Autonomous navigation of unicycle robots using MPC

M. Farina

marcello.farina@polimi.it

Dipartimento di Elettronica e Informazione

Politecnico di Milano

7 June 2016

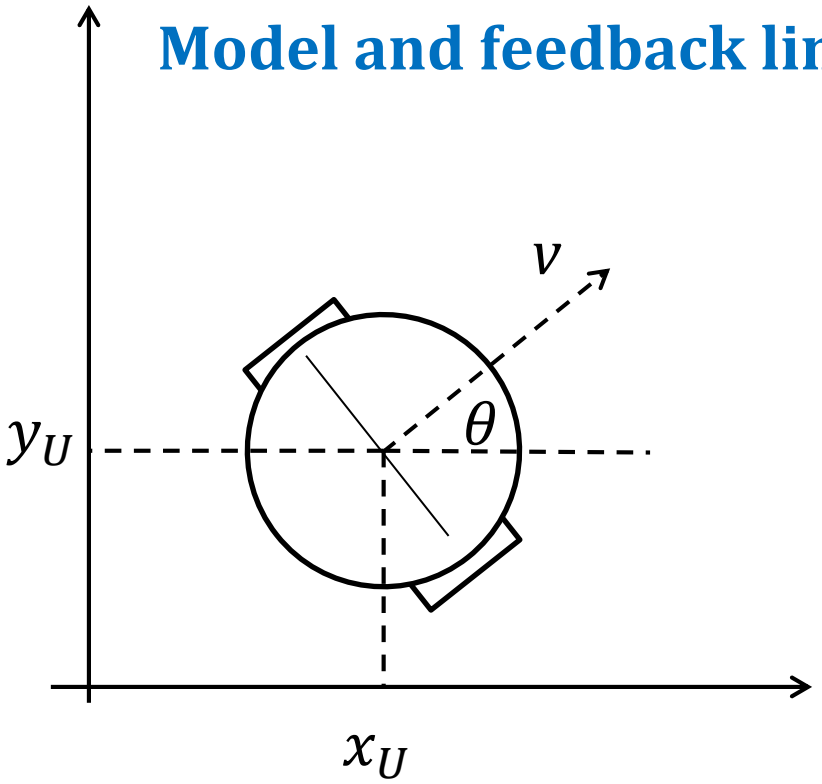
Outline

- Model and feedback linearization
- Goals and constraints
- Some experimental results

Outline

- Model and feedback linearization
- Goals and constraints
- Some experimental results

Model and feedback linearization



$$\begin{cases} \dot{x}_U = v\ \cos\theta \\ \dot{y}_U = v\ \sin\theta \\ \dot{\theta} = \omega \end{cases}$$

Our first goal is to find a way to linearize the system's dynamics or, in a way, to force the system to behave as a linear system.

We resort to ***feedback linearization.***
Two different approaches can be taken.

Model and feedback linearization – first approach

$$\begin{cases} \dot{x}_U = v\ \cos\theta \\ \dot{y}_U = v\ \sin\theta \\ \dot{\theta} = \omega \\ \boxed{\dot{v} = a} \end{cases} \Rightarrow \begin{cases} \dot{x}_U = v_x \\ \dot{y}_U = v_y \\ \dot{v}_x = \cos\theta\dot{v} - v\sin\theta\,\dot{\theta} = \cos\theta a - v\sin\theta\,\omega \\ \dot{v}_y = \sin\theta\dot{v} + v\cos\theta\,\dot{\theta} = \sin\theta a + v\cos\theta\,\omega \end{cases}$$

We define $\begin{aligned} a_x &= \cos\theta a - v\sin\theta\,\omega \\ a_y &= \sin\theta a + v\cos\theta\,\omega \end{cases}$ and the model becomes

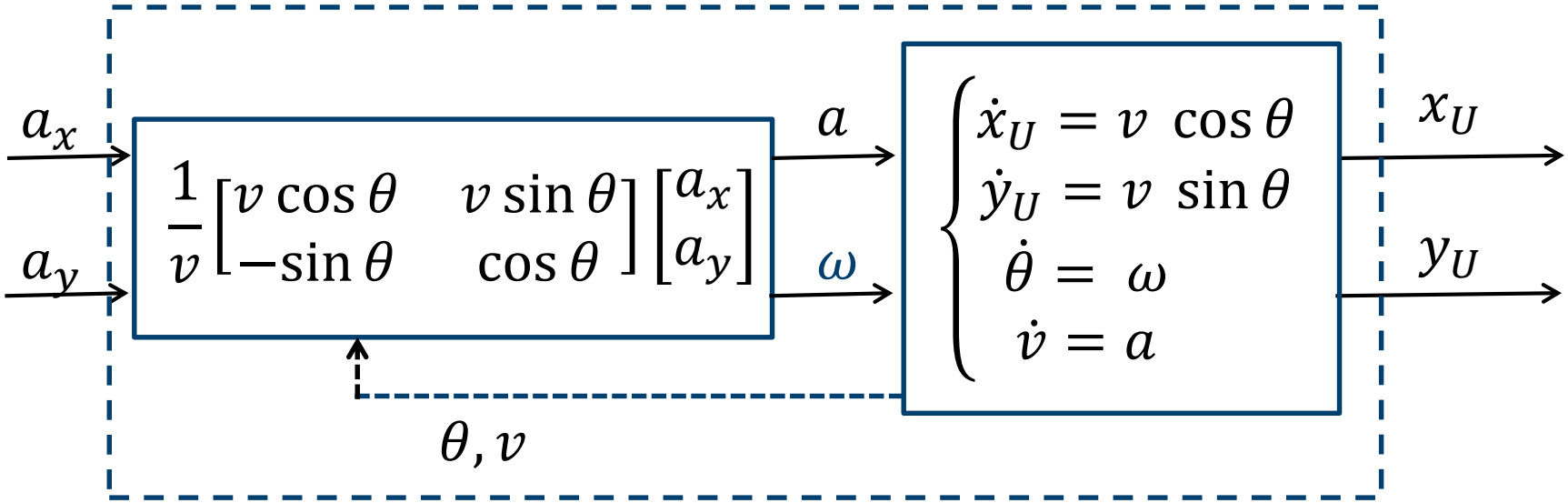
$$\begin{cases} \dot{x}_U = v_x \\ \dot{y}_U = v_y \\ \dot{v}_x = \boxed{a_x} \\ \dot{v}_y = \boxed{a_y} \end{cases}$$

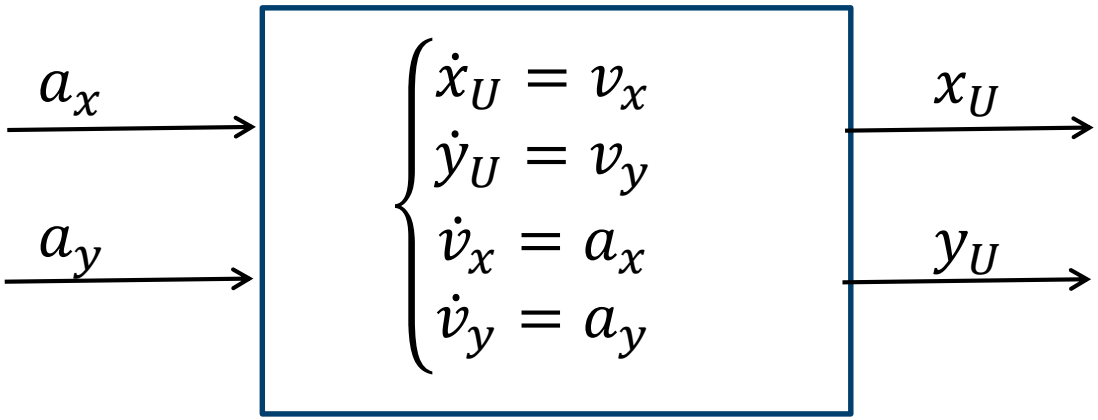
Inputs of the linearized model

$$\begin{bmatrix} a \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & -v\sin\theta \\ \sin\theta & v\cos\theta \end{bmatrix}^{-1} \begin{bmatrix} a_x \\ a_y \end{bmatrix} = \frac{1}{v}\begin{bmatrix} v\cos\theta & v\sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} a_x \\ a_y \end{bmatrix}$$

Real inputs of the unicycle

Model and feedback linearization – first approach





Model and feedback linearization – first approach

Feedback-linearized model:

$$\begin{cases} \dot{x}_U = v_x \\ \dot{y}_U = v_y \\ \dot{v}_x = a_x \\ \dot{v}_y = a_y \end{cases}$$

Discrete time feedback-linearized model (sampling time τ)

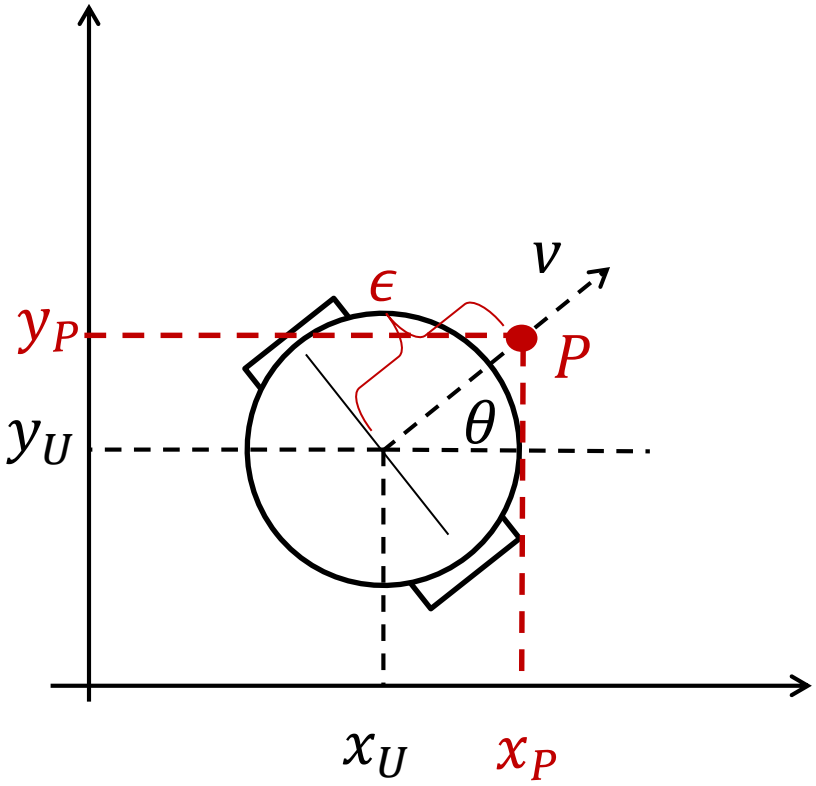
$$\begin{bmatrix} x_U(k+1) \\ v_x(k+1) \\ y_U(k+1) \\ v_y(k+1) \end{bmatrix} = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_U(k) \\ v_x(k) \\ y_U(k) \\ v_y(k) \end{bmatrix} + \begin{bmatrix} \tau^2/2 & 0 \\ \tau & 0 \\ 0 & \tau^2/2 \\ 0 & \tau \end{bmatrix} \begin{bmatrix} a_x(k) \\ a_y(k) \end{bmatrix}$$

$$x(t+1) = Ax(t) + Bu(t)$$

Output: position

$$\begin{bmatrix} x_U(k) \\ y_U(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_U(k) \\ v_x(k) \\ y_U(k) \\ v_y(k) \end{bmatrix} = Cx(k)$$

Model and feedback linearization – second approach



$$\begin{cases} \dot{x}_U = v\ \cos\theta \\ \dot{y}_U = v\ \sin\theta \\ \dot{\theta} = \omega \end{cases}$$

$$x_P = x_U + \epsilon\cos\theta$$
$$y_P = y_U + \epsilon\sin\theta$$

$$\begin{cases} \dot{x}_P = \dot{x}_U - \epsilon\sin\theta\,\dot{\theta} = v\ \cos\theta - \omega\epsilon\sin\theta \\ \dot{y}_P = \dot{y}_U + \epsilon\cos\theta\,\dot{\theta} = v\ \sin\theta + \omega\epsilon\cos\theta \end{cases}$$

Model and feedback linearization – second approach

$$\begin{cases} \dot{x}_P = v\ \cos\theta - \omega\epsilon \sin\theta \\ \dot{y}_P = v\ \sin\theta + \omega\epsilon \cos\theta \end{cases}$$

We define $\begin{aligned} v_x &= v\ \cos\theta - \omega\epsilon \sin\theta \\ v_y &= v\ \sin\theta + \omega\epsilon \cos\theta \end{aligned}$ and the model becomes

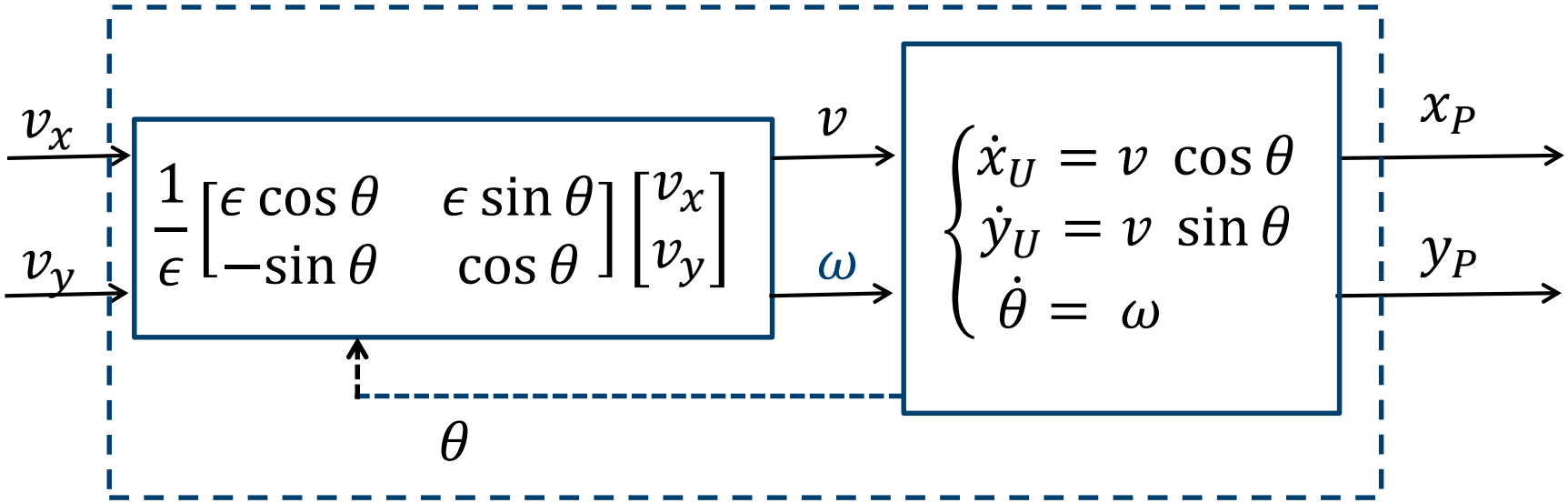
$$\begin{cases} \dot{x}_P = v_y \\ \dot{y}_P = v_y \end{cases}$$

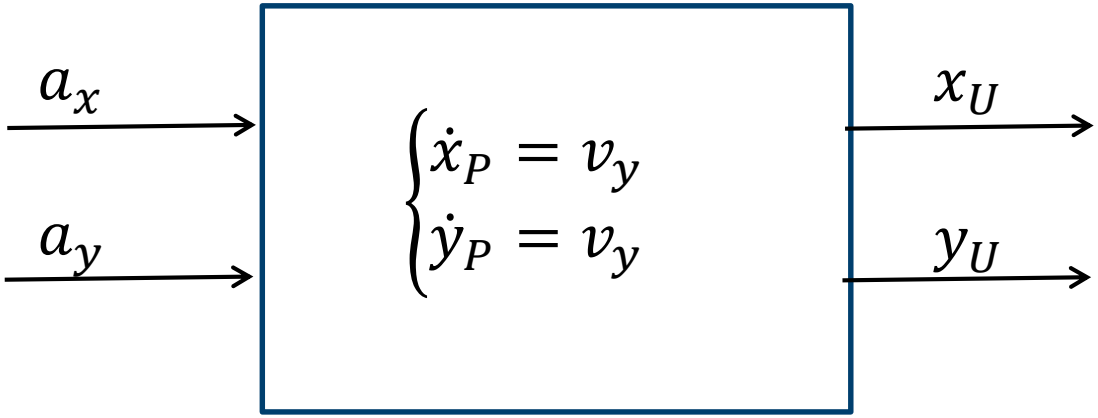
Inputs of the linearized model

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & -\epsilon \sin\theta \\ \sin\theta & \epsilon \cos\theta \end{bmatrix}^{-1} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \frac{1}{\epsilon} \begin{bmatrix} \epsilon \cos\theta & \epsilon \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

Real inputs of the unicycle

Model and feedback linearization – second approach





Model and feedback linearization – second approach

Feedback-linearized model:

$$\begin{cases} \dot{x}_P = v_x \\ \dot{y}_P = v_y \end{cases}$$

Discrete time feedback-linearized model (sampling time τ)

$$\begin{bmatrix} x_U(k+1) \\ y_U(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_U(k) \\ y_U(k) \end{bmatrix} + \begin{bmatrix} \tau & 0 \\ 0 & \tau \end{bmatrix} \begin{bmatrix} v_x(k) \\ v_y(k) \end{bmatrix}$$

$$\boxed{x(t+1) = Ax(t) + Bu(t)}$$

Output: position (i.e., the whole state)

$$y(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_U(k) \\ y_U(k) \end{bmatrix} = Cx(k)$$

Outline

- Model and feedback linearization
- Goals and constraints
- Some experimental results

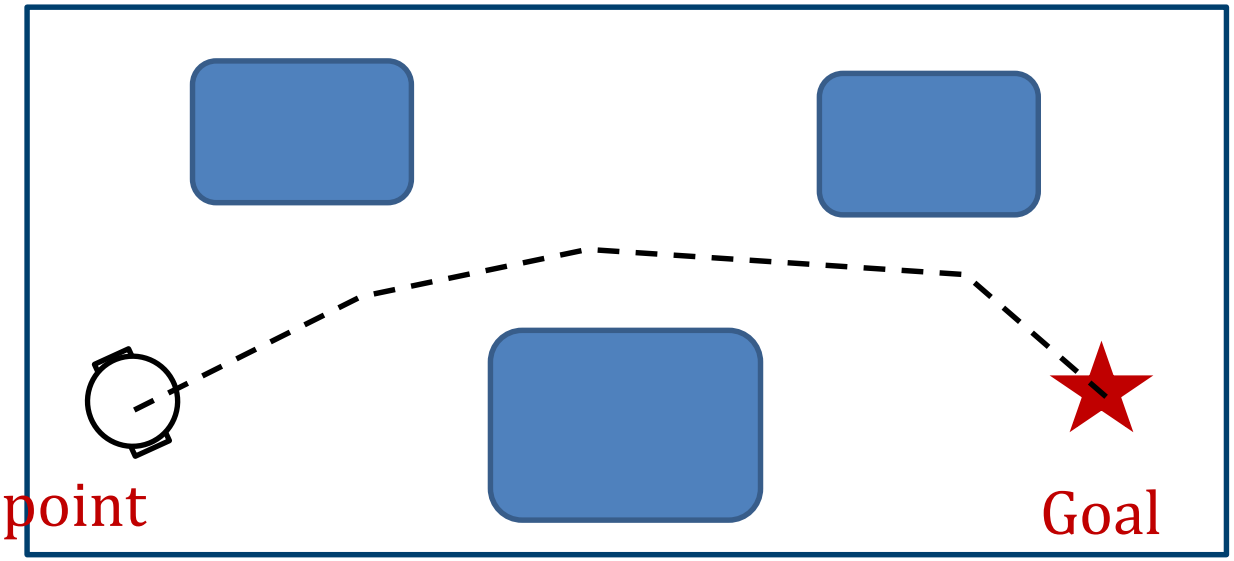
Goals and constraints

With any feedback linearization approach we obtain the linear model

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k)$$

position

We aim to solve the simplest problem, i.e., the *parking problem*

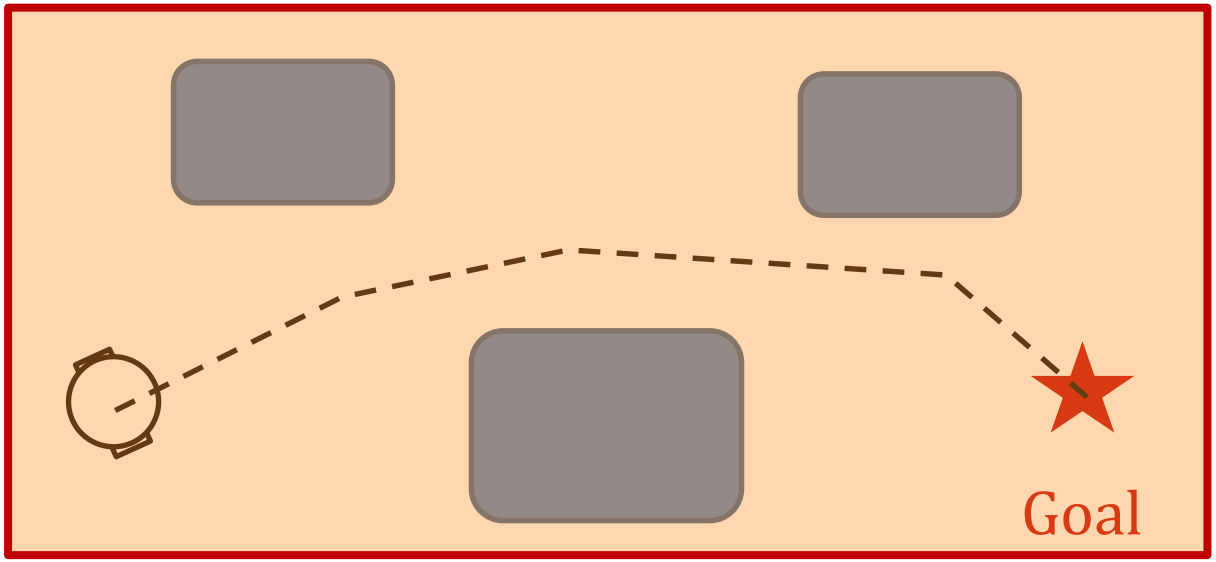


Goals and constraints

To fulfill this, we must include, in the MPC problem, the following ingredients:

- **Constraints**:
  - Collision avoidance constraints with respect to external walls
  - Fixed obstacle avoidance constraints
  - If more vehicles are included, inter-robot collision avoidance constraints
  - Constraints for fulfilling operational limitations (e.g., maximum speed, etc)
- **Cost function**:
  - For approaching the goal position
  - To prevent deadlock solutions with fixed obstacles
- Proper **terminal cost function and terminal constraints** for guaranteeing recursive feasibility.

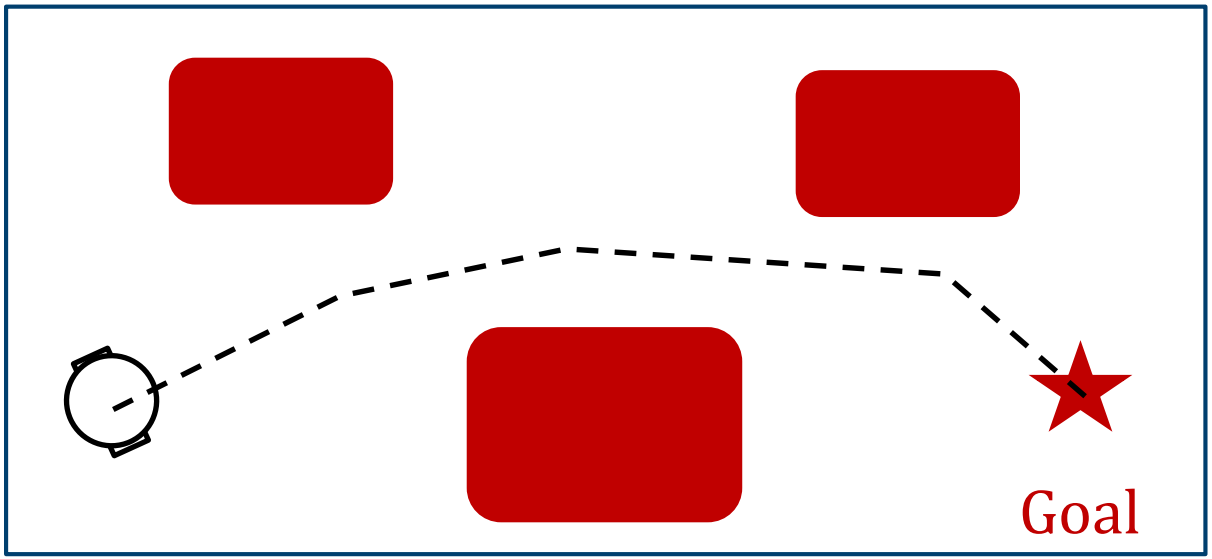
Collision avoidance constraints with respect to external walls



If the red area is polytopic and convex, we can write the corresponding constraint as follows:

$$A_{area} y(k) \leq b_{area} \quad \Rightarrow \quad A_{area} C x(k) \leq b_{area}$$

Collision avoidance constraints with respect to (fixed) obstacles



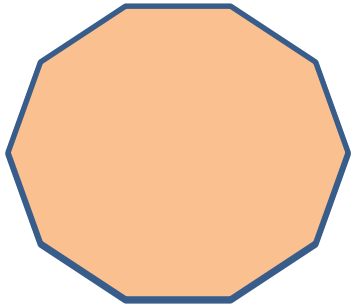
The problem is that, to avoid obstacles of this type, the «free area» is not convex, and so we cannot express the problem using constraints of the type:

$$A_{coll-avoidance} y(k) \leq b_{coll-avoidance}$$

Collision avoidance constraints with respect to (fixed) obstacles

Steps:

1. Assume that it is possible to circumscribe each obstacle with a polytope with L sides (better if L is large). The interior of the obstacle can be represented as

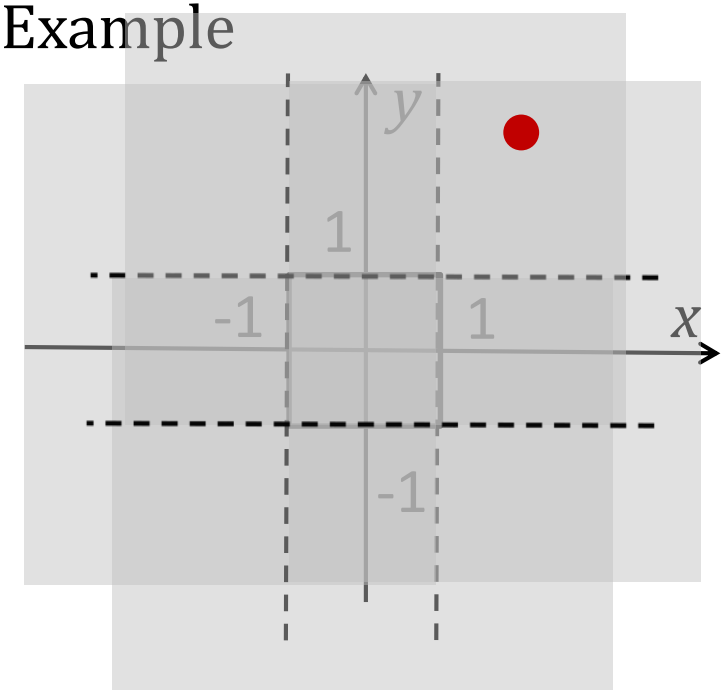


$$A_{obst} y \leq b_{obst} \rightarrow L \text{ inequalities, all satisfied}$$

2. Note that a collision-free point is such that at least one of the above L inequalities is violated, i.e., at least one of the elements of the vector $\rho = A_{obst} y - b_{obst}$ is positive

Collision avoidance constraints with respect to (fixed) obstacles

Example



The interior of the square is characterized by L=4 inequalities (intersection of half planes)

$$\begin{matrix} x \le 1 \\ -x \le 1 \\ y \le 1 \\ -y \le 1 \end{matrix} \Rightarrow \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \le \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Pick an external point, e.g., $(\bar{x}, \bar{y})$=(1.5,2). Which inequalities are violated? The vector

$$\rho = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -2.5 \\ 1 \\ -3 \end{bmatrix}$$

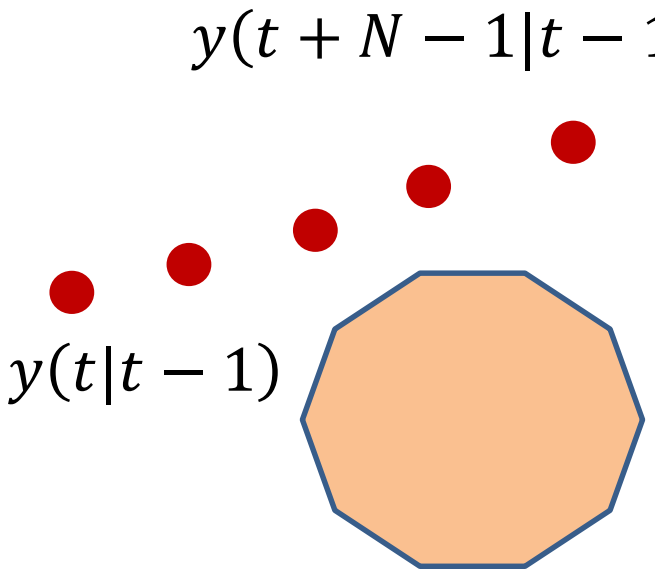
has two positive entries.

⇐ This is the «most violated».

Collision avoidance constraints with respect to (fixed) obstacles

3. In the MPC context, we can take advantage of the fact that, at time t, the optimal **collision-free** trajectory computed at the previous time instant t-1 is available, i.e.,

$u(t-1|t-1), u(t|t-1), \dots, u(t+N-2|t-1)$ Feasible inputs
$x(t|t-1), \dots, x(t+N-1|t-1)$ States
$y(t|t-1), \dots, y(t+N-1|t-1)$ Positions



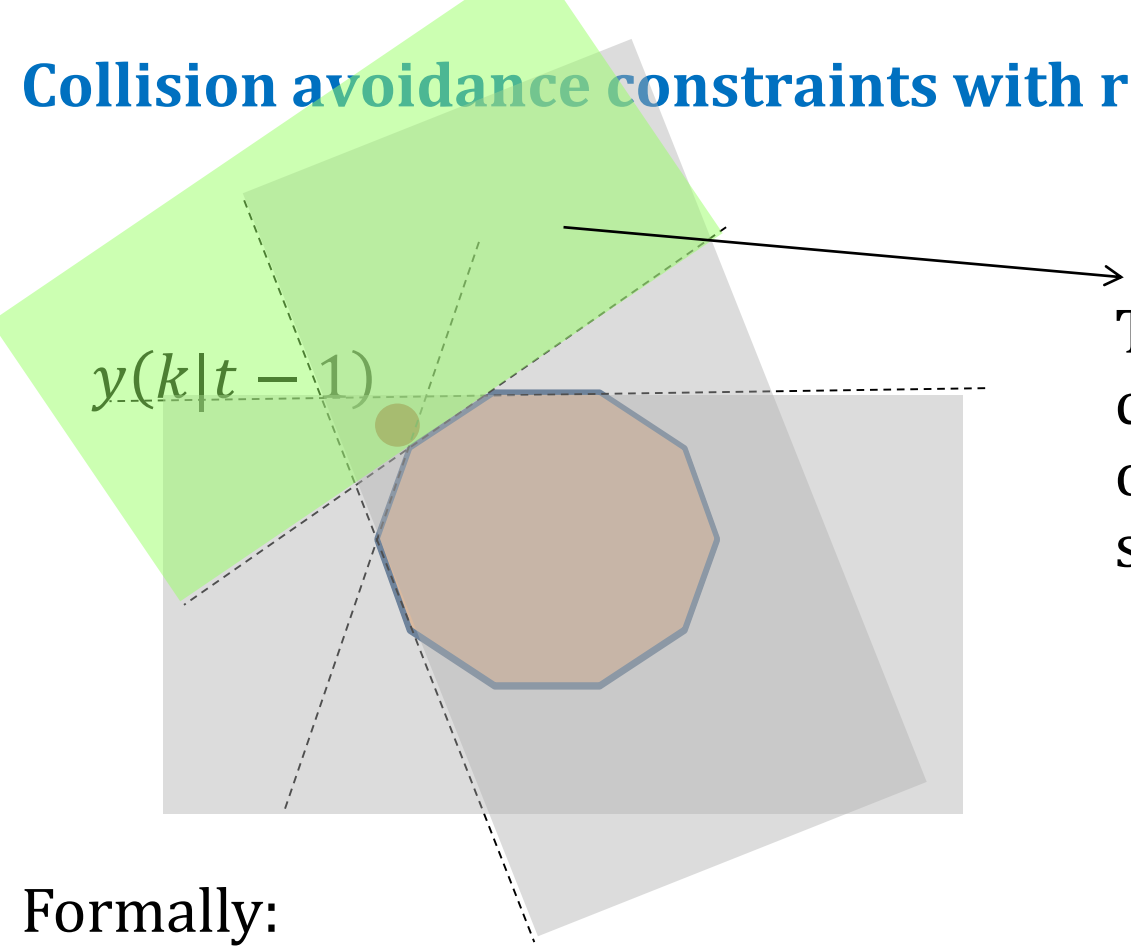
Since the trajectory is collision-free, for all k=t,…,t+N-1, at least one entry of the vector

$$\rho(k|t-1) = A_{obst}y(k|t-1) - b_{obst}$$

is positive.

We select the corresponding constraint to be enforced (at time t) on $y(k)$.

Collision avoidance constraints with respect to (fixed) obstacles



This **convex set** defines the constraint for $y(k)$ in the optimization problem solved at time t.

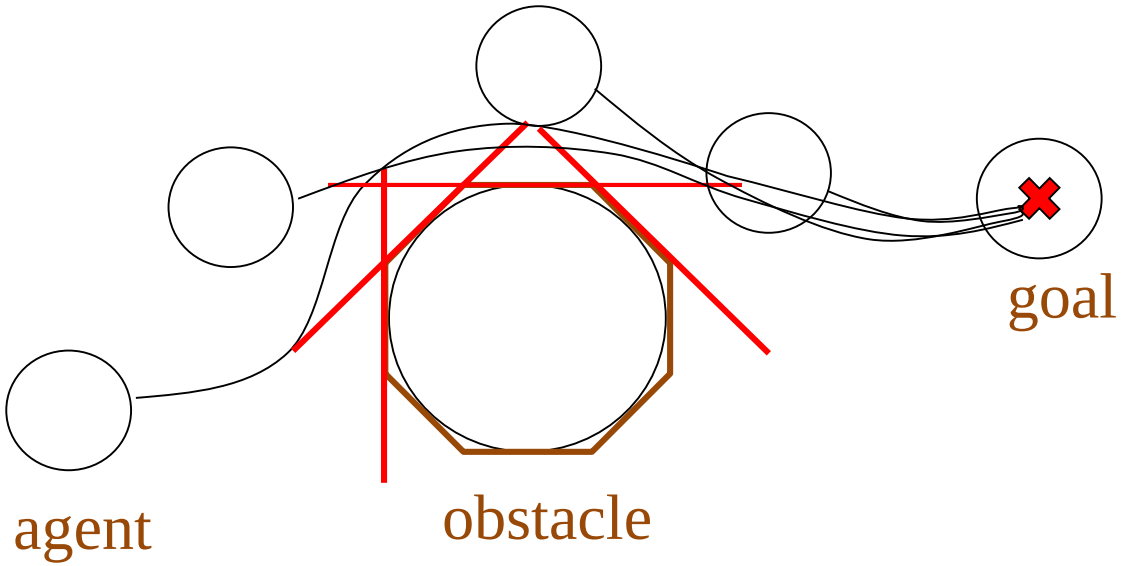
Formally:

- We compute, for all k=t,…,t+N-1 the vector

$$\rho(k|t-1) = A_{obst}y(k|t-1) - b_{obst}$$

- We select $\bar{\imath}$ such that the entry $\rho_{\bar{\imath}}(k|t-1) \geq \rho_i(k|t-1)$, i=1,…,L.
- We impose only the constraint $A_{obst,\bar{\imath}}y(k) \geq b_{obst,\bar{\imath}}$

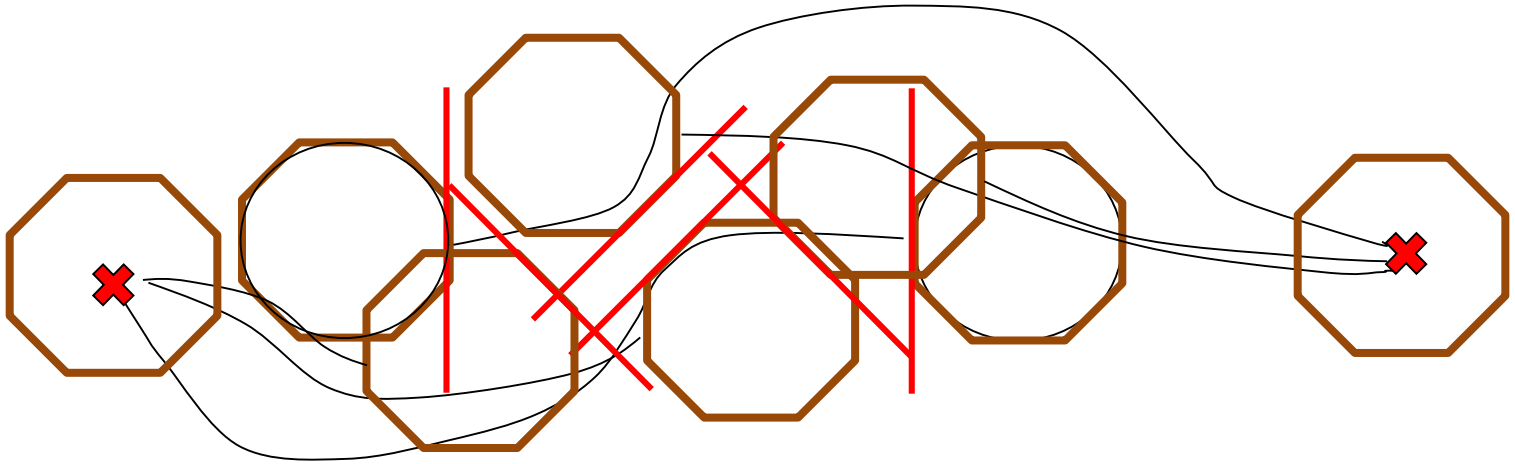
Collision avoidance constraints with respect to (fixed) obstacles



Inter-robot collision avoidance constraints

If two vehicles are present, we aim to generate collision-free trajectories. We exploit communication!

- Each agent is regarded, by the other agent, as a moving obstacle.
- The «most violated» constraint is selected (similarly to the previous case) based on the collision-free trajectory predicted at the previous optimization step.
- Each agent is allowed to move towards the constraint half the distance, to avoid collision.



Operational limitations

Consider, for example, the feedback-linearized model obtained with the first approach, i.e.,

$$\begin{bmatrix} x_U(k+1) \\ v_x(k+1) \\ y_U(k+1) \\ v_y(k+1) \end{bmatrix} = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_U(k) \\ v_x(k) \\ y_U(k) \\ v_y(k) \end{bmatrix} + \begin{bmatrix} \tau^2/2 & 0 \\ \tau & 0 \\ 0 & \tau^2/2 \\ 0 & \tau \end{bmatrix} \begin{bmatrix} a_x(k) \\ a_y(k) \end{bmatrix}$$

Velocities (in x and y directions) are states

Constraints on these variables may be used to enforce constraints on the longitudinal speed v.

Accelerations (in x and y directions) are inputs

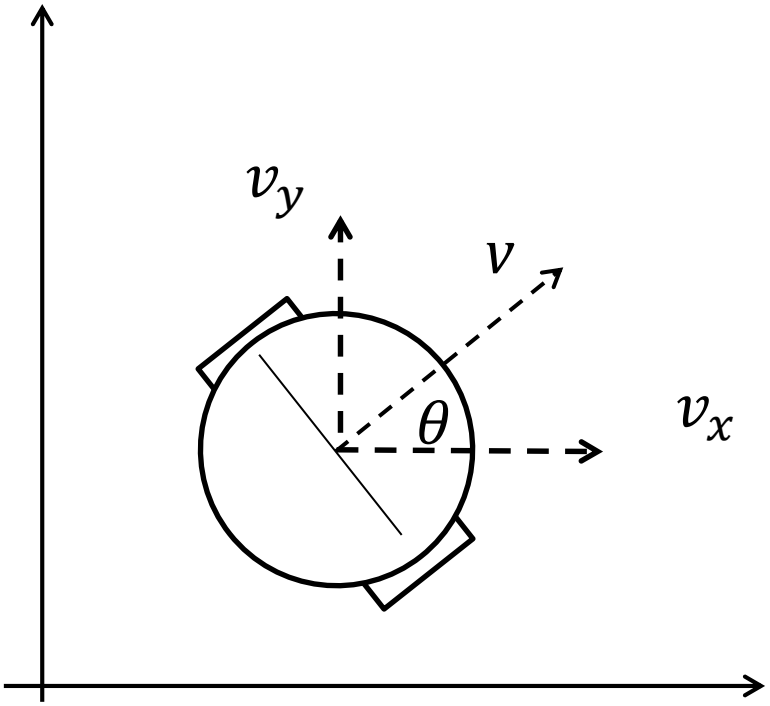
Constraints on these variables may be used to enforce constraints on the longitudinal acceleration a.

Operational limitations

We want to enforce:

$$-v_M \leq v \leq v_M$$

This can be done, for example, by imposing:



$$-\frac{v_M}{\sqrt{2}} \leq v_x(k) \leq \frac{v_M}{\sqrt{2}}$$

$$-\frac{v_M}{\sqrt{2}} \leq v_y(k) \leq \frac{v_M}{\sqrt{2}}$$

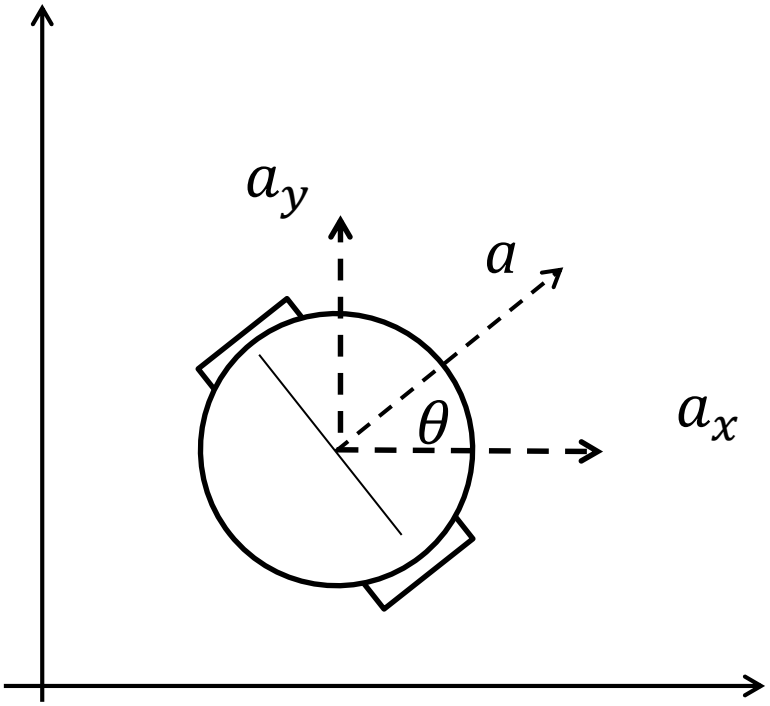
$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} x(k) \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \frac{v_M}{\sqrt{2}}$$

Operational limitations

We want to enforce:

$$-a_M \leq a \leq a_M$$

This can be done, for example, by imposing:



$$-\frac{a_M}{\sqrt{2}} \leq a_x(k) \leq \frac{a_M}{\sqrt{2}}$$

$$-\frac{a_M}{\sqrt{2}} \leq a_y(k) \leq \frac{a_M}{\sqrt{2}}$$

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} u(k) \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \frac{a_M}{\sqrt{2}}$$

Cost function for approaching the goal position

- The goal position is $\bar{y}_G$.

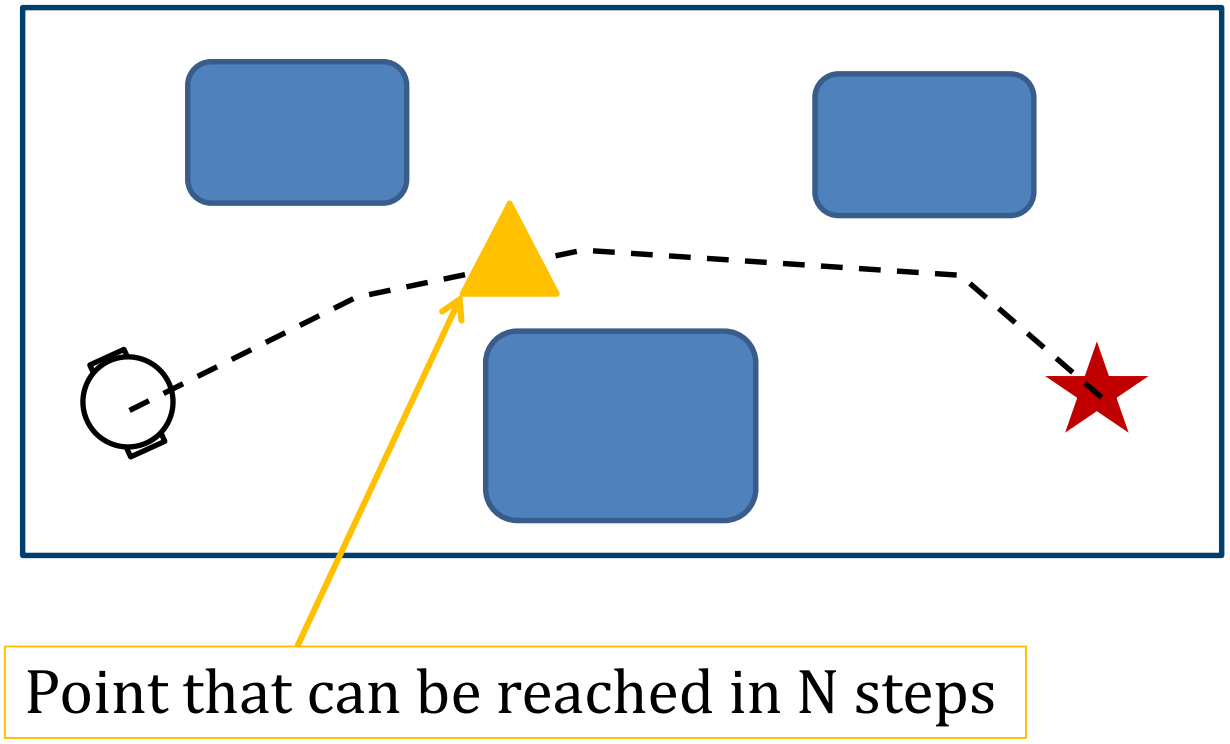
- The naive approach consists in minimizing, at time *t*, the following cost function:

$$J = \sum_{k=t}^{t+N-1} \|y(k) - \bar{y}_G\|_Q^2 + \|u(k)\|_R^2 + \|x(t+N) - \bar{x}_G\|_P^2$$

where $\bar{x}_G = \begin{bmatrix} \bar{y}_{G,x} \\ 0 \\ \bar{y}_{G,y} \\ 0 \end{bmatrix}$ (we aim to rest at the goal)

Cost function for approaching the goal position

However, assume that, in *N* steps not all the trajectory to the goal can be covered



In such case is more reasonable to define temporary goals

Cost function for approaching the goal position

An alternative approach, indeed, consists in the tracking one:

$$J = \sum_{k=t}^{t+N-1} \|y(k) - y_G(t)\|_Q^2 + \|u(k)\|_R^2 + \|x(t+N) - x_G(t)\|_P^2 + \gamma \|y_G(t) - \bar{y}_G\|^2$$

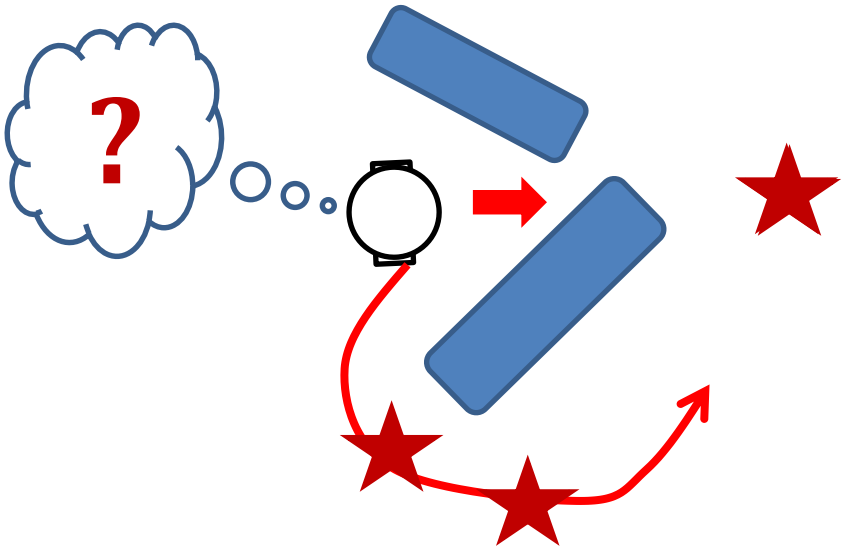
temporary goal (for the whole trajectory), additional degree of freedom

Other approaches can be taken depending also on the specific problem at hand (parking, trajectory tracking, etc.)

Cost function to prevent deadlock solutions with fixed obstacles

So far collision avoidance is enforced just by using hard constraints.

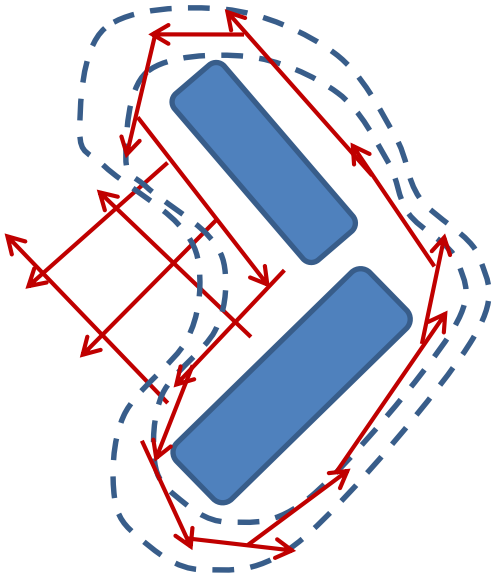
Deadlock situations may occurr!



- Without additional terms, the cost function pushes the robot towards the bottleneck…
- An additional «force» is necessary to circumnavigate the obstacle
- Temporary goals can be set using the theory of **vortex fields**

Cost function to prevent deadlock solutions with fixed obstacles

Two words on vortex fields



- If obstacles «generate» repulsive fields local minima (i.e., bottlenecks) may occur
- Instead, if repulsive forces are «rotated», i.e., they are directed in tangent direction with respect to the obstacle, circumnavigation is guaranteed.

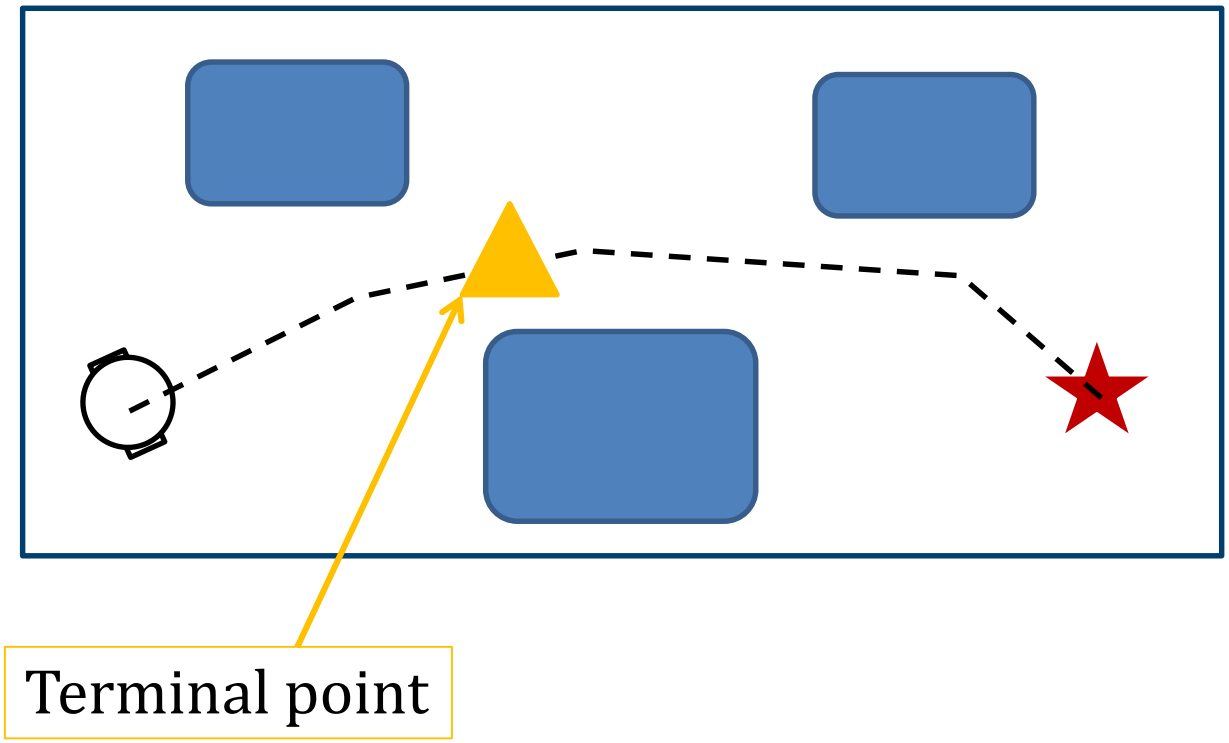
A fictitious goal $x_{VF}(t)$ is generated in the direction of the vector field and an additive term

$$\|x(k) - x_{VF}(t)\|^2$$

appears in the stage cost

Terminal constraints

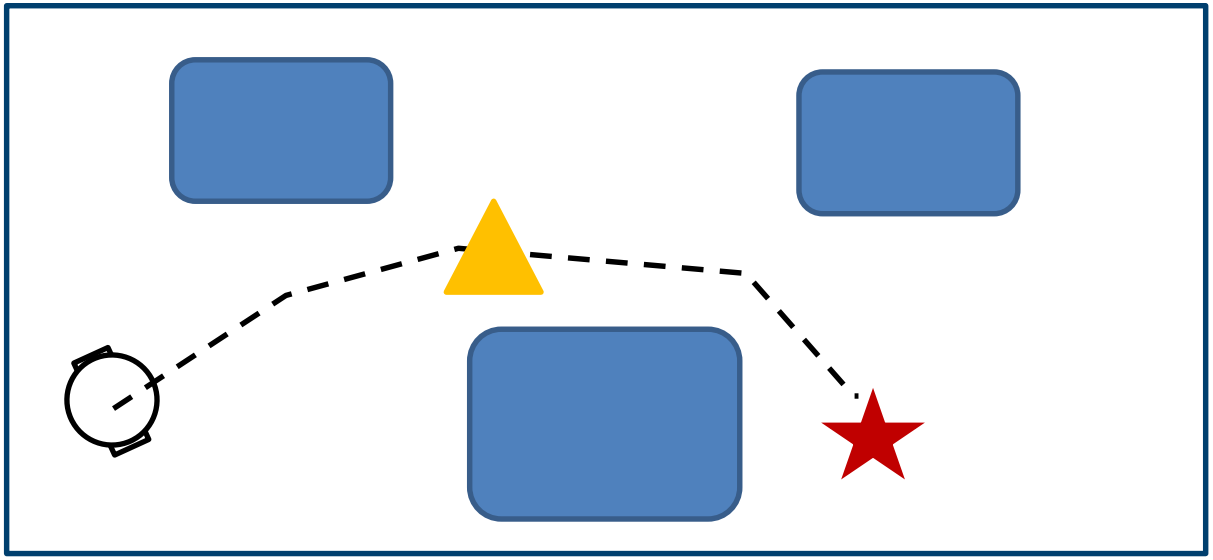
Assume again that, in N steps, not all the trajectory to the goal can be covered



Recursive fesibility can be compromized if this point is not in a «safe region»

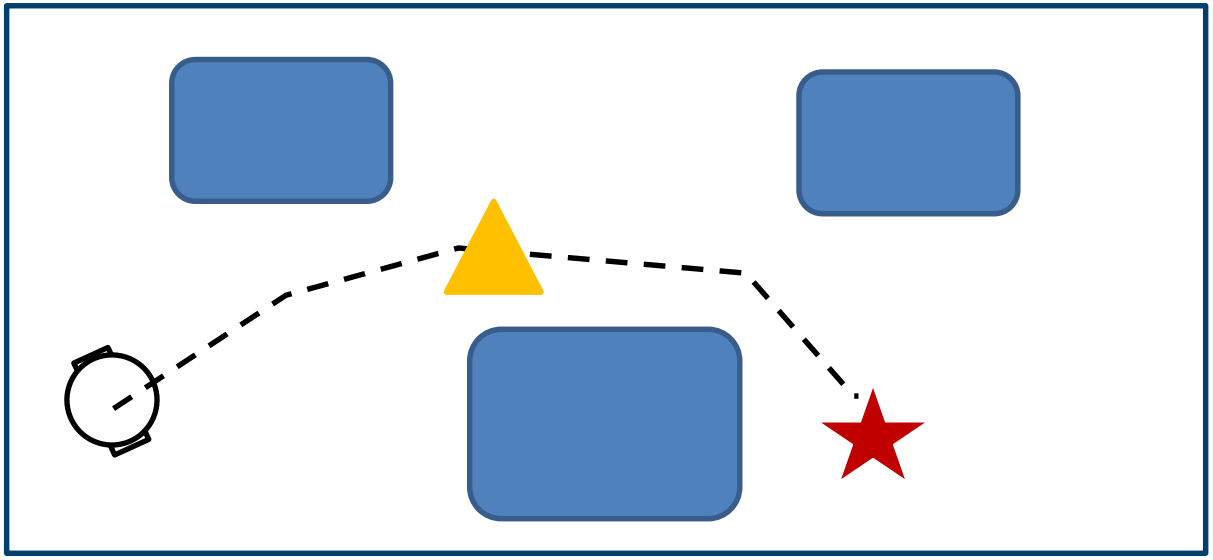
Terminal constraints

In regulation problems, the «safe region» is a region is characterized by the fact that, by using the **auxiliary control law**, we go towards the goal without colliding with obstacles or other vehicles and without violating the operational constraints



That's not possible in this case!

Terminal constraints



To guarantee theoretical properties we may rely on the «tracking» approach, with terminal point.

Basically, we require that, after N steps, we may be able to **stop at the intermediate goal, defined as the further degree of freedom for our MPC optimization problem.**

This is cast as the linear equality terminal constraint:

$$x(t+N) = x_G(t)$$

Goals and constraints - summary

Ingredients:

- **Constraints**:
  - Collision avoidance constraints with respect to external walls → Linear inequality (time invariant) constraint
  - Fixed obstacle avoidance constraints → Linear inequality (time-varying) constraint
  - If more vehicles are included, inter-robot collision avoidance constraints → Linear inequality (time-varying) constraint
  - Constraints for fulfilling operational limitations (e.g., maximum speed, etc) → Linear inequality (time invariant) constraint
- **Cost function**:
  - For approaching the goal position → Quadratic cost function
  - To prevent deadlock solutions with fixed obstacles → Quadratic additive terms
- Proper **terminal cost function and terminal constraints** for guaranteeing recursive feasibility. → Linear equality constraint

Outline

- Model and feedback linearization
- Goals and constraints
- Some experimental results

Some experimental results

Some experimental results

Some experimental results