- [ ] MPC theory
- [ ] MPC demo
- [ ] Dynamic Programming

Review: MPC

$$\min_{u_{t:T}} J(x_t, u_{t:T}, t)$$

$$\text{s.t.} \quad x_{s+1} = f_D(x_s, u_s) \quad s \in \{t, \cdots, T\}$$

$$\underline{u} \le u_s \le \bar{u}$$

optimization variables : $\begin{bmatrix} x_t \\ \vdots \\ x_{T+1} \\ u_t \\ \vdots \\ u_T \end{bmatrix}$ $\qquad u^*_{t:T}$

Feedback control

computing $u^*_{t:T}$ using MPC

↓

apply u^*_t to the real system and obtain x_{t+1}

↓

solve MPC again to compute $u^*_{t+1:T}$

↓

apply u^*_{t+1}

↓

⋮

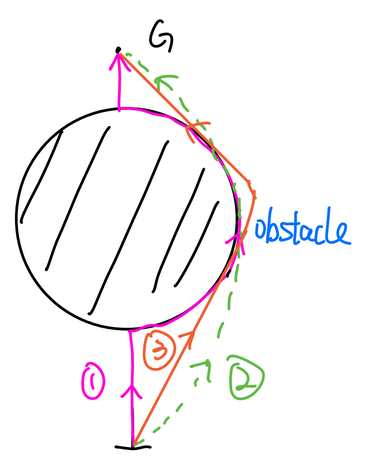
Receding horizon

Planning horizon : $H \ll (T-t)$

Time step	optimization problem
@ time t	$u^*_{t:t+H} = \arg\min_{u_{t:t+H}} J(x_t, u_{t:t+H}, t)$
horizon $= H$	s.t. dynamics and control bounds
	$\downarrow$ apply u^*_t to obtain x_{t+1}

$\vdots$

But solving a different OC problem, sometimes solution becomes greedy.



objective function:

$$J = \sum \text{dist to goal} + \lambda \sum \text{obstacle penetration}$$

$$\text{dist to goal}^2 = (P_x - g_x)^2 + (P_y - g_x)^2$$

$$\text{obstacle penetration} = \begin{cases} -\sqrt{(P_x - O_x)^2 + (P_y - O_y)^2} + R \\ 0 \quad \text{if outside obstacle} \end{cases}$$

Dynamics :

$$\dot{P}_x = V_x$$

$$\dot{P}_y = V_y$$

$$P_x(t+1) = P_x(t) + dt \cdot V_x(t)$$

$$P_y(t+1) = P_y(t) + dt \cdot V_y(t)$$

Control bounds:

$$|V_x| , |V_y| \leq 1 \text{ m/s}$$

MPC

Pros

(1) It can account for feedback, dynamics, and state constraints

(2) It can use modern compute to effectively solve OC problem

(3) It can produce behavior that are optimal w.r.t. future

(4) It can handle objective functions

(5) Objective/constraints can be updated online

Cons

(1) can only do discrete time

(2) It requires online compute

(3) A bunch of $\underbrace{\text{parameters}}_{H, \lambda}$ which it's sensitive to

(4) It's suboptimal