# Underactuated Robots
# Lecture 4: Model Predictive Control

Nicola Scianca

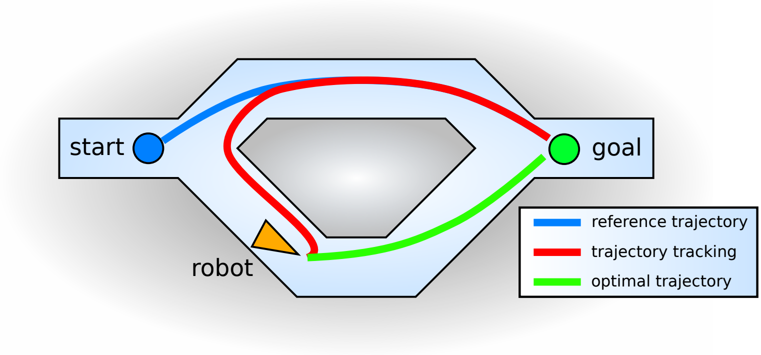
September 2024

# TO vs control

- trajectory optimization is good for finding **reference trajectories**
- the optimization can be performed **offline**, and take all the time necessary
- the control is then performed online via some form of **trajectory tracking**

# TO vs control

- if we deviate significantly from the initially planned trajectory, that trajectory might not be optimal anymore




- if we could **repeat the optimization** at every control cycle we would always be moving along an optimal trajectory

# model predictive control

- **Model Predictive Control** (MPC) looks similar to TO, but the optimization is performed at **every control cycle**

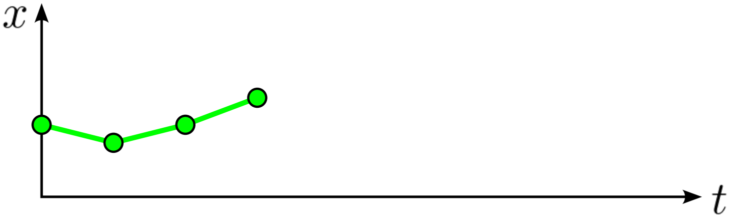
$$
\begin{aligned}
\min \sum_{k=0}^{N-1} & l(x_k, u_k) + l_N(x_N, u_N) \\
\text{s.t.} \quad x_{k+1} &= Ax_k + Bu_k \\
x_0 &= x_{\text{current}} \\
x_N &= 0
\end{aligned}
$$

- the initial state of the horizon $x_0$ is set at each iteration to be equal to the **measured state** $x_{\text{current}}$ (**feedback!**)

- only the **first input** of the optimal trajectory is applied

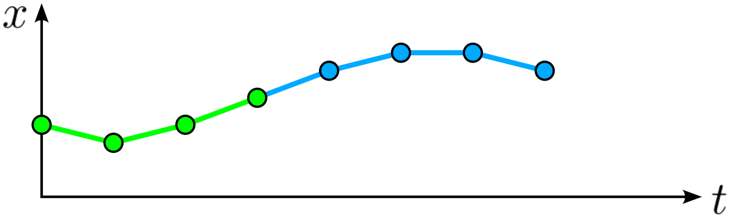
# model predictive control

- let's see an example of MPC in action: in green is our **realized trajectory** up to the present moment



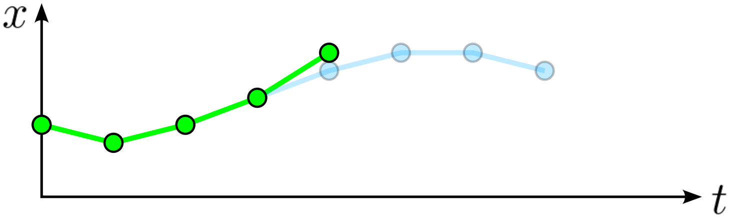
# model predictive control

- we **predict** an optimal trajectory starting from the current state and apply the **first predicted input**



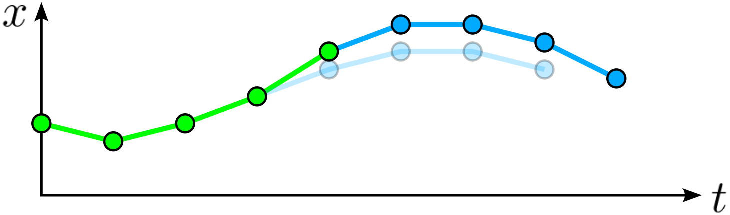
# model predictive control

- the new state will be slightly different than we predicted due to **model inaccuracy** and **disturbances**



# model predictive control

- now we find a new prediction starting from the current **measured state**



# linear MPC regulation

- consider a linear system

$$x_{k+1} = Ax_k + Bu_k$$

- let's write an MPC to regulate the state $x$ to the origin

$$\begin{aligned} \min \sum_{k=0}^{N-1} \left( x_k^T Q x_k + u_k^T R u_k \right) + x_N^T P x_N& \\ \text{s.t.} \quad x_{k+1} &= Ax_k + Bu_k \\ x_0 &= x_{\text{current}} \\ x_N &= 0 \end{aligned}$$

# terminal constraint

- $x_N = 0$ is an example of **terminal constraint** that keeps the final state within a **positively invariant set**
- a positively invariant set $\mathcal{X}$ is such that $x_N \in \mathcal{X}$, there exists a control input $u_N$ such that $x_{N+1} = Ax_N + Bu_N \in \mathcal{X}$
- in particular, since the system is linear, we just have to apply zero input to stay at the origin
- the next two slides will show how to prove **recursive feasibility** and **stability** for this simple MPC controller, in the nominal case (no disturbance and no modeling errors)

# recursive feasibility

- we can show that this MPC is **recursively feasible**: if a solution exists at time $t$, we can find one at $t+1$ (and so on)
- suppose that the solution we found at time $t$ is

$$\bar{u}_t^* = (u_{t|t}^*, u_{t+1|t}^*, u_{t+2|t}^*, \ldots, u_{t+N-1|t}^*)$$

  where $u_{t+i|t}^*$ is the optimal input at $t+i$ predicted at time $t$

- the following input sequence is feasible at $t+1$

$$\bar{u}_{t+1} = (u_{t+1|t}^*, u_{t+2|t}^*, \ldots, u_{t+N-1|t}^*, 0)$$

  because $\bar{u}_t^*$ satisfied the terminal constraint $x_{t+N|t} = 0$, and by applying zero input at the end we remain in the origin, thus also $\bar{u}_{t+1}$ satisfies the terminal constraint

# stability

- define $J(\bar{u})$ as the cost function evaluated for the input sequence $\bar{u}$

$$J(\bar{u}) = \sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k\right) + x_N^T P x_N$$

- $J(\bar{u}_t^*)$ and $J(\bar{u}_{t+1})$ are identical except for the fact that the second sum has one less term

$$J(\bar{u}_t^*) = \sum_{k=0}^{N-1} \left((x_{t+k|t}^*)^T Q x_{t+k|t}^* + (u_{t+k|t}^*)^T R u_{t+k|t}^*\right)$$

$$J(\bar{u}_{t+1}) = \sum_{k=1}^{N-1} \left((x_{t+k|t}^*)^T Q x_{t+k|t}^* + (u_{t+k|t}^*)^T R u_{t+k|t}^*\right)$$

- $J(\bar{u})$ is positive definite and $J(\bar{u}_t^*) \geq J(\bar{u}_{t+1}) \geq J(\bar{u}_{t+1}^*)$, thus $J(\bar{u})$ is a **Lyapunov function**

# model predictive control: strategies

- the main difficulty with MPC is **simplifying** the optimization so that it can be performed in a reasonable time (under $10$ ms, sometimes under $1$ ms)

- **shorten** the prediction horizon: instead of optimizing the full task, just the immediate future (a couple of seconds or even less than one second)

- come up with a good **terminal cost** or **terminal constraint** to make up for the reduced horizon length

# model predictive control: strategies

- use a **simplified model**: not all the details are equally important; sometimes a robot can be approximated as a rigid body or even a point!

- **parametrize** the trajectories: this can reduce the number of variables (e.g., larger time-steps, Bezier curves, ...)

- **linearize** around a previous solution: this can greatly reduce computation time because you start close to a minimum; it also avoid jumping between different local minima which can cause discontinuities

# real-time iteration

- **real-time iteration** is a common way of implementing nonlinear model predictive control in real-time
- instead of trying to converge to the optimal solution, we perform a **single SQP iteration** at each control cycle
- this is possible because we always **warmstart** the SQP with the solution found in the previous control cycle
- every time we find a suboptimal solution, but over multiple control cycles we still get closer to the optimum