the feasible set and cost function (i.e. parametric representation of the objective function) of the MPC controller.

   Q1: Can you explain why some parts of the constrained state area (1e) is not covered in the feasible domain (e.g. state $[5\ 4]^\intercal$)?

4.) Perform the same closed-loop simulation with the explicit controller and verify if it returns the same profiles as the online MPC policy.

   Q1: Is the closed-loop stable? Repeat the task for a different initial condition which is from the feasible domain (e.g. $[-1\ 0]$).

   Q2: Are you able to find an initial condition for which the loop would not be stable? (*Hint:* use the `clicksim` method).

5.) Verify closed-loop stability by calculating a Lyapunov function.

   Q1: Can we update penalisation matrices Q, R, or modify the prediction horizon N to achieve closed-loop stability?

6.) Assume that the actuator is damaged and its behaviour has changed. The system matrix is now $B = [1\ 0.45]^\intercal$ (instead of $B = [1\ 0.5]^\intercal$). Perform closed-loop simulation (e.g. from $x_0 = [-5; 3]$) and visualize the closed-loop performance. Does the MPC still stabilize the controlled system?

Module 2: Code Generation

In this module we will use the same (original) MPC setup (1), as in the previous module, and we will show how we can export an explicit controller into:

- Matlab code,
- C code,
- Python code,

as well as how to access matrices of the given optimisation problem.

1.) Construct the explicit solution of the MPC problem (1). Analyse properties of the controller. (*Hint:* look what `empc.optimizer` returns.)

2.) Export the explicit controller to pure MATLAB code using the optimizer's method `toMatlab()`. Evaluate the exported function for a particular initial condition (e.g. $x_0 = [3\ 0]^\intercal$). (*Hint:* We aim to export control law that is stored under function `'primal'`. For the tie-break function we can use `'first-region'`.)

3.) Notice that the exported function returns the entire open-loop, i.e., $u^\star(x_0) = [u_0\ u_1\ \ldots\ u_{N-1}]^\intercal$. Trim the function to return only the first control action $u^\star(x_0) = u_0$ and export this function to a new file. Compare if the two exported files return the same RHC for the same initial condition.

    Q1: Can we calculate how much memory (numbers) we have saved if the trimmed matrices will be used? (*Hint:* Either use rigorous calculations or simply compare matrices [H, ni, fF, fg] in both m-files.)

4.) Create a closed-loop simulation using the exported function as the controller and the controlled system is the prediction model. Compare the solution with results using build in MPT3 functions from previous module. (*Hint:* use the original `empc` controller and methods `ClosedLoop` + `simulate`.)

5.) Export the explicit MPC to C-language using method `toC()` providing function name to be exported `'primal'` and name of the file. Compile the generated mex file and evalaute the function in MATLAB. (*Hint:* Build in MATLAB toolbox MinGW-w64 C/C++ compiler is required. For different MPC formulations, the parametric vector is composed of $[x_0\ u_{-1}\ y_{\text{ref}}]^\intercal$, i.e., what is the input vector to the exported function.)

6.) Export the explicit MPC to Python-language using method `toPython()` providing name of the file, function name to be exported `'primal'`, and tie-break method `'first-region'`.

7.) Export matrices of the optimisation problem via using function `mpc.getMatrices('dense')`. Evaluate the MPC policy for an initial condition $x(t) = x_0$ to obtain open-loop sequence of optimal control actions $U(t) = [u_0\ u_1\ \ldots\ u_{N-1}]^\intercal$. Then analyse if all constraints $GU(t) <= W + Ex(t)$ are satisfied if such control action will be used. Use $x_0 = [2\ 0]^\intercal$ and $x_0 = [-5\ 3]^\intercal$.

Module 3: Stability

In this module we will use the same (original) MPC setup (1), as in the first module, of what we know that it is not stable. In this module we will show which ingredients we need to add the the MPC policy to provide guarantees of stability.

1.) Add terminal set to the MPC setup in (1). (*Hint:* The terminal set can be computed in MPT via `model.LQRSet`.)

2.) Add terminal penalty to the MPC setup in (1). (*Hint:* The terminal penalty is solution of the Ricatty equation, i.e. unconstrained MPC problem.)

3.) Construct a new explicit MPC feedback law and via `clicksim` function analyse if the controller stabilizes the controlled system (prediction model).

4.) Provide a certificate of stability (via constructing Lyapunov function).

5.) Provide answers to following questions:

  Q1: Can we choose arbitrary prediction horizon and the MPC (with terminal set/penalty) will always guarantee stability?

  Q2: Does the volume of the MPC feasible domain increase/decrease with the prediction horizon N?

  Q3: Can we change volume of the feasibility domain by changing N to infinity?

Module 4: Tracking

This module is devoted to tracking of a state/output/input reference. In this module we will consider following MPC optimisation problem:

$$
\begin{aligned}
u^\star(x_0) = \arg\min_u \; & \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k) + x_N^T P x_N && \text{(2a)}\\
\text{s.t. } & x_{k+1} = A x_k + B u_k,\ k = 0,\dots,N-1, && \text{(2b)}\\
& y_k = C x_k + D u_k,\ k = 0,\dots,N-1, && \text{(2c)}\\
& u_{\min} \le u_k \le u_{\max},\ k = 0,\dots,N-1, && \text{(2d)}\\
& x_{\min} \le x_k \le x_{\max},\ k = 0,\dots,N-1, && \text{(2e)}
\end{aligned}
$$

with

$$
\begin{aligned}
& A = \begin{bmatrix} 0.7115 & -0.4345 \\ 0.4345 & 0.8853 \end{bmatrix}, \quad B = \begin{bmatrix} 0.2173 \\ 0.0573 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \end{bmatrix}, \\
& u_{\min} = -5, \quad u_{\max} = 5, \quad x_{\min} = \begin{bmatrix} -2.8 \\ 0 \end{bmatrix}, \quad x_{\max} = \begin{bmatrix} 10 \\ 10 \end{bmatrix}, \\
& N = 5, \quad Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad R = 1, \quad P = P_{\text{LQR}}.
\end{aligned}
$$

1.) Obtain the explicit representation of the MPC problem in (2) and perform closed-loop simulation from $x_0 = [4\ 2]^\intercal$ with 20 steps. Plot resulting trajectories. (Note: this is just a regulation problem.)

2.) Instead of regulating towards the origin, we want the MPC controller to steer the system's states to $x_{\text{ref}} = [0\ 2]^\intercal$. To do so, modify the stage cost to $(x_k - x_{\text{ref}})^\intercal Q (x_k - x_{\text{ref}})^\intercal + u_k^\intercal R u_k$. Verify the performance of the controller in a closed-loop simulation starting from $x_0 = [4\ 2]^\intercal$.

   Q1: Is the reference tracked without a steady-state offset? Explain the reason behind your answer.

   Q2: Is the dimensionality of the feasible set of the explicit controller still the same as in the regulation problem? Explain why.

3.) To reject the steady-state offset, calculate the target control input u_{ref} such that $x_{\text{ref}} = A x_{\text{ref}} + B u_{\text{ref}}$ holds, and change the stage cost to $(x_k - x_{\text{ref}})^\intercal Q (x_k - x_{\text{ref}}) + (u_k - u_{\text{ref}})^\intercal R (u_k - u_{\text{ref}})$. Verify the performance in a closed-loop simulation. Why is the steady-state offset rejected?

4.) Create explicit MPC controller to track the output reference $y_{\text{ref}} = 2$ such that $x = Ax + B u_{\text{ref}}$ and $y_{\text{ref}} = Cx + D u_{\text{ref}}$ hold. To do this we need to change the objective function to $(y_k - y_{\text{ref}})^\intercal Q (y_k - y_{\text{ref}}) + (u_k - u_{\text{ref}})^\intercal R (u_k - u_{\text{ref}})$. Verify the performance in a closed-loop simulation. Is still the steady-state offset rejected?

5.) Create explicit MPC policy that tracks an arbitrary output reference $y_{\text{ref}} \in \mathbb{R}$. Verify the performance in a closed-loop simulation with $y_{\text{ref}} = 2$ for the first 50 steps and $y_{\text{ref}} = 1$ for another 50 steps.

   Q1: Can we define a negative reference? Why not?

   Q2: Is dimensionality of the feasible set (of the explicit controller) still the same as with the regulation problem? Why? (*Hint:* Use `empc.partition.Dim` to determine dimensionality of the polyhedral partition.)

Module 5: Advanced MPC Setup

This module provides a quick overview on some advanced MPC features such as: soft constraints, delta-u constraints/penalty, and move-blocking. In this module we will continue with the last MPC formulation from the previous modelu, i.e. let us assume following MPC problem:

$$u^\star(x_0) = \arg\min_u \sum_{k=0}^{N-1} \left((y_k - y_{\text{ref}})^\intercal Q (y_k - y_{\text{ref}}) + (u_k - u_{\text{ref}})^\intercal R (u_k - u_{\text{ref}})\right) \tag{3a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k,\ k = 0,\dots,N-1, \tag{3b}$$

$$y_k = Cx_k + Du_k,\ k = 0,\dots,N-1, \tag{3c}$$

$$u_{\min} \le u_k \le u_{\max},\ k = 0,\dots,N-1, \tag{3d}$$

$$x_{\min} \le x_k \le x_{\max},\ k = 0,\dots,N-1, \tag{3e}$$

with

$$A = \begin{bmatrix} 0.7115 & -0.4345 \\ 0.4345 & 0.8853 \end{bmatrix},\quad B = \begin{bmatrix} 0.2173 \\ 0.0573 \end{bmatrix},\quad C = \begin{bmatrix} 0 & 1 \end{bmatrix},\quad D = \begin{bmatrix} 0 \end{bmatrix},$$

$$u_{\min} = -5,\quad u_{\max} = 5,\quad x_{\min} = \begin{bmatrix} -2.8 \\ 0 \end{bmatrix},\quad x_{\max} = \begin{bmatrix} 10 \\ 10 \end{bmatrix},$$

$$N = 5,\quad Q = 10,\quad R = 1.$$

1.) Construct online MPC policy for the problem (3). Perform closed-loop simulation with $y_{\text{ref}} = 2$ for the first 50 steps and $y_{\text{ref}} = 0$ for another 50 steps starting from $x_0 = [4\ 2]^\intercal$. Assume that at time step 80 a sensor will provide a faulty value of the second state with deviation of -1 from the correct value, i.e., at time step 80 perform $x_0 = x_0 + [0\ \ -1]^\intercal$. Plot resulting closed-loop trajectories.

   Q1: Will MPC handle such a sensor error? Explain why it not.

   Q2: Will soft constraints $x_{\min} - s \le x_k \le x_{\max} + s$, where slacks $s \ge 0$ are heavily penalized, solve this problem? (*Hint:* Use `model.x.with('softMax')` and `model.x.with('softMin')` to add soft constraints on states.)

2.) Construct online MPC policy that will restrict change of the control action in two subsequent steps to ± 1, i.e., we want to add $-1 <= u_k - u_{k-1} <= 1$. Perform closed-loop simulation and verify if this delta-u constraint is satisfied.

3.) Construct online MPC policy that will penalize the change of the control action in two subsequent steps by a penalty matrix H, i.e., $(u_k - u_{k-1})^\intercal H (u_k - u_{k-1})$. Verify how the matrix H affects the closed-loop performance.

   Q1: Can we remove the 'u.reference' formulation and will the MPC track the reference without an offset? In another words, can we replace $((u_k - u_{\text{ref}})^\intercal R (u_k - u_{\text{ref}})$ with $(u_k - u_{k-1})^\intercal H (u_k - u_{k-1})$? (*Hint:* Comment the `model.u.with('reference')` and `model.u.reference = 'free';`. Moreover, in the `simulate()` function remove `'u.reference', Uref,"`. Rerun the scenario.)

4.) Construct online MPC policy that uses move-blocking technique with blocked control action from $N = 3$ to $N = 5$. in another words, we want to force the last 3 optimized control actions to be identical $u_2 = u_3 = u_4$. Perform closed-loop simulation and analyse how this technique affects control performance.

   Q1: Are we able to construct explicit controller of the original problem (without the move blocking technique)? Try to change complexity of the MPC problem via modifying the length of blocking interval (as well as the N) and try to construct the explicit solution.