## Table of Contents I





notes

- this is the table of contents of this document; each section corresponds to a specific part of the course

# Tuning Model Predictive Control for LTI Systems



notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| tuning MPC | u2, e2 |

| prerequisite content units | taxonomy levels |
|---|---|
| MPC fundamentals | u1, e1 |
| LTI systems | u1, e1 |



notes

## Main ILO of sub-module "Tuning Model Predictive Control for LTI Systems"

Design and tune an MPC controller for LTI systems to meet specified performance criteria

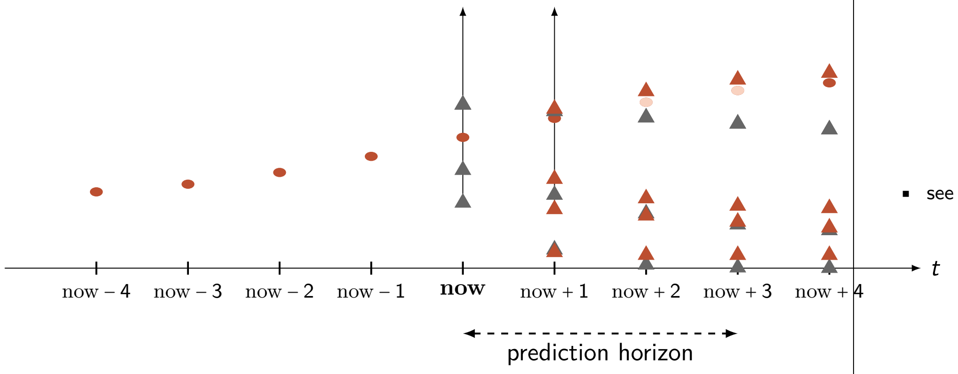


notes

- By the end, you should be able to select appropriate tuning parameters, understand performance trade-offs in MPC, and attempt to implement a basic tuned MPC controller

## The working principle, graphically

signals measured past $y$ wished future $y$ potential $u$, and corresponding forecasted $y$ best $u$ = "best" forecasted $y$ execute $u(\mathrm{now})$ measure, and repeat






notes

- see also `https://www.youtube.com/watch?v=UR0hOmjaHp0`!

## MPC in formulas

(for LTI systems)

assumed dynamics: $x_{k+1} = Ax_k + Bu_k$

optimization problem:

$$\begin{aligned}\min_{u[0],\dots,u[N]} \; & \sum_{k=0}^{N-1} \underbrace{x[k]^T Q x[k]}_{\text{state cost}} + \underbrace{u[k]^T R u[k]}_{\text{control cost}} + \underbrace{x[N]^T P x[N]}_{\text{terminal cost}} \\ \text{s.t.} \; & x_{k+1} = Ax[k] + Bu[k] \quad \forall k \in \{0, \dots, N-1\} \\ & u_{\text{min}} \leq u[k] \leq u_{\text{max}} \\ & x_{\text{min}} \leq x[k] \leq x_{\text{max}} \\ & x[0] = x(t) \quad \text{(initial condition)}\end{aligned}$$



notes

- $N$ is called the "prediction horizon length"
- $Q \geq 0$, $R > 0$ are instead weight matrices (that actually are tuning parameters)
- $P$ is dubbed the "terminal cost" weight
- and the interesting thing is that we can put constraints, that make MPC powerful (but increase its computational complexity)

## Key parameters

$$\min_{u[0],\ldots,u[N]} \sum_{k=0}^{N-1} \underbrace{x[k]^T Q x[k]}_{\text{state cost}} + \underbrace{u[k]^T R u[k]}_{\text{control cost}} + \underbrace{x[N]^T P x[N]}_{\text{terminal cost}}$$

- prediction horizon $N$
- weight matrices $Q$, $R$, $P$
- the constraints parameters $u_{\text{min}}, u_{\text{max}}, x_{\text{min}}, x_{\text{max}}$



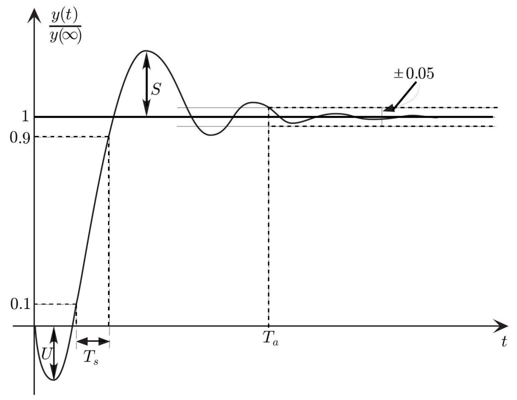
notes

- Prediction horizon affects stability and computational load
- Control horizon offers implementation flexibility
- Weights balance state vs. control effort

## But which performance criteria shall we optimize?

Standard options:

- settling time
- overshoot
- control effort
- robustness
- computational efficiency






notes

- Note that these are often conflicting objectives
- You will need to prioritize one or a few of them wrt the other, based on you application - so you need to know what you want

## General trade-offs

$$\min_{u[0],\ldots,u[N]} \sum_{k=0}^{N-1} \underbrace{x[k]^T Q x[k]}_{\text{state cost}} + \underbrace{u[k]^T R u[k]}_{\text{control cost}} + \underbrace{x[N]^T P x[N]}_{\text{terminal cost}}$$

- $\uparrow N \implies$ better performance but more computations
- $\uparrow Q \implies$ faster state convergence but more aggressive control
- $\uparrow R \implies$ smoother control but slower response



notes

- Show example curves of these relationships
- Emphasize application-dependent choices

## Tuning methodology

$$\min_{u[0],\ldots,u[N]} \sum_{k=0}^{N-1} \underbrace{x[k]^T Q x[k]}_{\text{state cost}} + \underbrace{u[k]^T R u[k]}_{\text{control cost}} + \underbrace{x[N]^T P x[N]}_{\text{terminal cost}}$$

at every iteration, evaluate the performance and iteratively refine the parameters

- start with the infinite horizon equivalent (i.e., LQR)
- move to a shorter prediction horizon (5-20 samples)
- then adjust the weights ($Q$ first, then $R$)



notes

- Systematic approach prevents trial-and-error
- Physical meaning helps initial guesses

## Summarizing

Design and tune an MPC controller for LTI systems to meet specified performance criteria

- determining performance requirements require simulating and evaluating to iteratively refine parameters



notes

- You should now understand this systematic tuning approach
- Remember the fundamental trade-offs

Most important python code for this sub-module



notes

## Model predictive control python toolbox

https://www.do-mpc.com/en/latest/



notes

- Show how to adjust parameters
- Demonstrate effect of changing weights

Self-assessment material



notes

## Question 1

What is the primary effect of increasing the Q matrix in MPC tuning?

**Potential answers:**

I: **(wrong)** Reduced computational requirements
II: **(wrong)** Smoother control actions
III: **(correct)** Faster state convergence
IV: **(wrong)** Increased robustness to disturbances

**Solution 1:**

The Q matrix weights the state error, so increasing it prioritizes faster convergence to the desired state.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 2

What is the fundamental purpose of the terminal cost (P) in MPC?

**Potential answers:**

I: **(wrong)** To reduce the computational complexity of the optimization
II: **(correct)** To ensure stability by approximating infinite horizon behavior
III: **(wrong)** To enforce hard constraints on the system states
IV: **(wrong)** To prioritize certain states over others in the transient response
V: **(wrong)** I do not know

**Solution 1:**

The terminal cost P is typically chosen as the solution to the algebraic Riccati equation to guarantee stability, effectively approximating the infinite horizon cost-to-go beyond the prediction horizon N.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 3

Why might increasing the prediction horizon N improve controller performance?

**Potential answers:**

I: **(wrong)** It allows using larger Q matrices in the cost function
II: **(correct)** The controller can account for longer-term system behavior
III: **(wrong)** It reduces the need for state constraints
IV: **(wrong)** It makes the optimization problem convex
V: **(wrong)** I do not know

**Solution 1:**

A longer prediction horizon enables the controller to "see further ahead" and make better decisions by considering more future states, though this comes at increased computational cost.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 4

What is the primary consequence of setting $R = 0$ in the MPC cost function?

**Potential answers:**

I: **(wrong)** The controller will become unstable
II: **(wrong)** The state constraints will be ignored
III: **(correct)** The controller may use arbitrarily large control inputs
IV: **(wrong)** The prediction horizon becomes irrelevant
V: **(wrong)** I do not know

**Solution 1:**

The R matrix penalizes control effort. With R=0, the optimizer has no incentive to limit control inputs, potentially leading to aggressive (and possibly impractical) control actions.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 5

Which of these represents a fundamental trade-off in MPC tuning?

**Potential answers:**

I: **(wrong)** Between continuous-time and discrete-time formulations
II: **(wrong)** Between state estimation and control computation
III: **(correct)** Between performance and computational complexity
IV: **(wrong)** Between linear and nonlinear system models
V: **(wrong)** I do not know

**Solution 1:**

MPC involves balancing control performance (better with longer horizons, more constraints) against computational tractability (worse with these same factors), which is a fundamental design consideration.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 6

What is the main advantage of MPC compared to LQR control?

**Potential answers:**

I: **(wrong)** MPC always requires less computational power
II: **(wrong)** MPC guarantees global optimality for nonlinear systems
III: **(correct)** MPC can explicitly handle state and input constraints
IV: **(wrong)** MPC doesn't require a system model
V: **(wrong)** I do not know

**Solution 1:**

While both are optimal controllers, MPC's key advantage is its ability to explicitly incorporate and satisfy constraints during the optimization process, which LQR cannot do natively.



notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "Tuning MPC for LTI Systems"

- MPC performance depends on careful parameter selection
- Prediction horizon affects stability and computation
- Weight matrices balance state vs control objectives
- Systematic tuning follows an iterative procedure



notes

- Remember these key points when implementing your own MPC