Underactuated Robots
Lecture 4: Model Predictive Control

Nicola Scianca

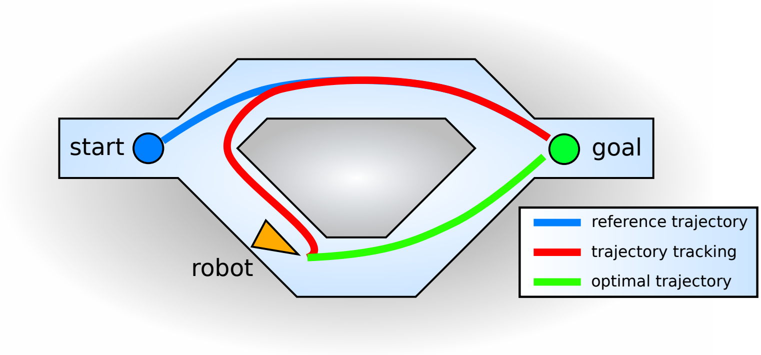
December 2024

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**

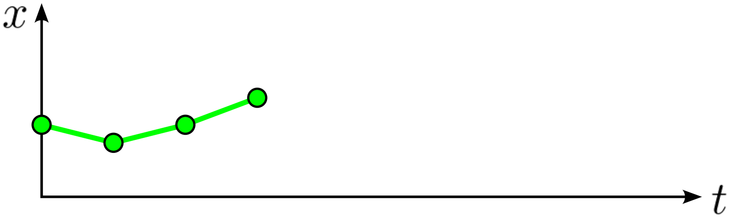
$$
\begin{aligned}
\min \sum_{k=0}^{N-1} & l(x_k, u_k) + l_N(x_N, u_N) \\
\text{s.t.} \quad x_{k+1} &= A x_k + B u_k \\
x_0 &= x_{\text{current}} \\
x_N &= 0
\end{aligned}
$$

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

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

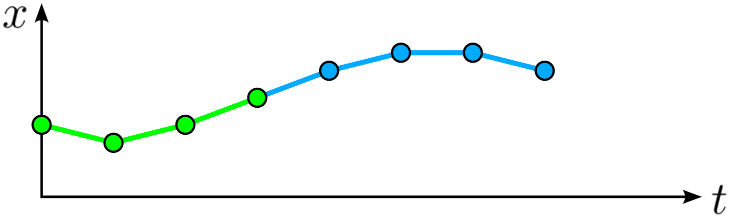
model predictive control

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



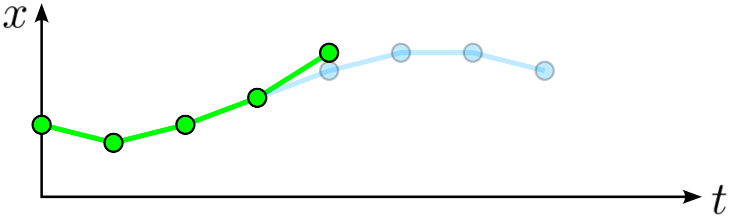
model predictive control

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



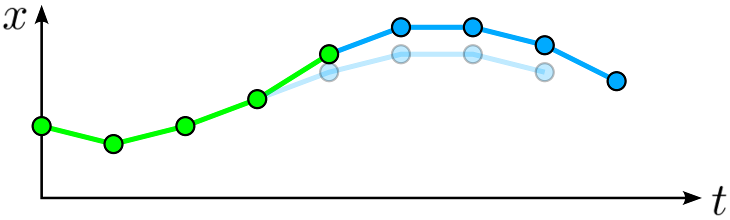
model predictive control

- 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 - unconstrained

- consider a linear system

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

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

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

- for the moment, we have no constraints, aside from the linear dynamics and the initial state

linear MPC - unconstrained

- in this problem we have no constraints, aside from the linear dynamics and the initial state
- this means that if we adopt a **single shooting** approach we can eliminate all the constraints
- to do this, we need to perform a series of substitutions so that the **dynamic constraints** disappear

linear MPC - unconstrained

- the dynamic constraints are

$$x_{k+1} = Ax_k + Bu_k, \quad \text{for } i = 0, \ldots, N-1$$

- starting from the initial state x_0, we can write

$$\begin{aligned}
x_1 &= Ax_0 + Bu_0 \\
x_2 &= Ax_1 + Bu_1 = A(Ax_0 + Bu_0) + Bu_1 \\
&= A^2 x_0 + ABu_0 + Bu_1 \\
x_3 &= Ax_2 + Bu_2 = A(A^2 x_0 + ABu_0 + Bu_1) + Bu_2 \\
&= A^3 x_0 + A^2 Bu_0 + ABu_1 + Bu_2 \\
&\vdots
\end{aligned}$$

linear MPC - unconstrained

- if we keep going until the end of the horizon, we get

$$
\begin{aligned}
x_1 &= Ax_0 + Bu_0 \\
x_2 &= A^2x_0 + ABu_0 + Bu_1 \\
x_3 &= A^3x_0 + A^2Bu_0 + ABu_1 + Bu_2 \\
&\vdots \\
x_N &= A^Nx_0 + A^{N-1}Bu_0 + \cdots + ABu_{N-2} + Bu_{N-1}
\end{aligned}
$$

linear MPC - unconstrained

- we can express this as matrix multiplication

$$\underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{pmatrix}}_{X_{k+1}} = \underbrace{\begin{pmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^N \end{pmatrix}}_{\bar{T}} x_0 + \underbrace{\begin{pmatrix} B & 0 & 0 & \dots & 0 \\ AB & B & 0 & \dots & 0 \\ A^2B & AB & B & \dots & 0 \\ \vdots & & & & \\ A^{N-1}B & A^{N-2}B & A^{N-3}B & \dots & B \end{pmatrix}}_{\bar{S}} \underbrace{\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{pmatrix}}_{U_k}$$

- we find an expression for the **predicted state** X_{k+1} in terms of the **current state** x_0 and the **predicted inputs** U_k

$$X_{k+1} = \bar{T}x_0 + \bar{S}U_k$$

linear MPC - unconstrained

- recall the cost function that we want to minimize

$$J = \frac{1}{2}\sum_{i=k}^{k+N-1}\left(x_i^T Q x_i + u_i^T R u_i\right) + x_{k+N}^T Q x_{k+N}$$

- this too we can write it in terms of the entire prediction

$$\frac{1}{2}\underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}^T}_{X_{k+1}^T}\underbrace{\begin{pmatrix} Q & 0 & \dots & 0 \\ 0 & Q & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & Q \end{pmatrix}}_{\bar{Q}}\underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}}_{X_{k+1}} + \frac{1}{2}\underbrace{\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix}^T}_{U_k^T}\underbrace{\begin{pmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & R \end{pmatrix}}_{\bar{R}}\underbrace{\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix}}_{U_k}$$

linear MPC - unconstrained

- we can eliminate the state from the cost function we just wrote by substituting in the expression for the **prediction**

$$
\begin{aligned}
J &= \frac{1}{2} X_{k+1}^T \bar{Q} X^{k+1} + \frac{1}{2} U_k^T \bar{R} U_k \\
&= \frac{1}{2} (\bar{T} x_k + \bar{S} U_k)^T \bar{Q} (\bar{T} x_k + \bar{S} U_k) + \frac{1}{2} U_k^T \bar{R} U_k \\
&= \frac{1}{2} x_k^T \bar{T}^T \bar{Q} \bar{T} x_k + \frac{1}{2} U_k^T \bar{S}^T \bar{Q} \bar{S} U_k + U_k^T \bar{S}^T \bar{Q} \bar{T} x_k + \frac{1}{2} U_k^T \bar{R} U_k \\
&= \underbrace{\cancel{\frac{1}{2} x_k^T \bar{T}^T \bar{Q} \bar{T} x_k}}_{\text{constant term}} + \frac{1}{2} U_k^T \underbrace{(\bar{S}^T \bar{Q} \bar{S} + \bar{R})}_{H} U_k + U_k^T \underbrace{\bar{S}^T \bar{Q} \bar{T}}_{F} x_k
\end{aligned}
$$

- we can eliminate the linear term as it only changes the value of the minimum, not where the minimum is

linear MPC - unconstrained

- we managed to express the problem as

$$\min_{U_k} \frac{1}{2} U_k^T H U_k + U_k^T F x_k$$

- if there are **no constraints**, we can solve this easily by zeroing the gradient

$$\nabla \left(\frac{1}{2} U_k^T H U_k + U_k^T F x_k \right) = 0 \quad \Longrightarrow \quad H U_k + F x_k = 0$$

$$U_k = -H^{-1} F x_k$$

linear MPC - unconstrained

- since we want to only apply the first input, let's use a matrix $I_{\text{sel}} = (I, 0, 0, \dots)$ to select it

$$u_k = -I_{\text{sel}} H^{-1} F x_k$$

- after having applied the input u_k, we measure the new state, and then repeat the process once again

- as it turns out, with **no constraints**, linear MPC is a form of **linear state feedback**

linear MPC - constrained

- let's now add some constraints

$$
\begin{aligned}
\min \frac{1}{2} \sum_{i=k}^{k+N-1} & \left(x_{i+1}^T Q x_{i+1} + u_i^T R u_i \right) \\
\text{s.t.} \quad x_{i+1} &= A x_i + B u_i \\
x_k &= x_{\text{meas}} \\
u_{\min} \leq u_i &\leq u_{\max} \qquad \text{for } i = 0, \dots, N-1 \\
x_{\min} \leq x_i &\leq x_{\max} \qquad \text{for } i = 1, \dots, N
\end{aligned}
$$

linear MPC - constrained

- our goal is to be able to write the problem in the form

$$
\begin{aligned}
&\min \frac{1}{2} U_k^T H U_k + U_k F \\
&\text{s.t.} \quad A U_k \leq b
\end{aligned}
$$

- this is a standard **quadratic program** (QP)

- it has no closed-form solution but there are libraries to solve it very efficiently (HPIPM, OSQP, PROXQP, etc, ...)

linear MPC - constrained

- write the input constraints $u_{\min} \leq u_i \leq u_{\max}$ in matrix form

$$\underbrace{\begin{pmatrix} u_{\min} \\ u_{\min} \\ \vdots \\ u_{\min} \end{pmatrix}}_{U_{\min}} \leq \underbrace{\begin{pmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-1} \end{pmatrix}}_{U_k} \leq \underbrace{\begin{pmatrix} u_{\max} \\ u_{\max} \\ \vdots \\ u_{\max} \end{pmatrix}}_{U_{\max}}$$

$$U_{\min} \leq U_k \leq U_{\max}$$

- in order to get it into the form $Ax \leq b$, we split it in two constraints and multiply the first one by -1 to switch the sign of the inequality

$$\begin{matrix} U_k \geq U_{\min} \\ U_k \leq U_{\max} \end{matrix} \implies \begin{matrix} -U_k \leq -U_{\min} \\ U_k \leq U_{\max} \end{matrix} \implies \begin{pmatrix} -I \\ I \end{pmatrix} U_k \leq \begin{pmatrix} -U_{\min} \\ U_{\max} \end{pmatrix}$$

linear MPC - multiple shooting

- we can also write a linear MPC problem with a multiple shooting formulation
- in this case, instead of performing substitutions, we keep all the state variables inside the problem
- let's call W_k the vector of decision variables, which now includes inputs and states

$$W_k = (x_k, u_k, x_{k+1}, u_{k+1}, \dots, x_{k+N-1}, u_{k+N-1}, x_{k+N})^T$$

linear MPC - multiple shooting

- we can write the **dynamics constraint** and the **initial state constraint** on the vector of decision variables as

$$
\begin{pmatrix}
I & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
-A & -B & I & 0 & 0 & \dots & 0 & 0 & 0 \\
0 & 0 & -A & -B & I & \dots & 0 & 0 & 0 \\
\vdots & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & \dots & -A & -B & I
\end{pmatrix}
\begin{pmatrix}
x_k \\ u_k \\ x_{k+1} \\ u_{k+1} \\ \vdots \\ x_{k+N-1} \\ u_{k+N-1} \\ x_{k+N}
\end{pmatrix}
=
\begin{pmatrix}
x_{\text{current}} \\ 0 \\ 0 \\ \vdots \\ 0
\end{pmatrix}
$$

- this is a very **sparse** constraint: its matrix has lots of zeros

linear MPC - multiple shooting

- the cost function is very easy to write

$$
J = \frac{1}{2} \begin{pmatrix} x_k \\ u_k \\ x_{k+1} \\ u_{k+1} \\ \vdots \\ x_{k+N-1} \\ u_{k+N-1} \\ x_{k+N} \end{pmatrix}^T \begin{pmatrix} Q & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & R & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & Q & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & R & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & \dots & Q & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & R & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & Q \end{pmatrix} \begin{pmatrix} x_k \\ u_k \\ x_{k+1} \\ u_{k+1} \\ \vdots \\ x_{k+N-1} \\ u_{k+N-1} \\ x_{k+N} \end{pmatrix}
$$

- we can't solve the multiple shooting formulation with a simple matrix inverse: we have to use a QP solver
- we should use a solver suited for sparse problems (e.g., OSQP, HPIPM, PROXQP, ...)

terminal constraints

- adding a **terminal constraint** to the MPC often provides stronger theoretical properties

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

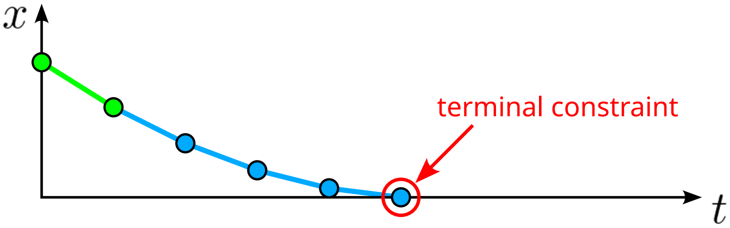
- in this example, the terminal constraint forces the end of **every prediction** to be at the origin

terminal constraints

- having a terminal constraint doesn't mean that the system will reach the origin in finite time! let's see why

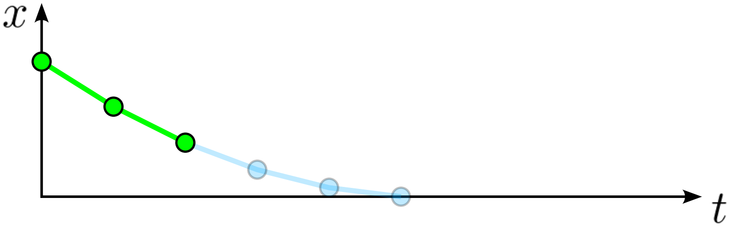
terminal constraints

- at time $t = 1$ we predict a trajectory that reaches the origin at time $t = 5$



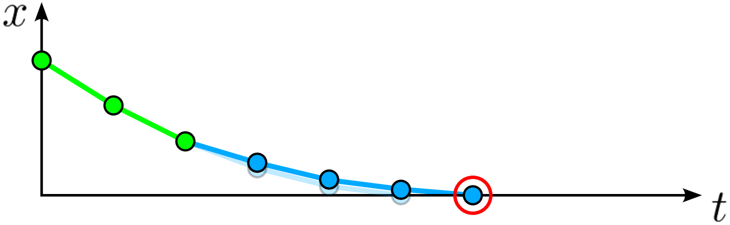
terminal constraints

- we apply the first control input and reach the first predicted state (assuming no disturbances and model mismatch)



terminal constraints

- the next prediction will reache the origin at time $t = 6$, because the terminal constraint keeps shifting forward



terminal constraints

- $x_N = 0$ 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