Applicazioni dell'Automatica

Introduction to mobile robotics: Planning and control for WMRs

Prof. Giuseppe Oriolo

Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti

Sapienza Università di Roma

outline of this lecture

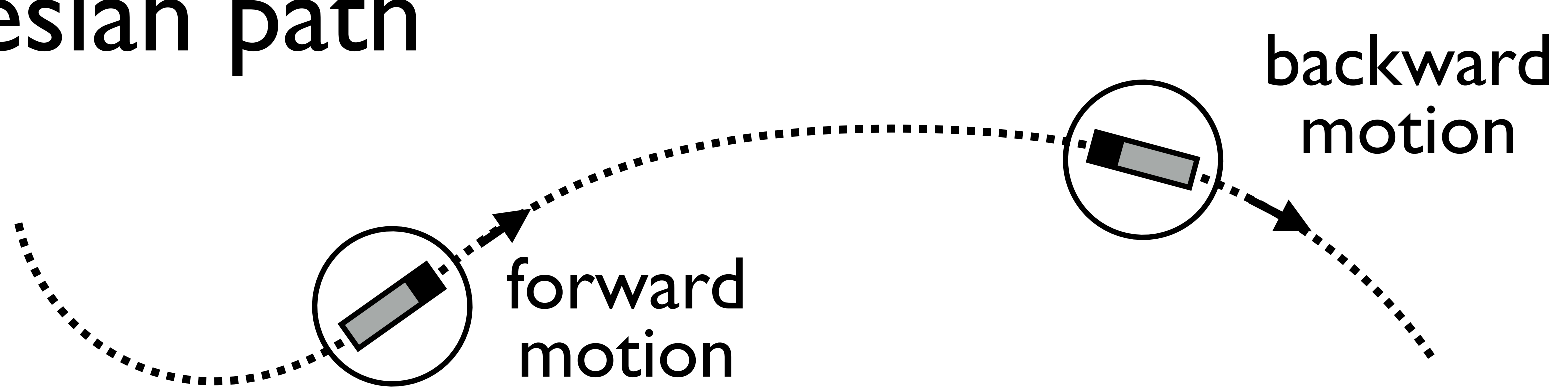
- path/trajectory planning (no obstacles)
- motion planning (among obstacles)
- the RRT algorithm
- motion control problems
- trajectory tracking via approximate linearization
- trajectory tracking via input-output linearization
- Cartesian regulation

Path/trajectory planning

- throughout these slides we consider a unicycle
- assume we want to plan a trajectory between a start configuration $\boldsymbol{q}_s$ and a goal configuration $\boldsymbol{q}_g$ in the absence of obstacles
- from the kinematic model of the unicycle we have

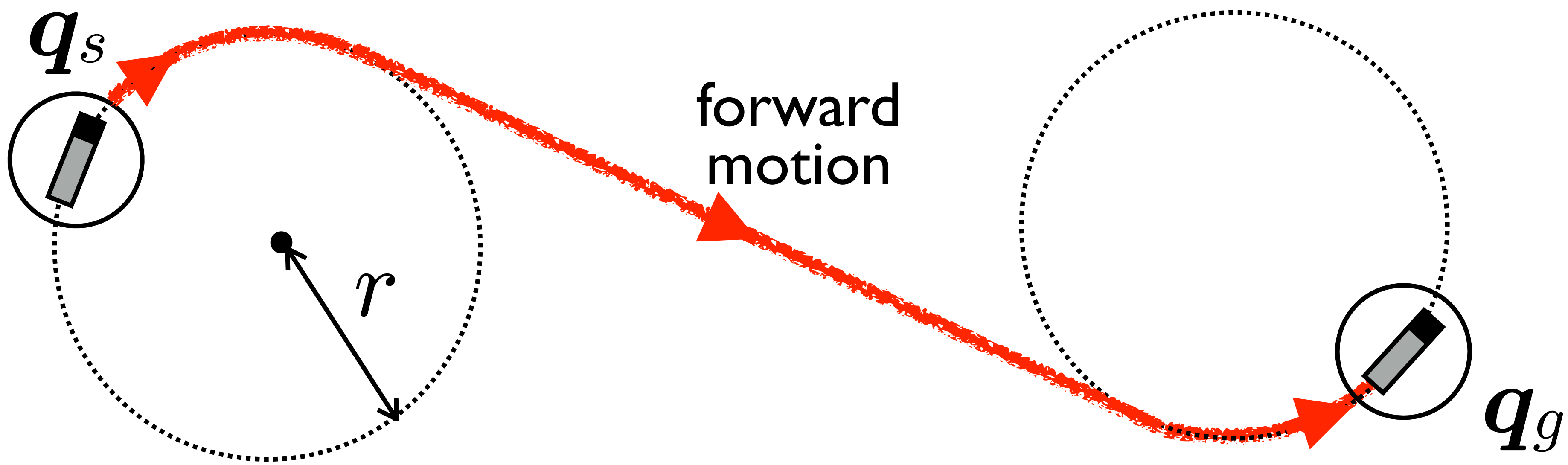
$$\theta = \mathrm{ATAN2}(\dot{y}, \dot{x}) + k\,\pi \qquad k = 0, 1$$

- this means that the unicycle must always be tangent to the Cartesian path



Path/trajectory planning

- based on this, it is easy to plan a feasible path/trajectory from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$
- for example, use circular-linear-circular (CLC) paths



- once a CLC path has been chosen, one may choose a profile for v along it, e.g., trapezoidal over time
- then, the profile of ω can be easily derived considering that $\omega = v/r$ along C tracts and $\omega = 0$ on L tracts
- as a consequence, a trajectory has been defined

Motion planning

- assume that we want to plan a trajectory between a start configuration $\boldsymbol{q}_s$ and a goal configuration $\boldsymbol{q}_g$, now in the presence of obstacles
- suppose a geometrical description of the obstacles is given as subsets of the workspace
- let us first ignore the pure rolling constraint, so that the kinematic model of the robot is

$$\dot{\boldsymbol{q}} = \boldsymbol{u} \qquad \text{or} \qquad \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}$$

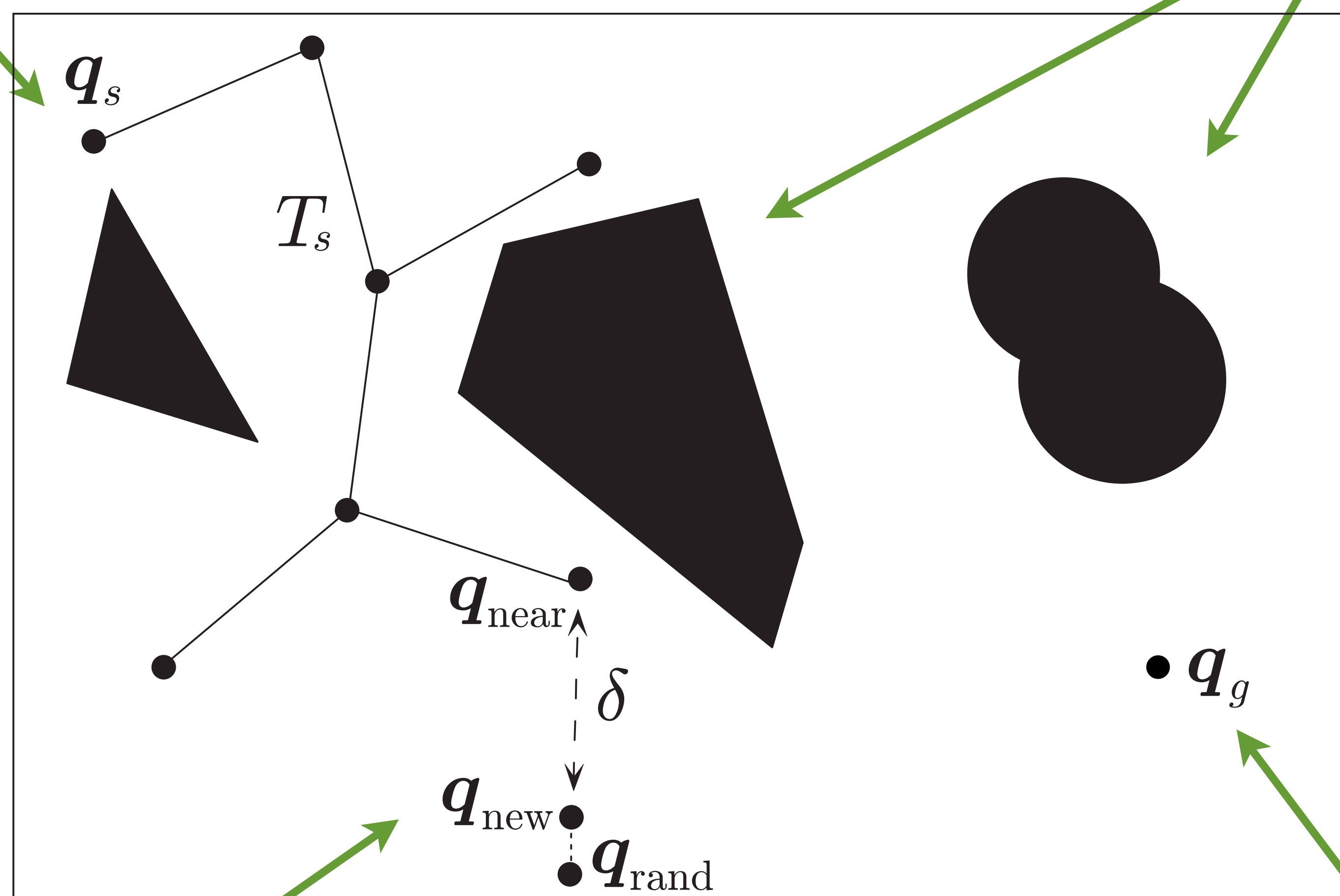
this means that the robot can instantaneously move in any direction of the configuration space (free-flying)

RRT (Rapidly-exploring Random Tree)

- modern motion planning is based on the idea of randomly sampling the configuration space $\mathcal{C}$, checking each sample for collision and connecting collision-free samples in a network
- a very popular algorithm: RRT
- basic iteration to build a tree T_s rooted at $\boldsymbol{q}_s$:
  - generate $\boldsymbol{q}_{\rm rand}$ in $\mathcal{C}$ with uniform probability distribution
  - search the tree for the nearest configuration $\boldsymbol{q}_{\rm near}$
  - choose $\boldsymbol{q}_{\rm new}$ at a distance δ from $\boldsymbol{q}_{\rm near}$ in the direction of $\boldsymbol{q}_{\rm rand}$
  - check for collision $\boldsymbol{q}_{\rm new}$ and the segment from $\boldsymbol{q}_{\rm near}$ to $\boldsymbol{q}_{\rm new}$
  - if check is negative, add $\boldsymbol{q}_{\rm new}$ to T_s (expansion)

tree is rooted at $\boldsymbol{q}_s$

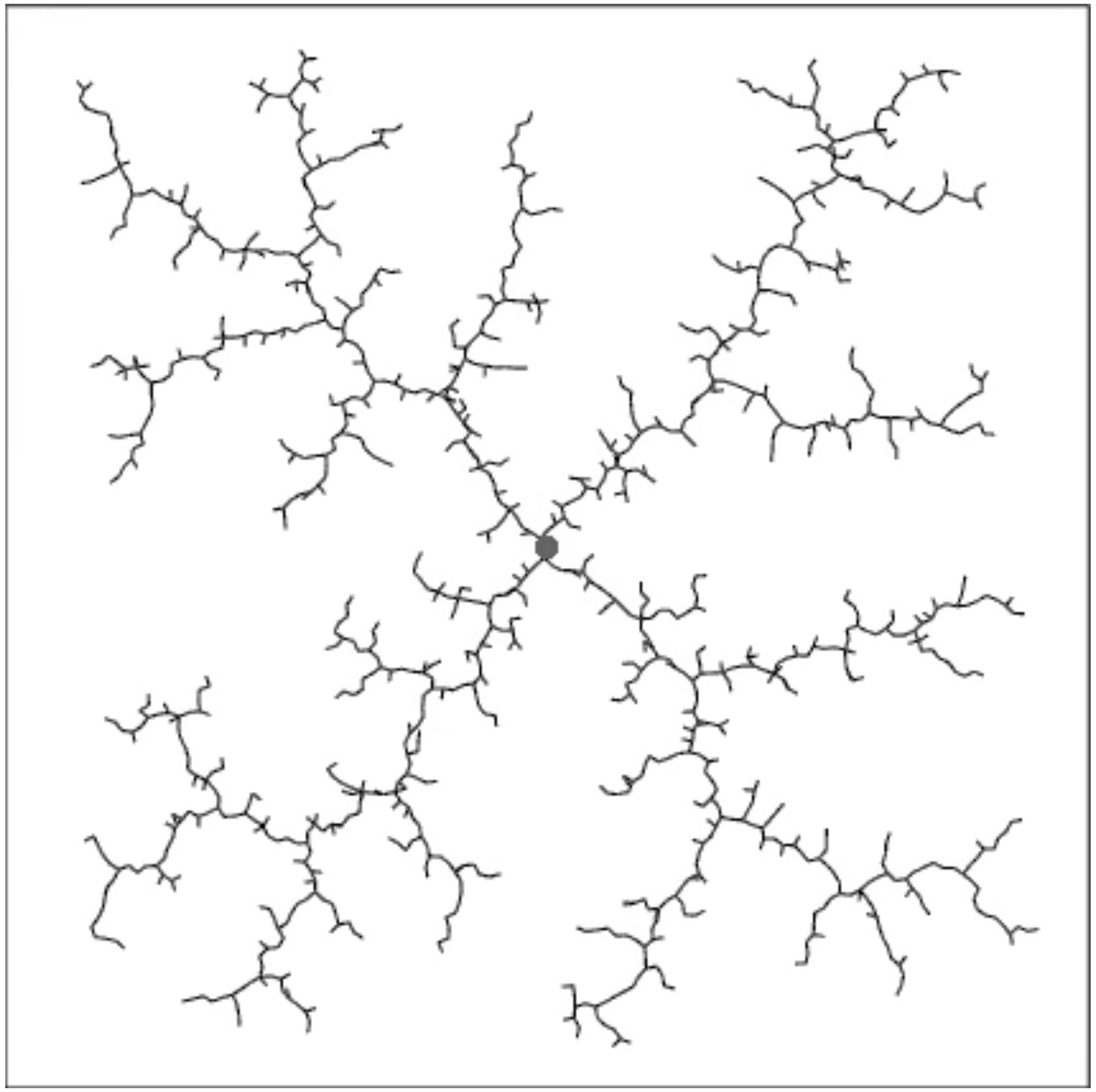
obstacle images in $\mathcal{C}$ are never computed



tree expansion

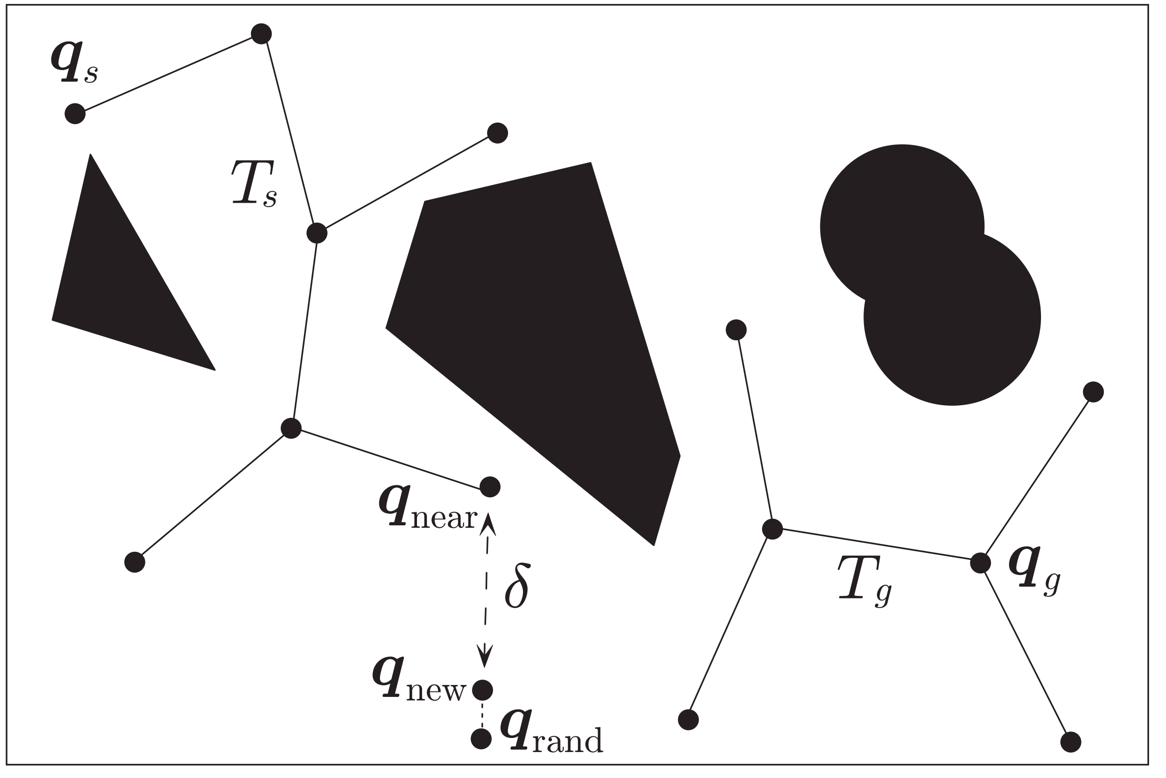
no bias towards $\boldsymbol{q}_g$

RRT in empty 2D space



- explores all areas very quickly because the expansion is biased towards the unexplored areas (to be precise, towards larger Voronoi regions)

- to introduce a bias towards $\boldsymbol{q}_g$, grow two trees T_s and T_g, respectively rooted at $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ (bidirectional RRT)
- alternate expansion and connection phases: use the last generated $\boldsymbol{q}_{\rm new}$ of T_s as a $\boldsymbol{q}_{\rm rand}$ for T_g, and then repeat switching the roles of T_s and T_g



- bidirectional RRT is probabilistically complete: over the iterations, the probability of finding a collision-free path from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$ (provided that it exists) tends to 1

RRT: extension to nonholonomic robots

- motion planning for a unicycle in $\mathcal{C} = \mathrm{R}^2 \times SO(2)$

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \omega$$

- linear paths in $\mathcal{C}$ such as those used to connect $\boldsymbol{q}_{\mathrm{near}}$ to $\boldsymbol{q}_{\mathrm{rand}}$ are not admissible in general

- one possibility is to use motion primitives, i.e., a finite set of admissible local paths, produced by a specific choice of the velocity inputs

- for example, one may use

$$v = \bar{v} \qquad \omega = \{-\bar{\omega}, 0, \bar{\omega}\} \qquad t \in [0, \Delta]$$

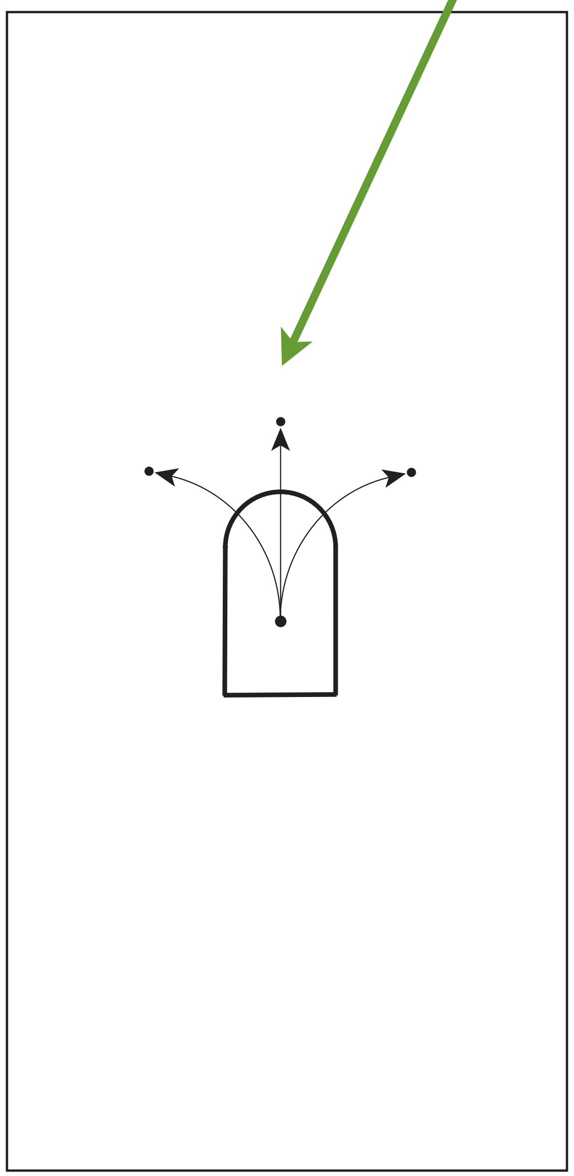
resulting in 3 possible paths in forward motion

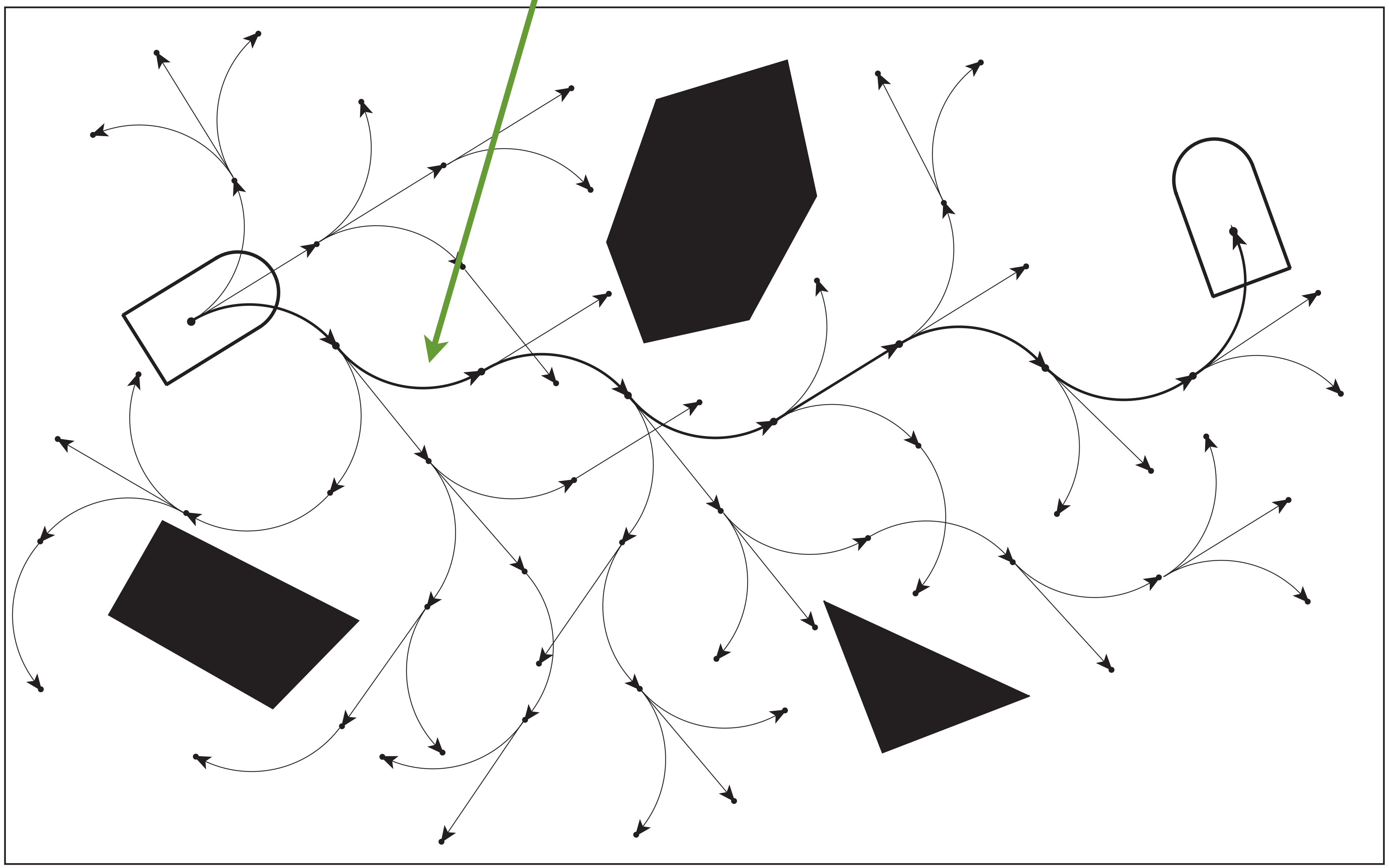
- the algorithm is the same with the only difference that $\boldsymbol{q}_{\rm new}$ is generated from $\boldsymbol{q}_{\rm near}$ selecting one of the possible paths (either randomly or as the one that leads the unicycle closer to $\boldsymbol{q}_{\rm rand}$)

- if $\boldsymbol{q}_g$ can be reached from $\boldsymbol{q}_s$ with a collision-free concatenation of primitives, the probability that a solution is found tends to 1 as the time tends to ∞

primitives

solution path made by concatenation

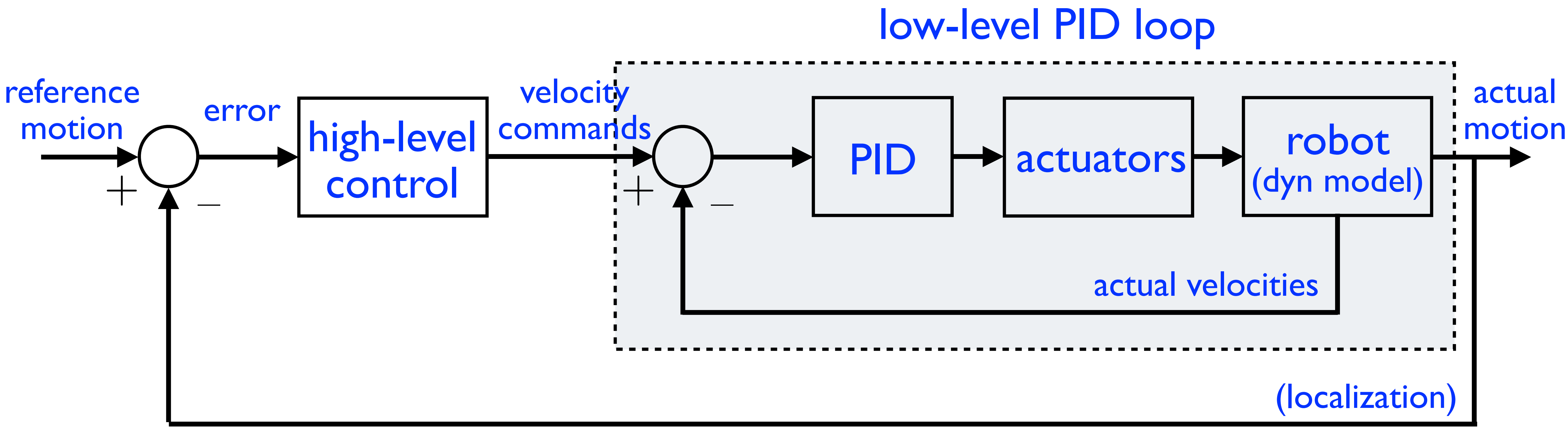




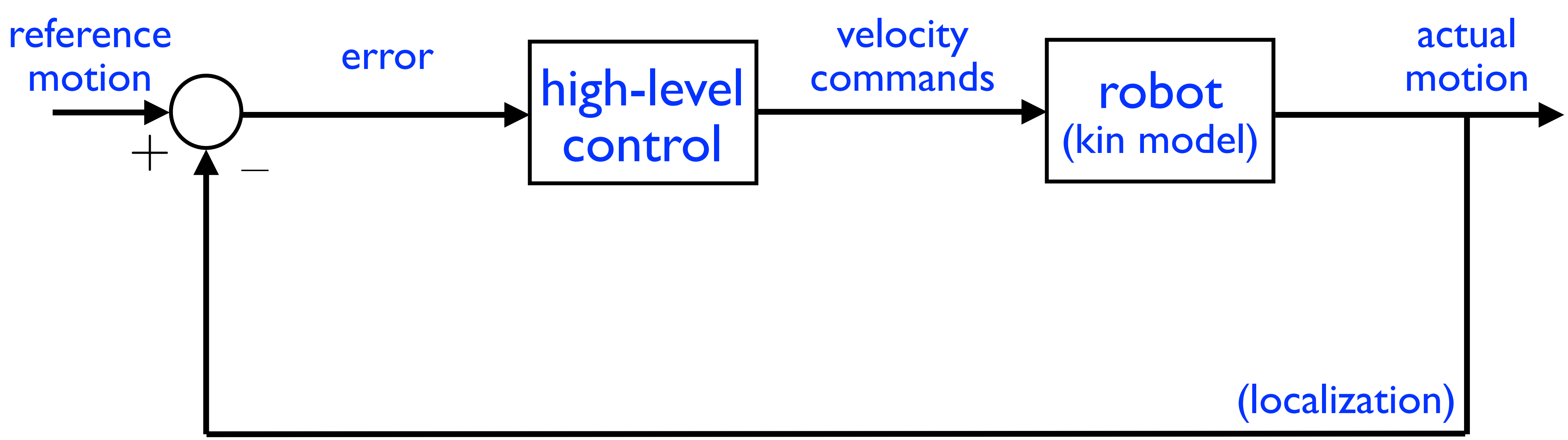
motion control

- a desired motion is assigned for the WMR, and the associated nominal inputs have been computed
- to execute the desired motion, we need feedback control because the application of nominal inputs in open-loop would lead to very poor performance
- in manipulators, we use dynamic models to compute commands at the generalized force level
- in WMRs, we use kinematic models because (1) wheels are equipped with low-level PID loops that accept velocities as reference (2) dynamics is simpler and can be mostly canceled via feedback

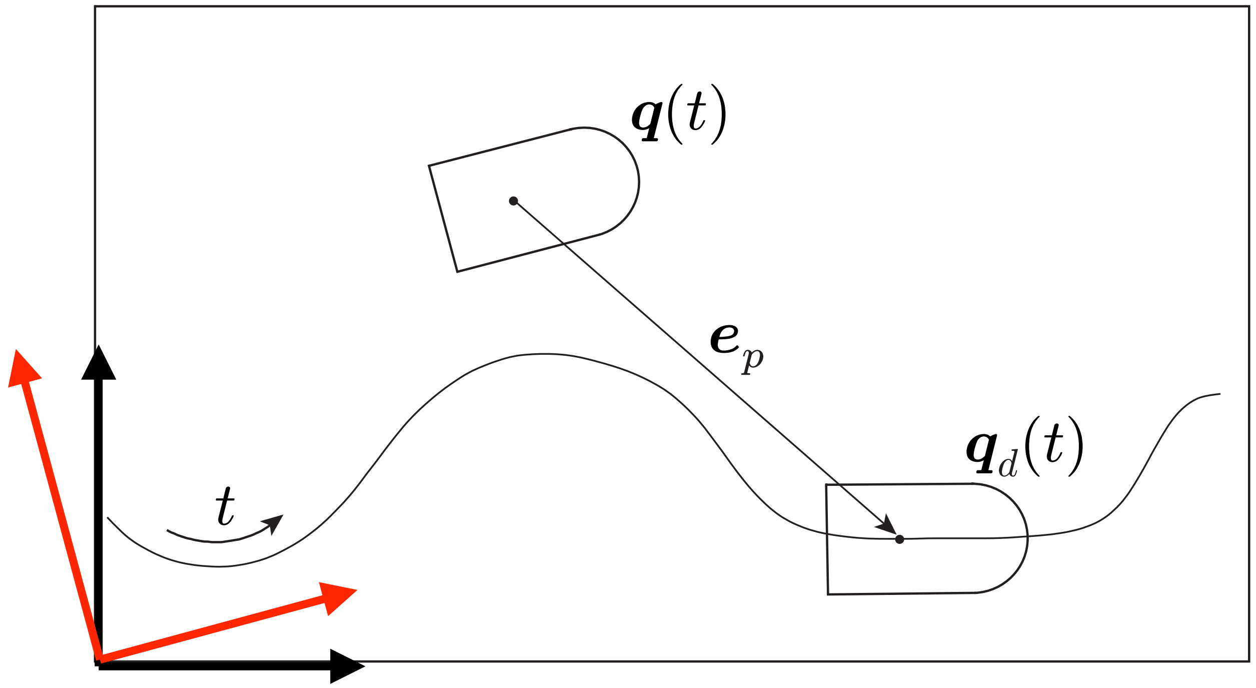
- **actual** control scheme



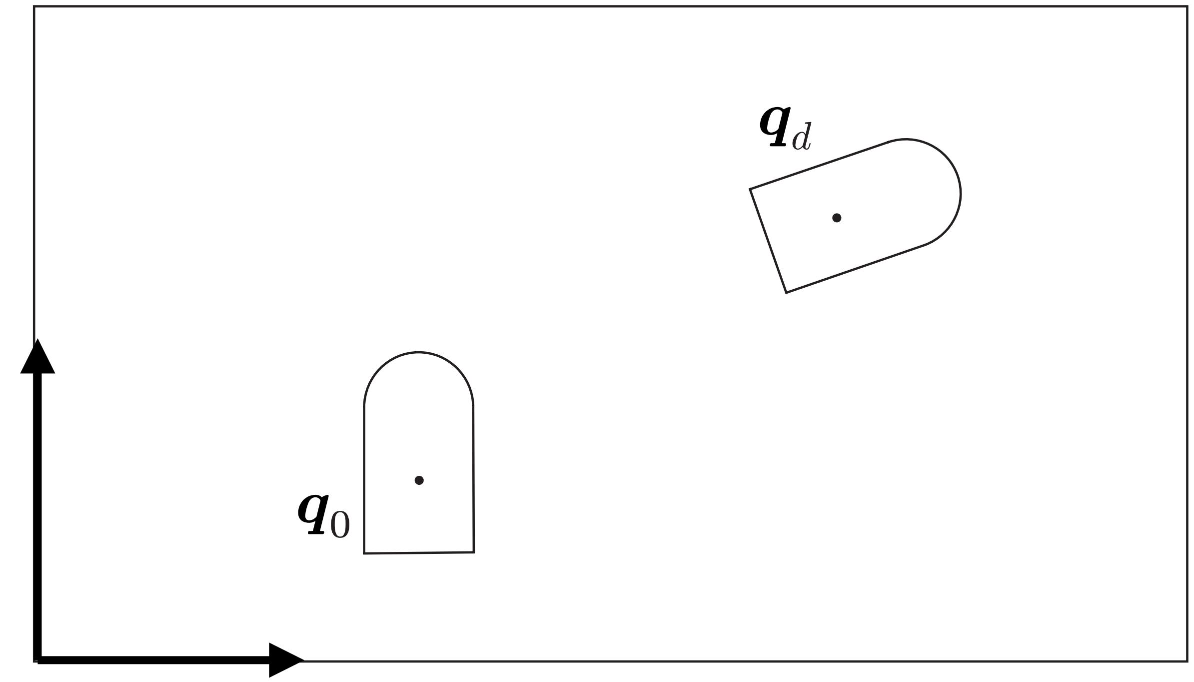
- **equivalent** control scheme (for design)



motion control problems



trajectory tracking
(predictable transients)



posture regulation
(no prior planning)

- other problems of interest
  - path tracking (only geometric motion)
  - Cartesian regulation (final orientation is free)

trajectory tracking: state error feedback

- the unicycle must track a Cartesian desired trajectory $(x_d(t), y_d(t))$ that is admissible, i.e., there exist v_d and ω_d such that

$$\dot{x}_d = v_d \cos\theta_d$$
$$\dot{y}_d = v_d \sin\theta_d$$
$$\dot{\theta}_d = \omega_d$$

- from $(x_d(t), y_d(t))$ we can compute

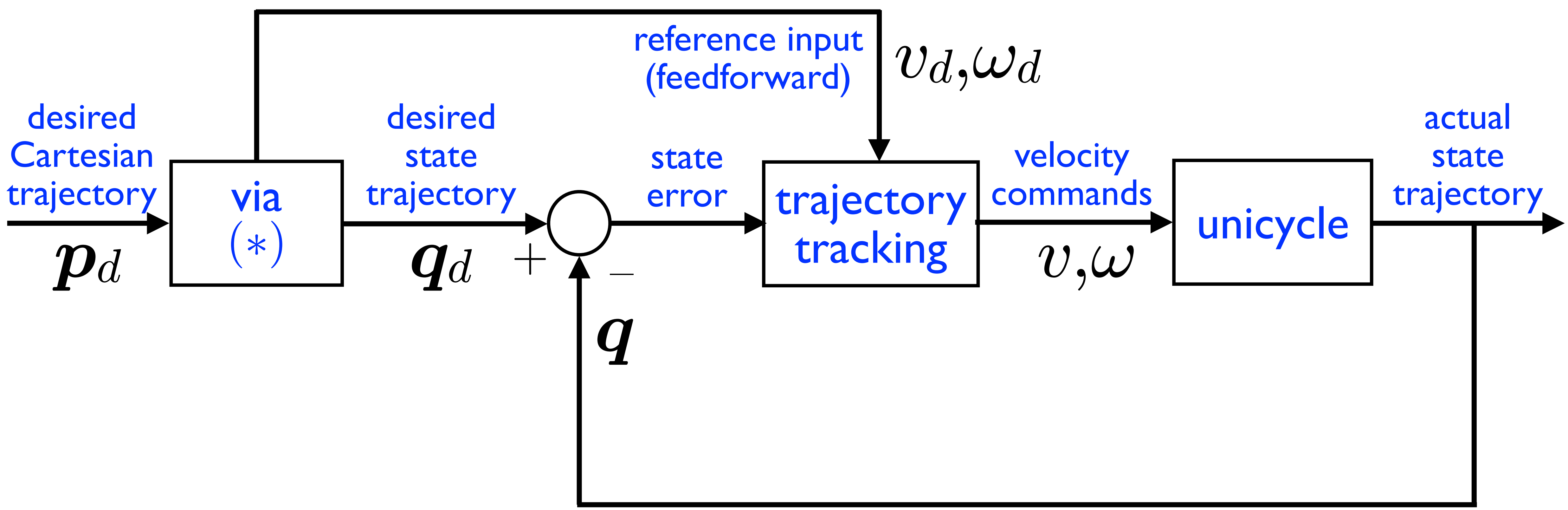
$$\theta_d(t) = \mathrm{Atan2}\left(\dot{y}_d(t), \dot{x}_d(t)\right) + k\pi \qquad k = 0, 1$$

$$v_d(t) = \pm\sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)} \qquad (*)$$

$$\omega_d(t) = \frac{\ddot{y}_d(t)\dot{x}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{\dot{x}_d^2(t) + \dot{y}_d^2(t)}$$

- the desired state trajectory can be used to compute the state error, from which the feedback action is generated; whereas the nominal input can be used as a feedforward term

- the resulting block scheme is



- rather than using directly the state error $\boldsymbol{q}_d - \boldsymbol{q}$, use its rotated version defined as

$$\boldsymbol{e} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{pmatrix}$$

(e_1, e_2) is the Cartesian error $\boldsymbol{e}_p$ in a frame rotated by θ (in red in slide 15)

- the error dynamics is nonlinear and time-varying

$$\begin{aligned} \dot{e}_1 &= v_d \cos e_3 - v + e_2\, \omega \\ \dot{e}_2 &= v_d \sin e_3 - e_1\, \omega \\ \dot{e}_3 &= \omega_d - \omega \end{aligned}$$

via approximate linearization

- a simple approach for stabilizing the error dynamics is to use its linearization around the reference trajectory (indirect Lyapunov method $\Rightarrow$ local results)
- to make the reference trajectory an unforced equilibrium for the error dynamics

$$
\begin{aligned}
\dot{e}_1 &= v_d \cos e_3 - v + e_2\, \omega \\
\dot{e}_2 &= v_d \sin e_3 - e_1\, \omega \\
\dot{e}_3 &= \omega_d - \omega
\end{aligned}
$$

use the following (invertible) input transformation

$$
\begin{aligned}
u_1 &= v_d \cos e_3 - v \\
u_2 &= \omega_d - \omega
\end{aligned}
$$

- we obtain

$$\begin{aligned}\dot{e}_1 &= \omega_d\, e_2 + u_1 - e_2\, u_2 \\ \dot{e}_2 &= -\omega_d\, e_1 + v_d \sin e_3 + e_1\, u_2 \\ \dot{e}_3 &= u_2\end{aligned}$$

that is, $\dot{\boldsymbol{e}} = \boldsymbol{\varphi}(\boldsymbol{e}, \boldsymbol{u}, t)$ with $\boldsymbol{\varphi}(\mathbf{0}, \mathbf{0}, t) = \mathbf{0}$

- note that

$$\dot{\boldsymbol{e}} = \underbrace{\begin{pmatrix} \omega_d\, e_2 \\ -\omega_d\, e_1 + v_d \sin e_3 \\ 0 \end{pmatrix}}_{\boldsymbol{f}(\boldsymbol{e}, t)} + \underbrace{\begin{pmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}}_{\boldsymbol{G}(\boldsymbol{e})\boldsymbol{u}}$$

drift term
nonlinear, time-varying

input term
nonlinear, linear in $\boldsymbol{u}$

- since

$$\boldsymbol{\varphi}(\boldsymbol{e},\boldsymbol{u},t) \approx \cancel{\boldsymbol{\varphi}(\mathbf{0},\mathbf{0},t)} + \left.\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{e}}\right|_{\substack{\boldsymbol{e}=\mathbf{0}\\ \boldsymbol{u}=\mathbf{0}}} \boldsymbol{e} + \left.\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{e}=\mathbf{0}\\ \boldsymbol{u}=\mathbf{0}}} \boldsymbol{u}$$

the linear approximation of the error dynamics is

$$\dot{\boldsymbol{e}} = \boldsymbol{\varphi}(\boldsymbol{e},\boldsymbol{u},t) \approx \left.\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{e}}\right|_{\substack{\boldsymbol{e}=\mathbf{0}\\ \boldsymbol{u}=\mathbf{0}}} \boldsymbol{e} + \left.\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{e}=\mathbf{0}\\ \boldsymbol{u}=\mathbf{0}}} \boldsymbol{u} = \boldsymbol{A}(t)\boldsymbol{e} + \boldsymbol{B}\boldsymbol{u}$$

- one easily finds

$$\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{e}} = \begin{pmatrix} 0 & \omega_d - u_2 & 0 \\ -\omega_d + u_2 & 0 & v_d \cos e_3 \\ 0 & 0 & 0 \end{pmatrix} \qquad \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{u}} = \begin{pmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{pmatrix}$$

- hence, the linearization of the error dynamics around the reference trajectory is easily computed as

$$\dot{\boldsymbol{e}} = \begin{pmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & 0 & 0 \end{pmatrix} \boldsymbol{e} + \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

- define the linear feedback

$$\boldsymbol{u} = \boldsymbol{K}\boldsymbol{e} = \begin{pmatrix} -k_1 & 0 & 0 \\ 0 & -k_2 & -k_3 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix}$$

- the closed-loop error dynamics is still time-varying!

$$\dot{\boldsymbol{e}} = \boldsymbol{A}(t)\,\boldsymbol{e} = \begin{pmatrix} -k_1 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & -k_2 & -k_3 \end{pmatrix} \boldsymbol{e}$$

- letting

$$k_1 = k_3 = 2\zeta a \qquad k_2 = \frac{a^2 - \omega_d^2}{v_d}$$

with $a > 0$, $\zeta \in (0,1)$, the characteristic polynomial of $\boldsymbol{A}(t)$ becomes time-invariant and Hurwitz

$$p(\lambda) = \underbrace{(\lambda + 2\zeta a)}_{\text{real negative eigenvalue}} \underbrace{(\lambda^2 + 2\zeta a \lambda + a^2)}_{\text{pair of complex eigenvalues with negative real part}}$$

- caveat: this does not guarantee asymptotic stability, unless v_d and ω_d are constant (rectilinear and circular trajectories); even in this case, asymptotic stability of the unicycle is not global (indirect Lyapunov method)

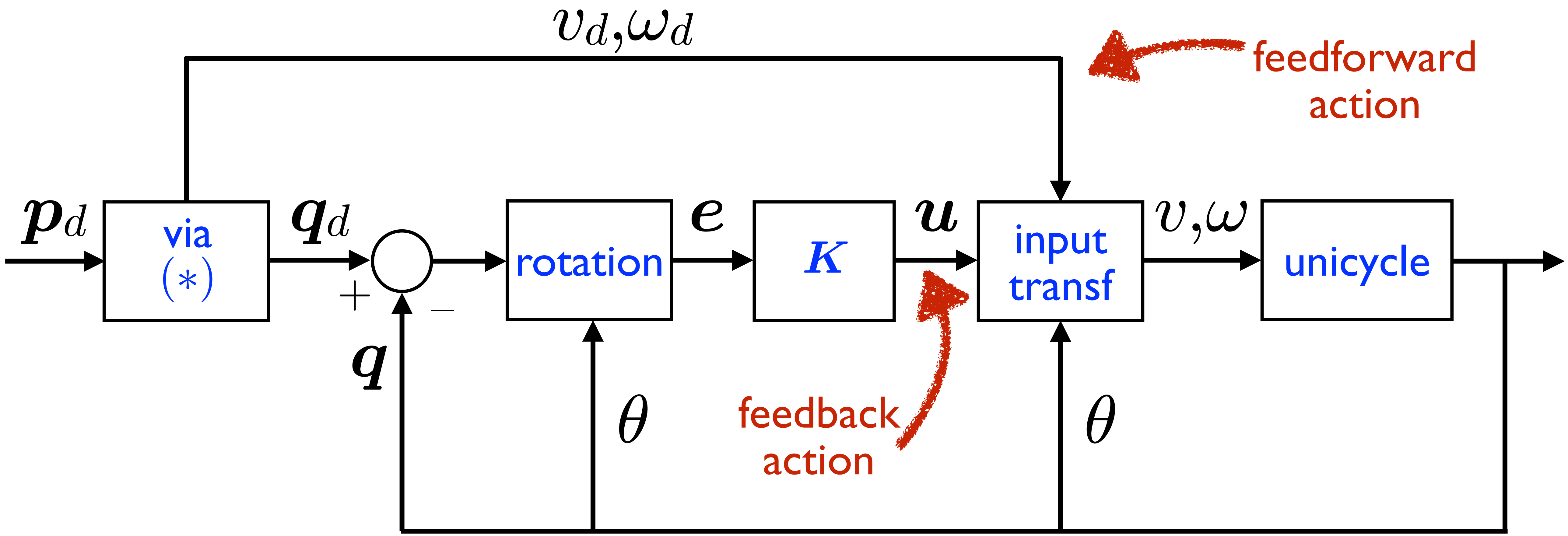
- the actual velocity inputs v,ω are obtained plugging the feedbacks u_1, u_2 in the input transformation
- note: $(v,\omega) \to (v_d,\omega_d)$ as $\boldsymbol{e} \to \mathbf{0}$ (pure feedforward)
- note: $k_2 \to \infty$ as $v_d \to 0$, hence this controller can only be used with persistent Cartesian trajectories (stops are not allowed)
- global stability is guaranteed by a nonlinear version

$$u_1 = -k_1(v_d, \omega_d)\, e_1$$

$$u_2 = -k_2\, v_d\, \frac{\sin e_3}{e_3}\, e_2 - k_3(v_d, \omega_d)\, e_3$$

if k_1,k_3 bounded, positive, with bounded derivatives

- the final block scheme for trajectory tracking via state error feedback and approximate linearization is

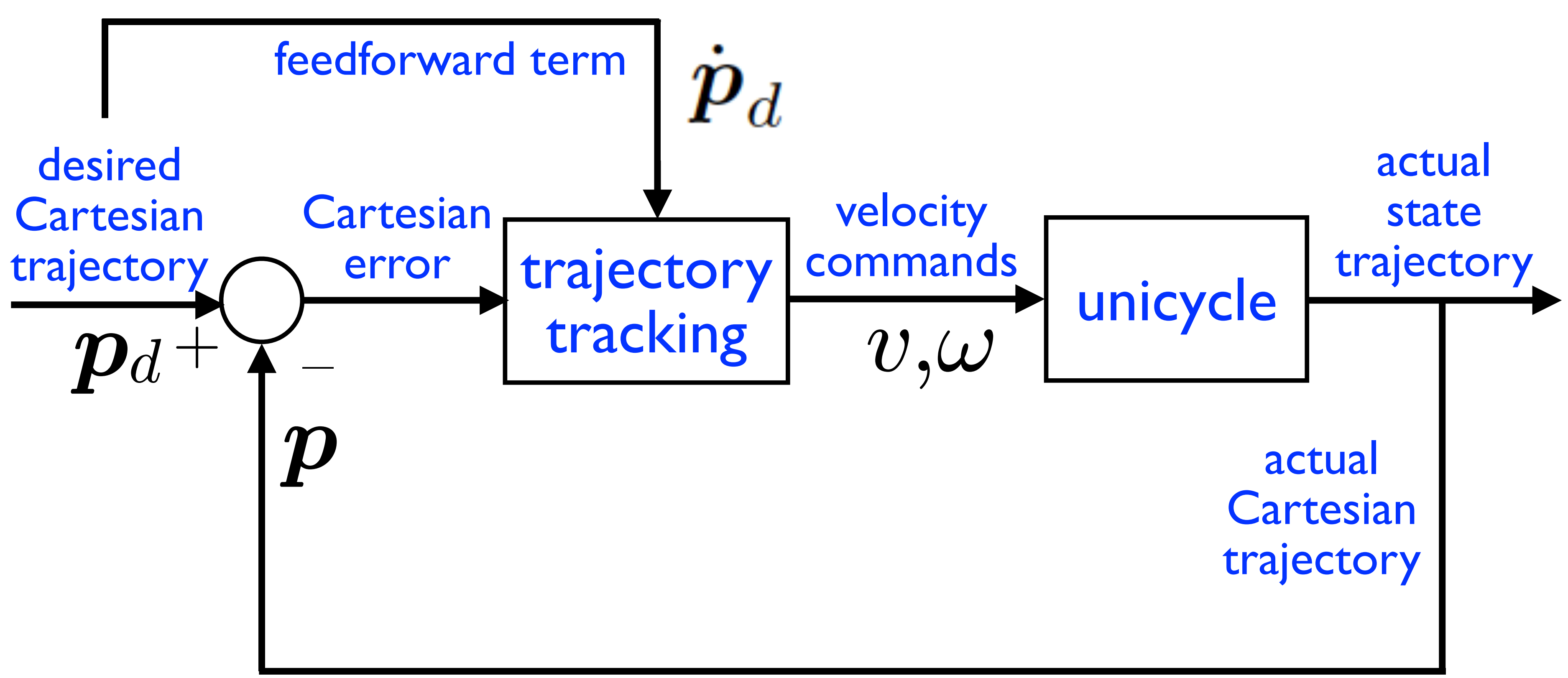


- a static controller based on state error
- needs v_d, ω_d
- needs θ also for error rotation + input transformation

trajectory tracking: output error feedback

- another approach: develop the feedback action from the output (Cartesian) error only, without computing a desired state trajectory, while the feedforward term is the Cartesian velocity along the reference trajectory

- the resulting block scheme is



exact i/o linearization: brush-up

- consider a driftless nonlinear system

$$\dot{\boldsymbol{x}} = \boldsymbol{G}(\boldsymbol{x})\boldsymbol{u} \qquad \boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{u} \in \mathbb{R}^m,\ \boldsymbol{y} \in \mathbb{R}^m$$
$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x})$$

- being

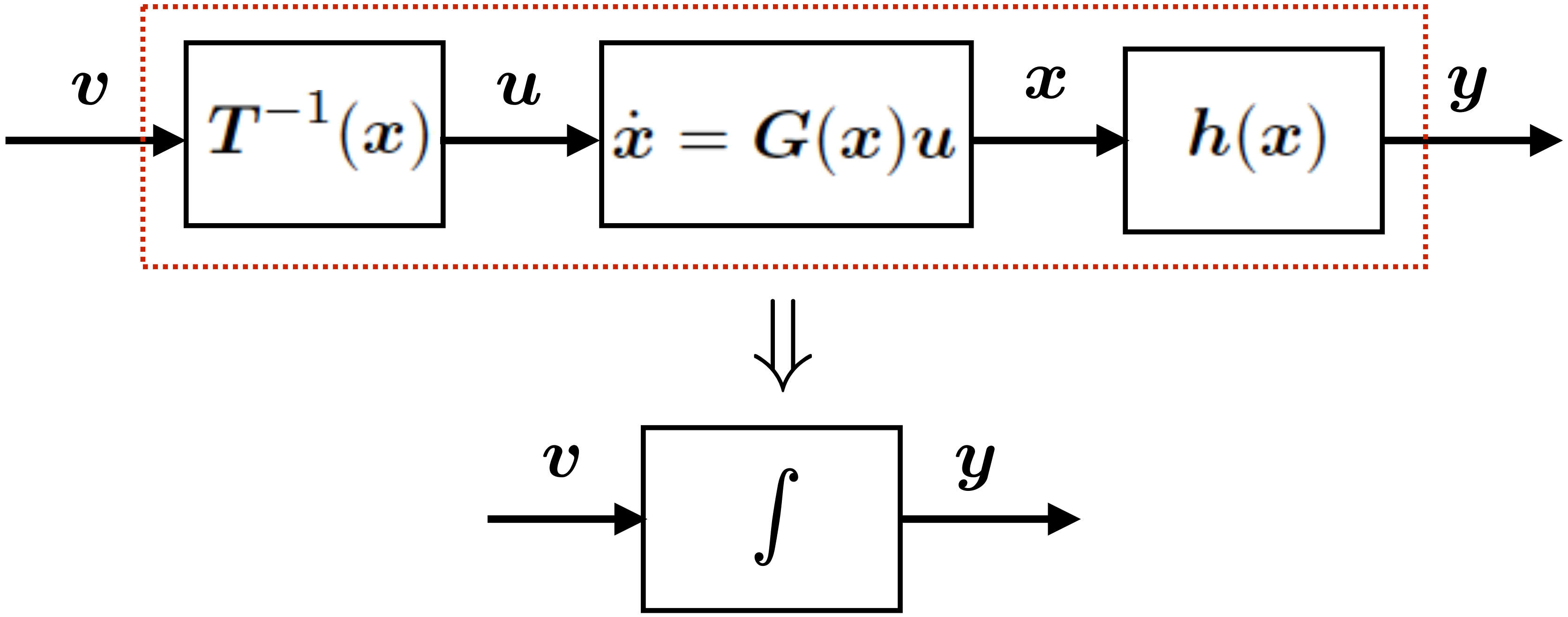
$$\dot{\boldsymbol{y}} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\boldsymbol{G}(\boldsymbol{x})\boldsymbol{u} = \boldsymbol{T}(\boldsymbol{x})\boldsymbol{u}$$

if the $m \times m$ decoupling matrix $\boldsymbol{T}$ is invertible we set

$\boldsymbol{u} = \boldsymbol{T}^{-1}(\boldsymbol{x})\boldsymbol{v}$ obtaining $\dot{\boldsymbol{y}} = \boldsymbol{T}(\boldsymbol{x})\boldsymbol{T}^{-1}(\boldsymbol{x})\boldsymbol{v} = \boldsymbol{v}$

i.e., an exactly linear map between the inputs and the (time derivative of) the outputs

- in pictures



under the action
of the linearizing feedback
$u = T^{-1}(x)v\ldots$

the system behaves as

…a simple (vector) integrator
from v to y

- given a reference output $y_d(t)$, the dynamics of the output error $e = y_d - y$ is $\dot{e} = \dot{y}_d - \dot{y} = \dot{y}_d - v$
- let $v = \dot{y}_d + Ke$ (feedforward+proportional feedback) to obtain $\dot{e} = -Ke$, i.e., global exponential stability provided that the eigenvalues of K are in the rhp
- the final control law is $u = T^{-1}(x)(\dot{y}_d + Ke)$

via exact input/output linearization

- let us adopt the exact i/o linearization approach to design a Cartesian trajectory tracking controller for the unicycle

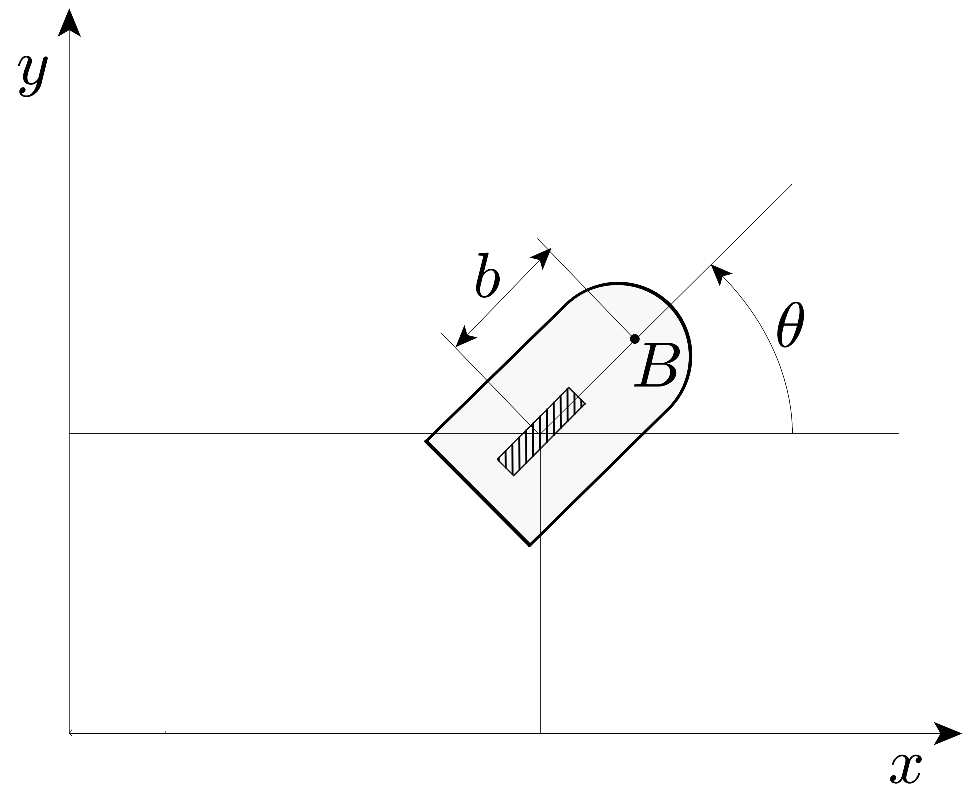
- however, for the unicycle the map between the velocity inputs and the Cartesian output is singular

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}$$

as a consequence, input-output linearization is not possible in this case

- solution: change slightly the output so that the new input-output map is invertible and exact linearization becomes possible

- displace the output from the contact point of the wheel to point B along the sagittal axis



$$y_1 = x_B = x + b\cos\theta$$
$$y_2 = y_B = y + b\sin\theta$$

- differentiating wrt time

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\theta & -b\sin\theta \\ \sin\theta & b\cos\theta \end{pmatrix}}_{\text{det} = b} \begin{pmatrix} v \\ \omega \end{pmatrix} = \boldsymbol{T}(\theta) \begin{pmatrix} v \\ \omega \end{pmatrix}$$

- if $b \neq 0$, we may set

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \boldsymbol{T}^{-1}(\theta) \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta/b & \cos\theta/b \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

obtaining an input-output linearized system

$$\begin{aligned} \dot{y}_1 &= u_1 \\ \dot{y}_2 &= u_2 \\ \dot{\theta} &= \frac{u_2\cos\theta - u_1\sin\theta}{b} \end{aligned}$$

- achieve global exponential convergence of x_B, y_B to the desired trajectory by letting

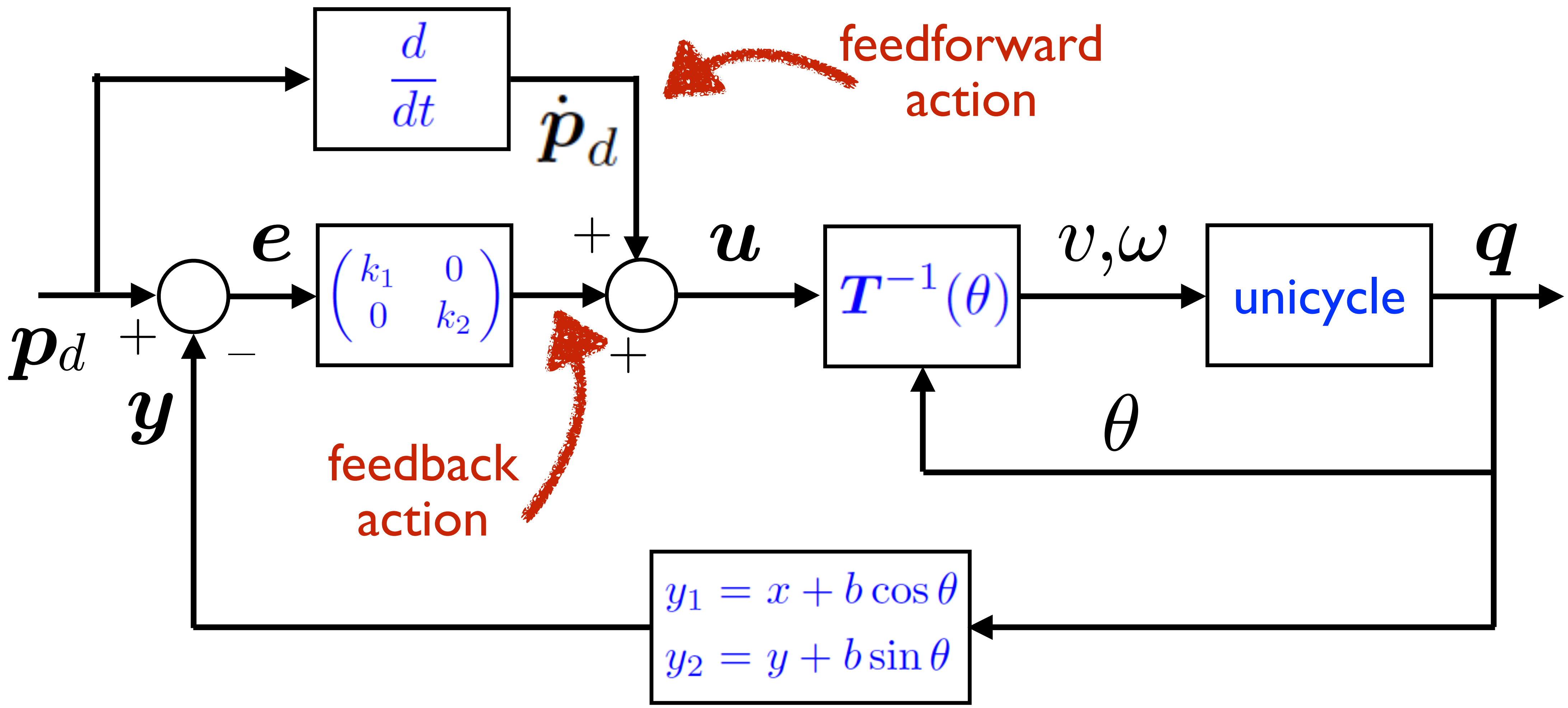
$$u_1 = \dot{x}_d + k_1(x_d - x_B)$$
$$u_2 = \dot{y}_d + k_2(y_d - y_B)$$

with $k_1, k_2 > 0$

- θ is not controlled with this scheme, which is based on output error feedback

- the desired trajectory for B can be arbitrary; in particular, square corners may be included

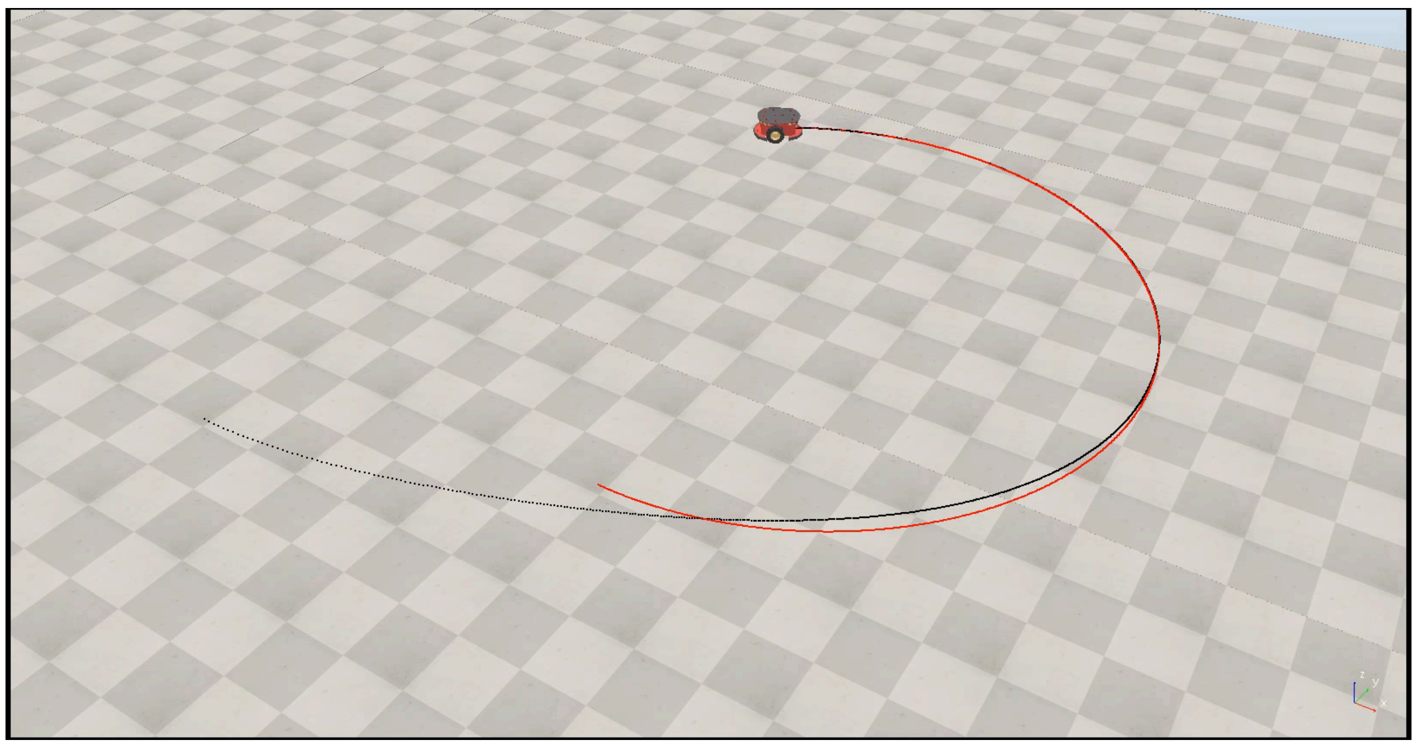
- the final block scheme for trajectory tracking via output error feedback + input-output linearization is



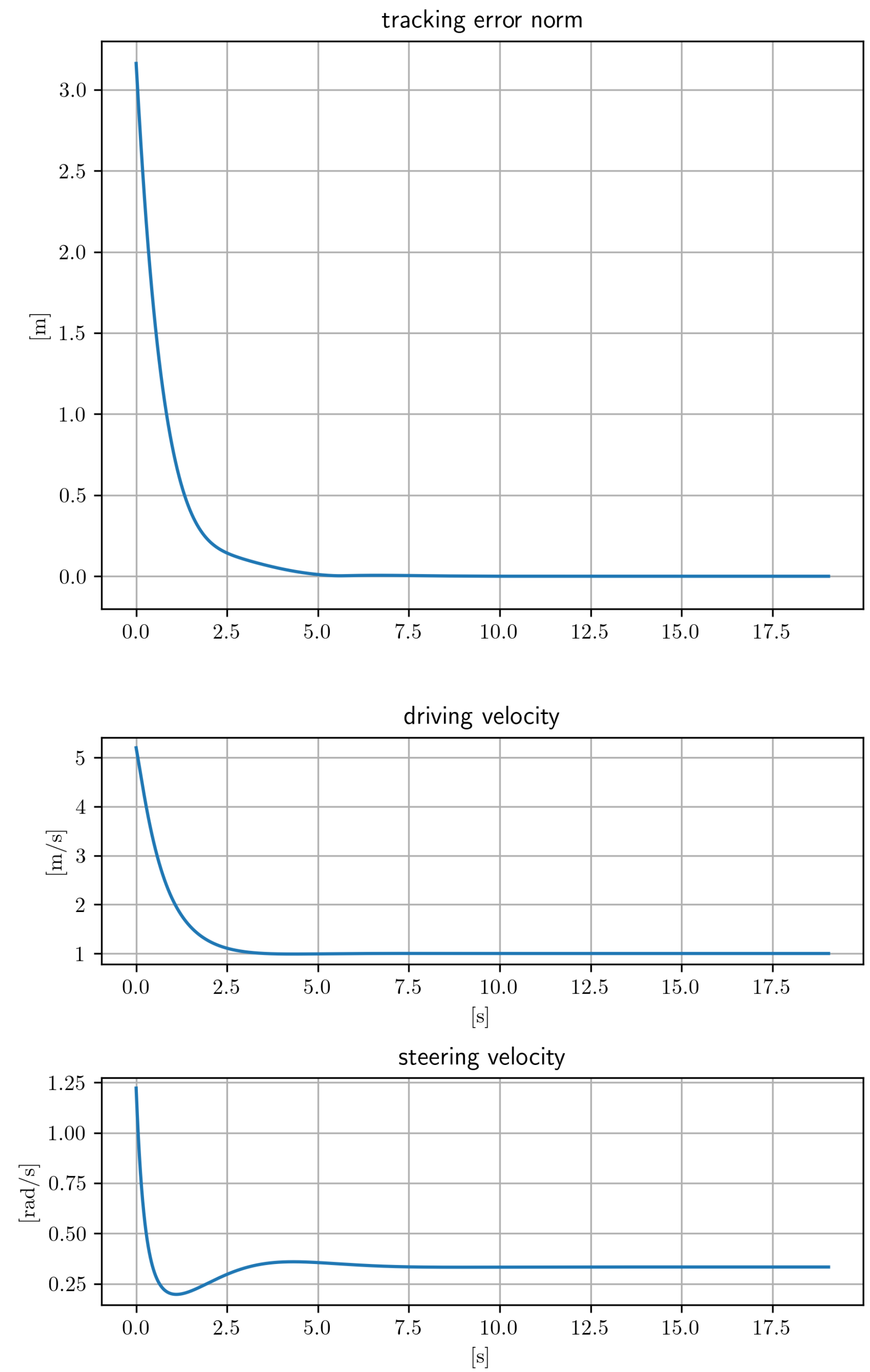
- based on output error
- needs $\dot{\boldsymbol{p}}_d$
- needs x,y,θ for output reconstruction and θ also for input transformation

simulations

tracking a circle via approximate linearization

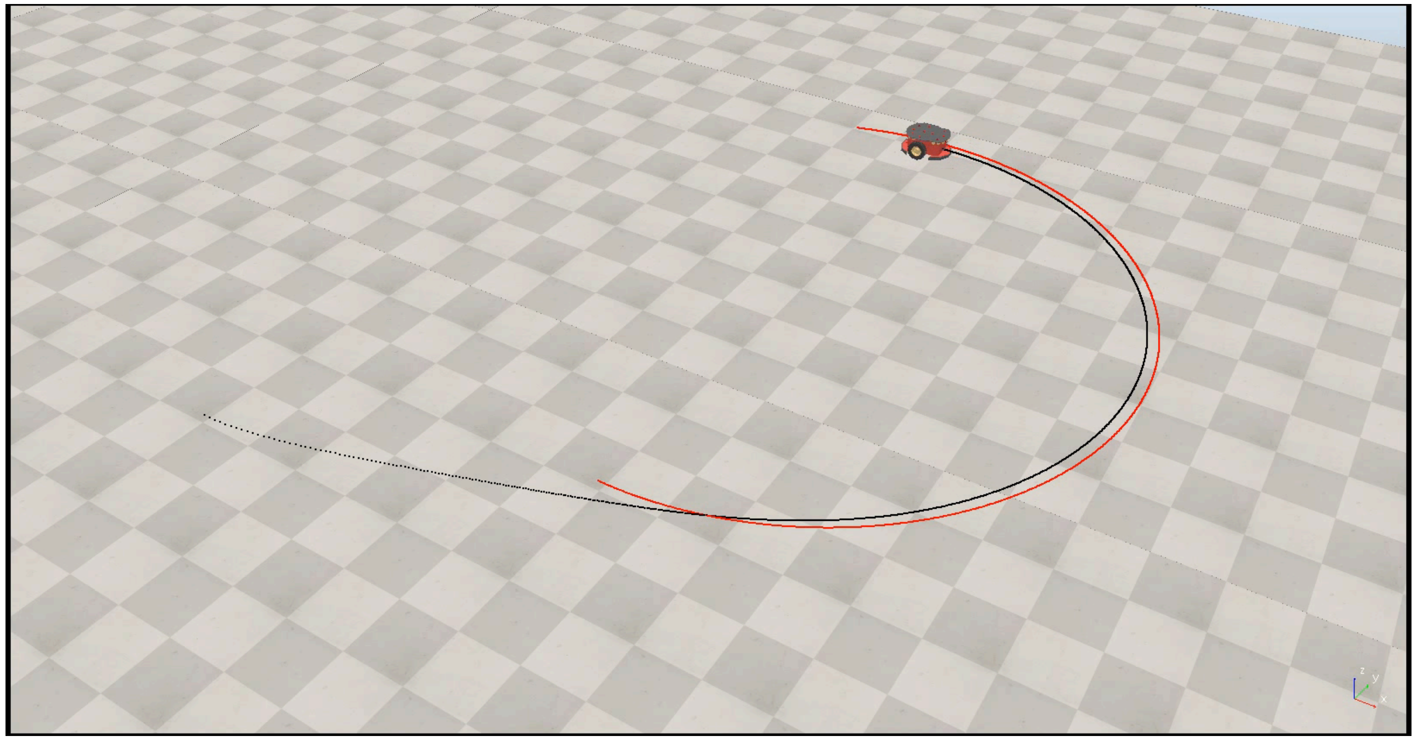


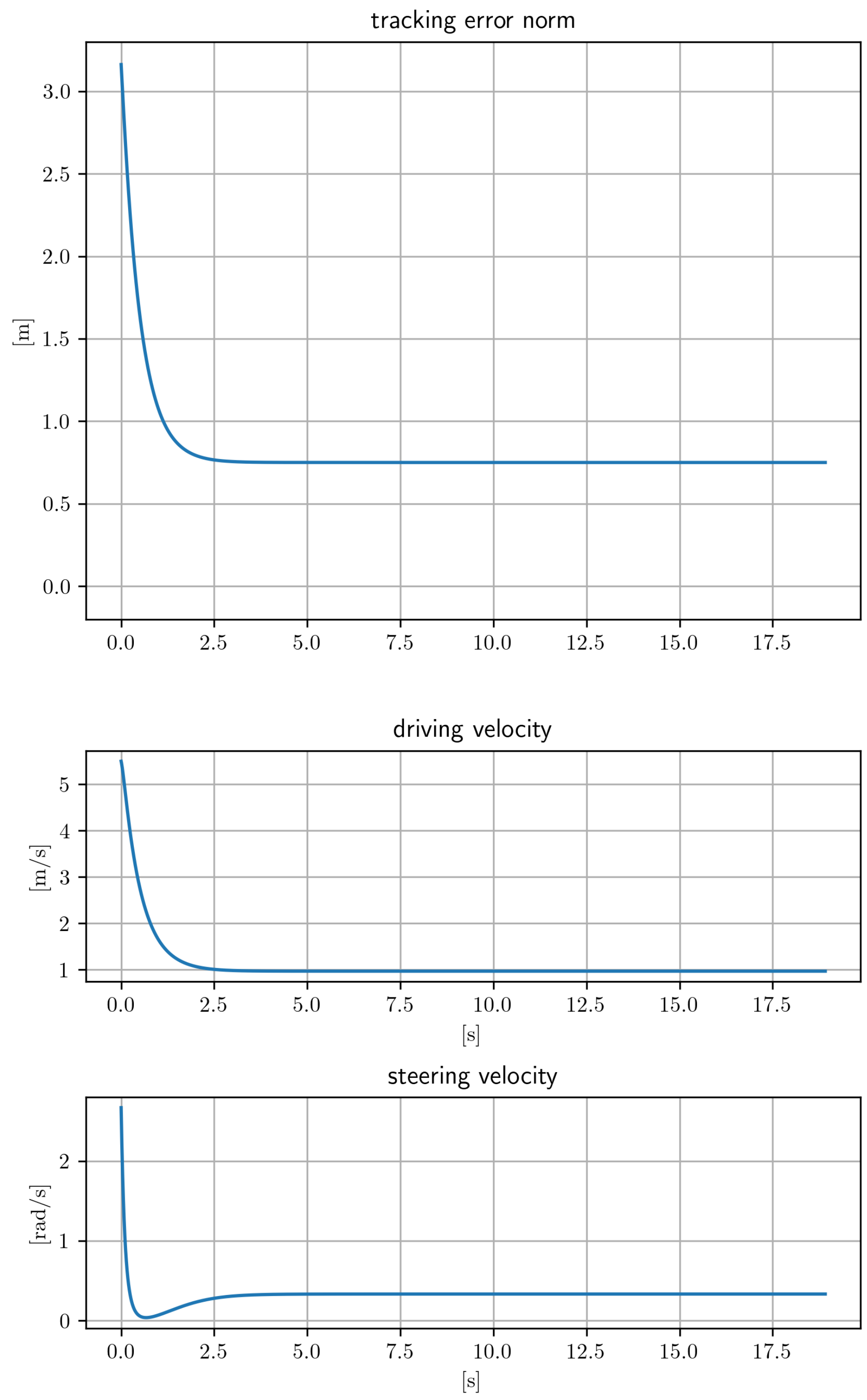
- only local stability is guaranteed



simulations

tracking a circle via exact i/o linearization (b=0.75)

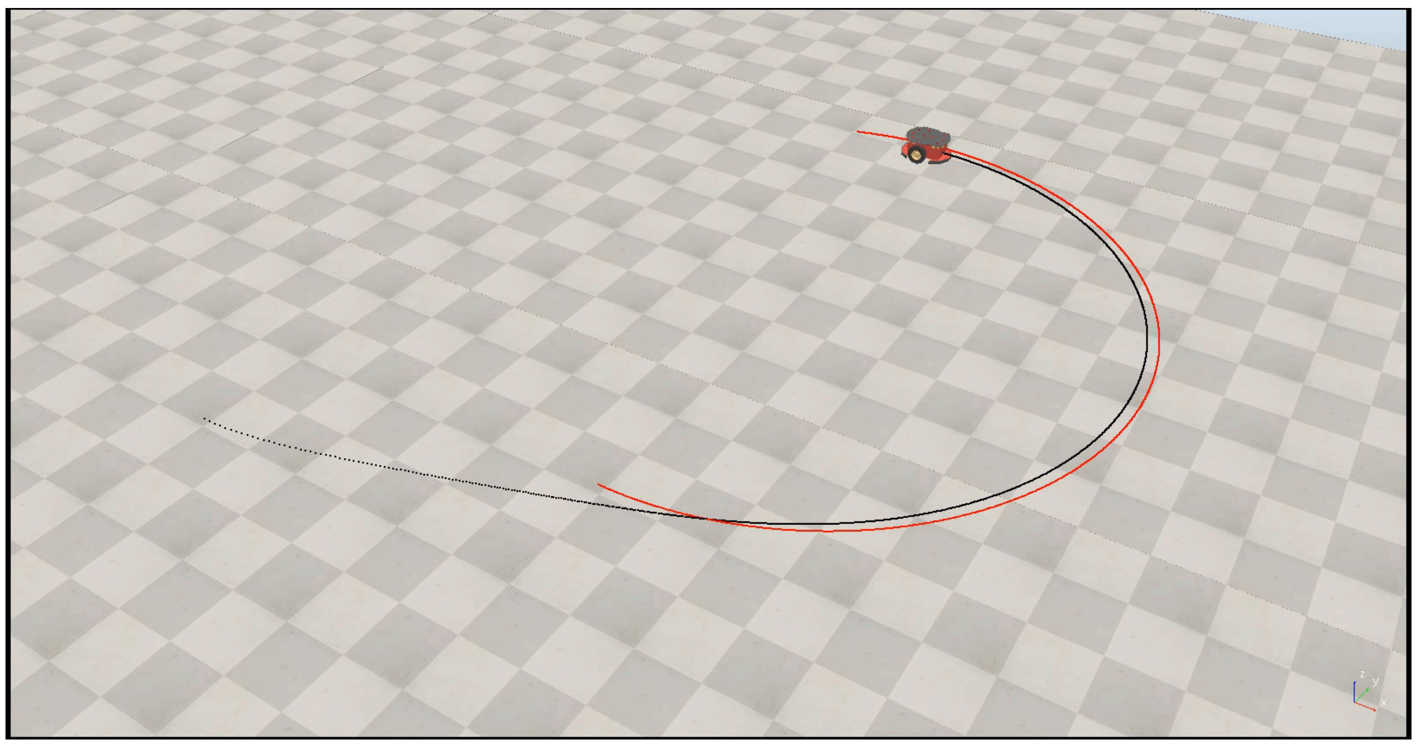


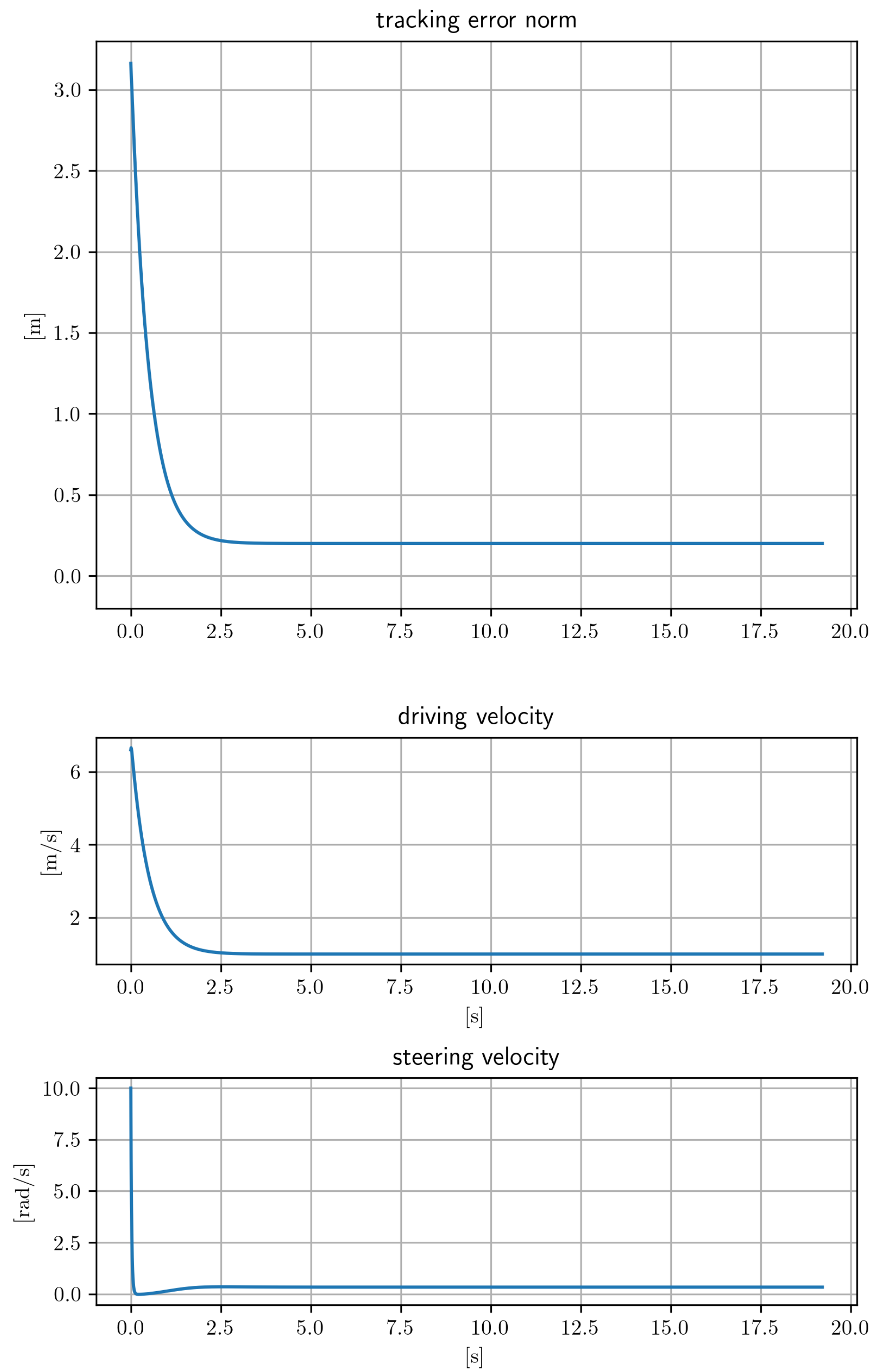


- steady-state error $= b$

simulations

tracking a circle via exact i/o linearization (b=0.2)

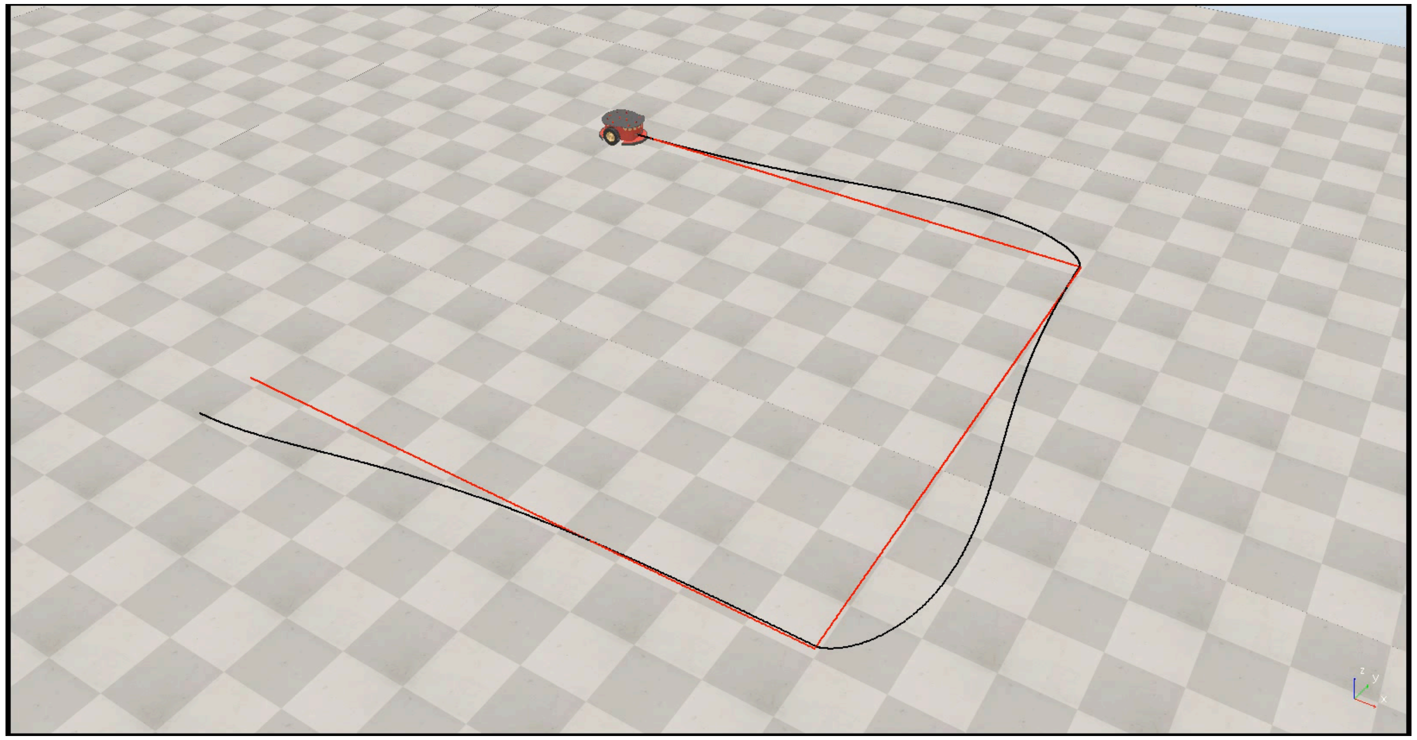


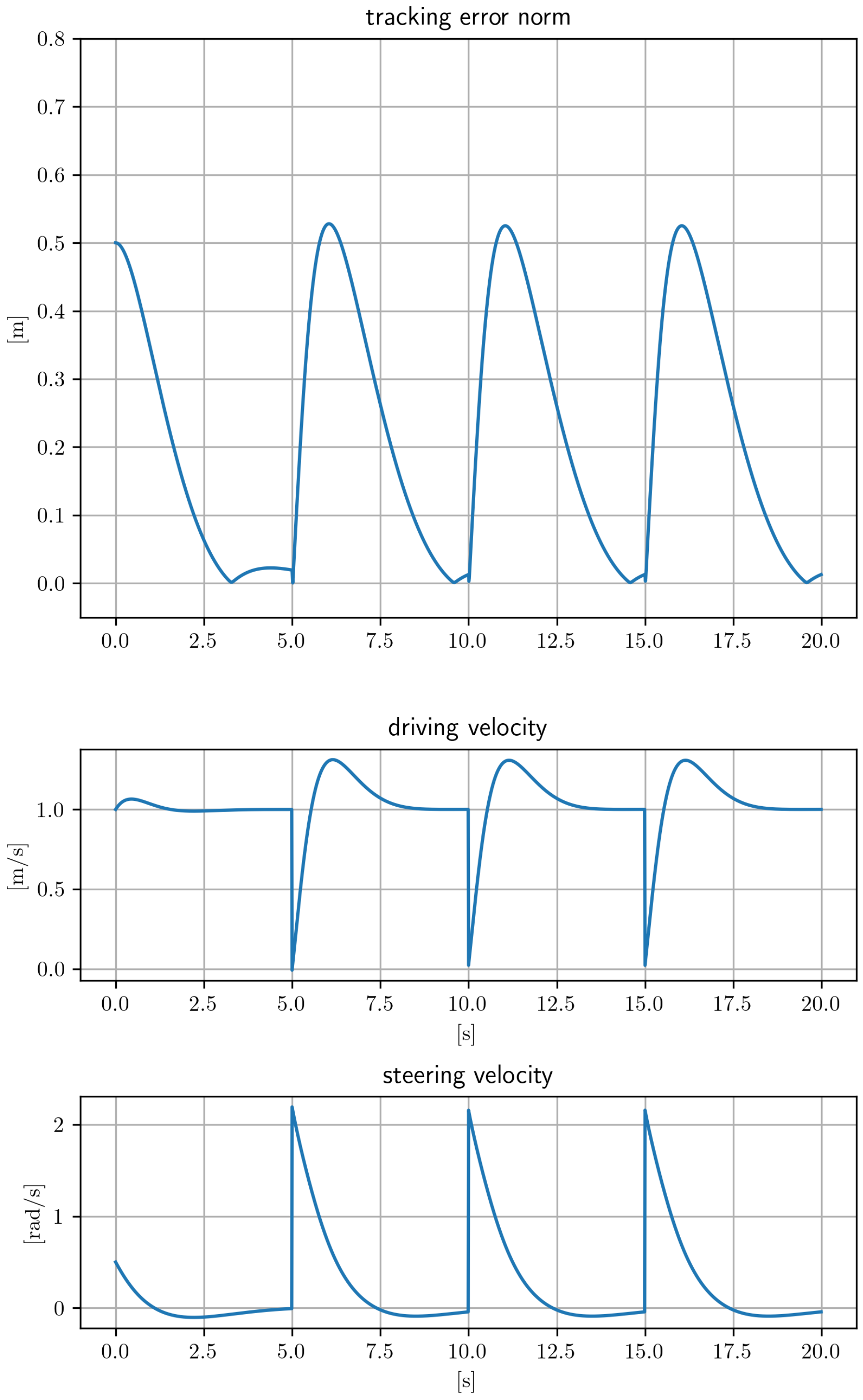


- steady-state error is now reduced but steering velocity increases

simulations

tracking a square via approximate linearization

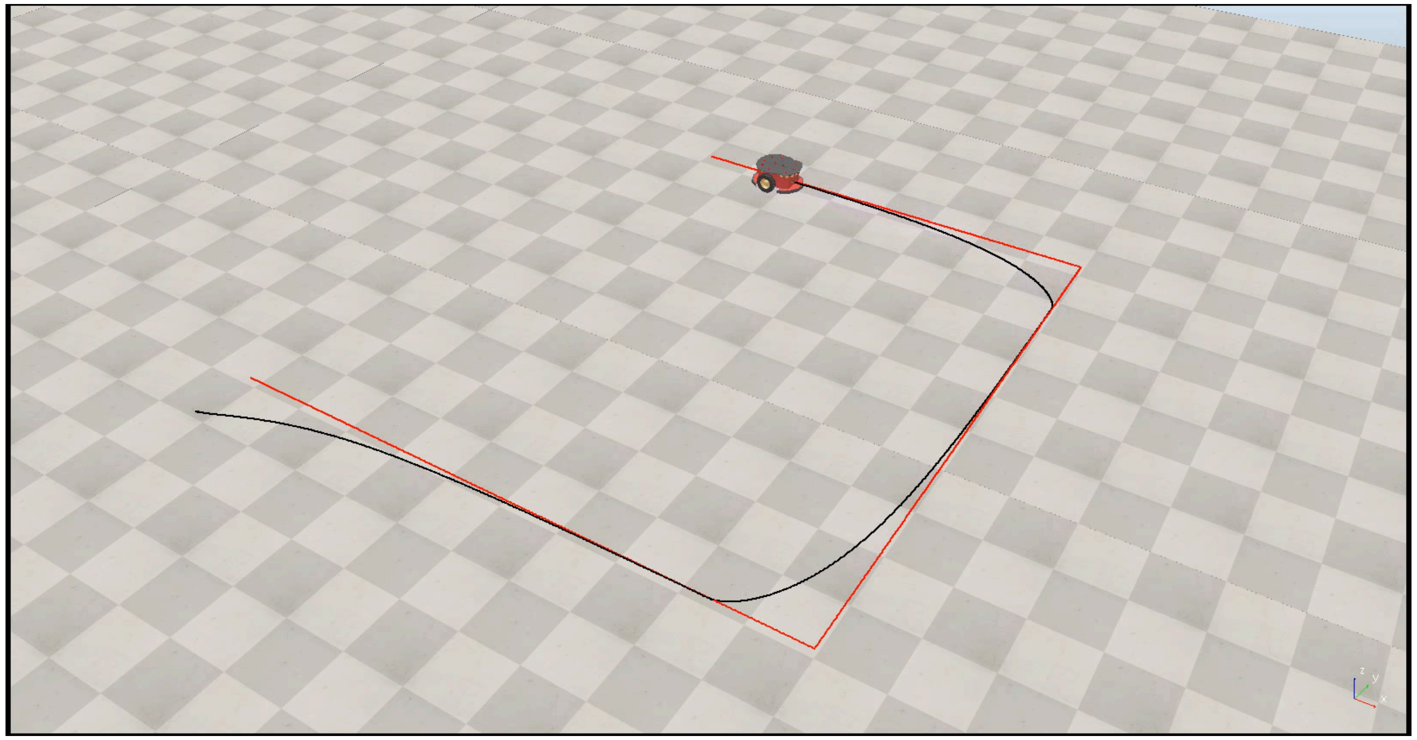


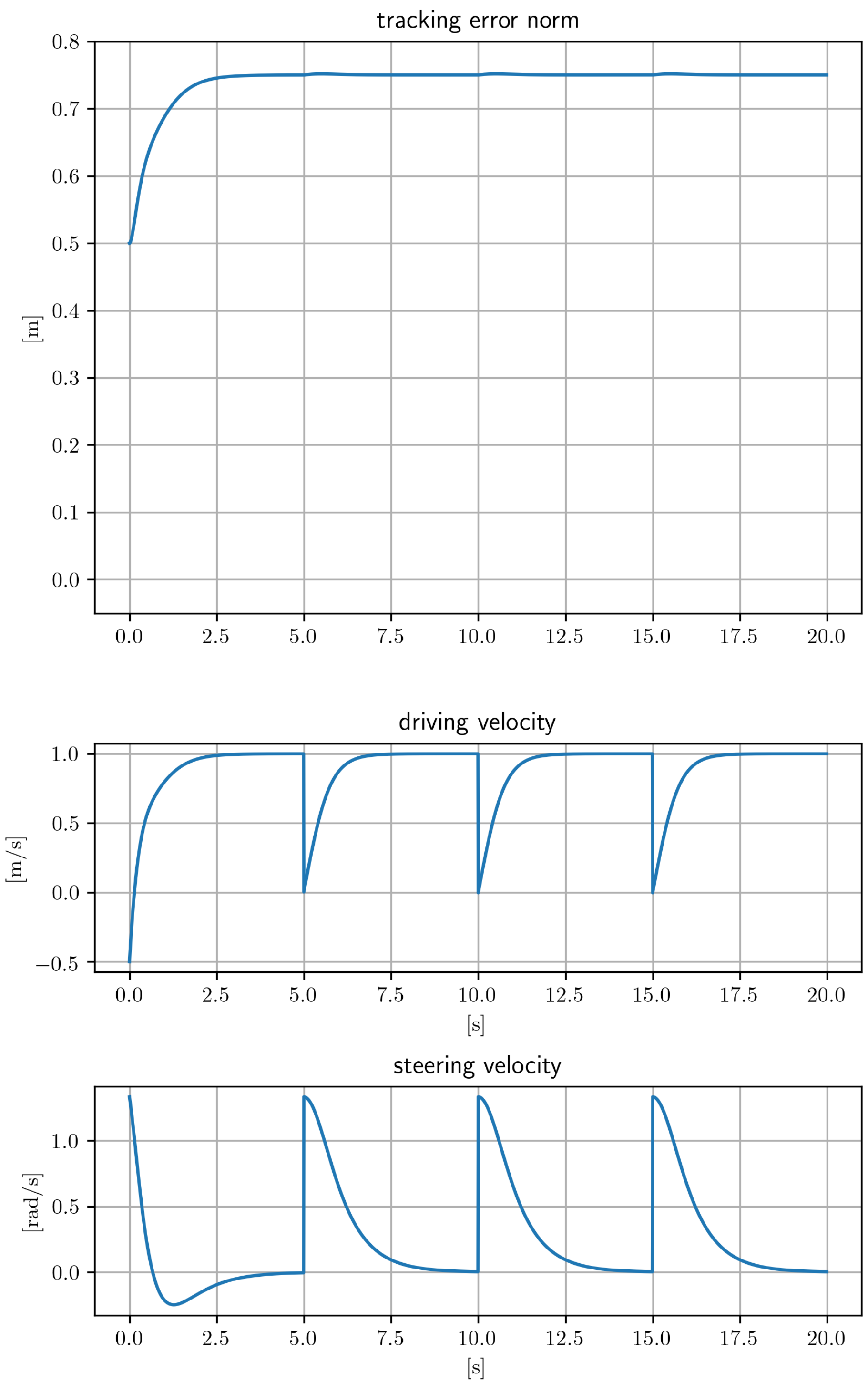


- only local stability is guaranteed
- a new transient at each corner

simulations

tracking a square via exact i/o linearization (b=0.75)

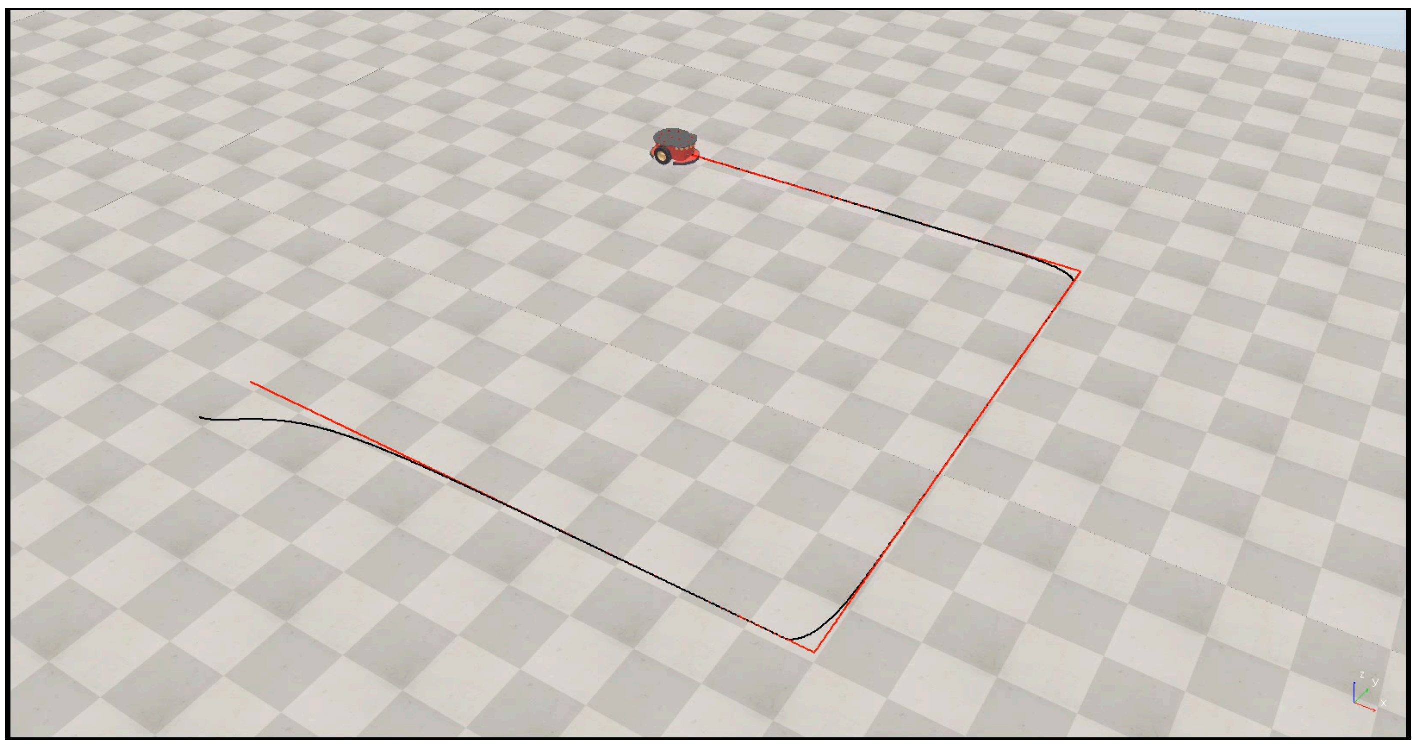


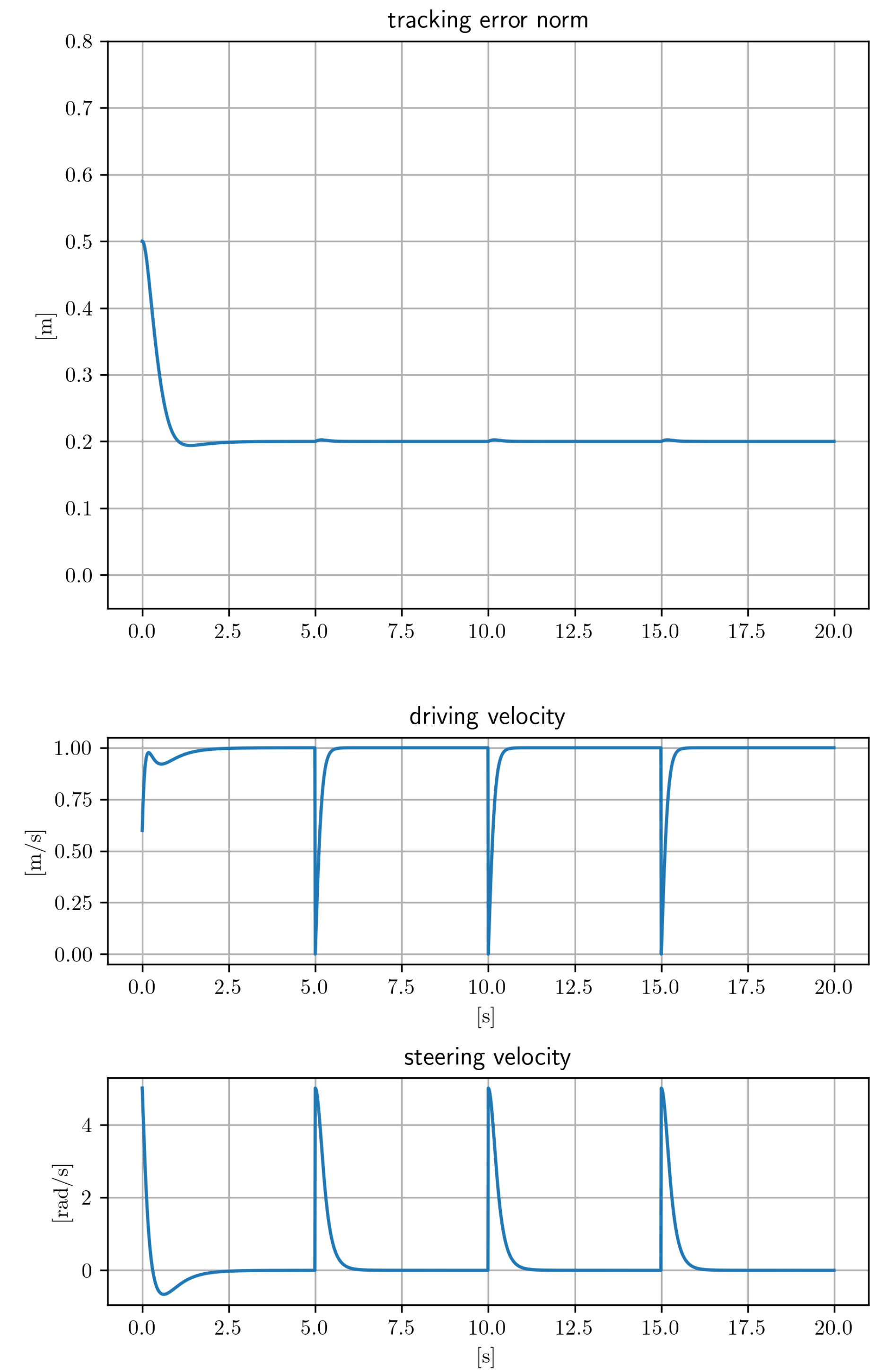


- steady-state error $= b$
- the displaced output provides a lookahead behavior

simulations

tracking a square via exact i/o linearization (b=0.2)

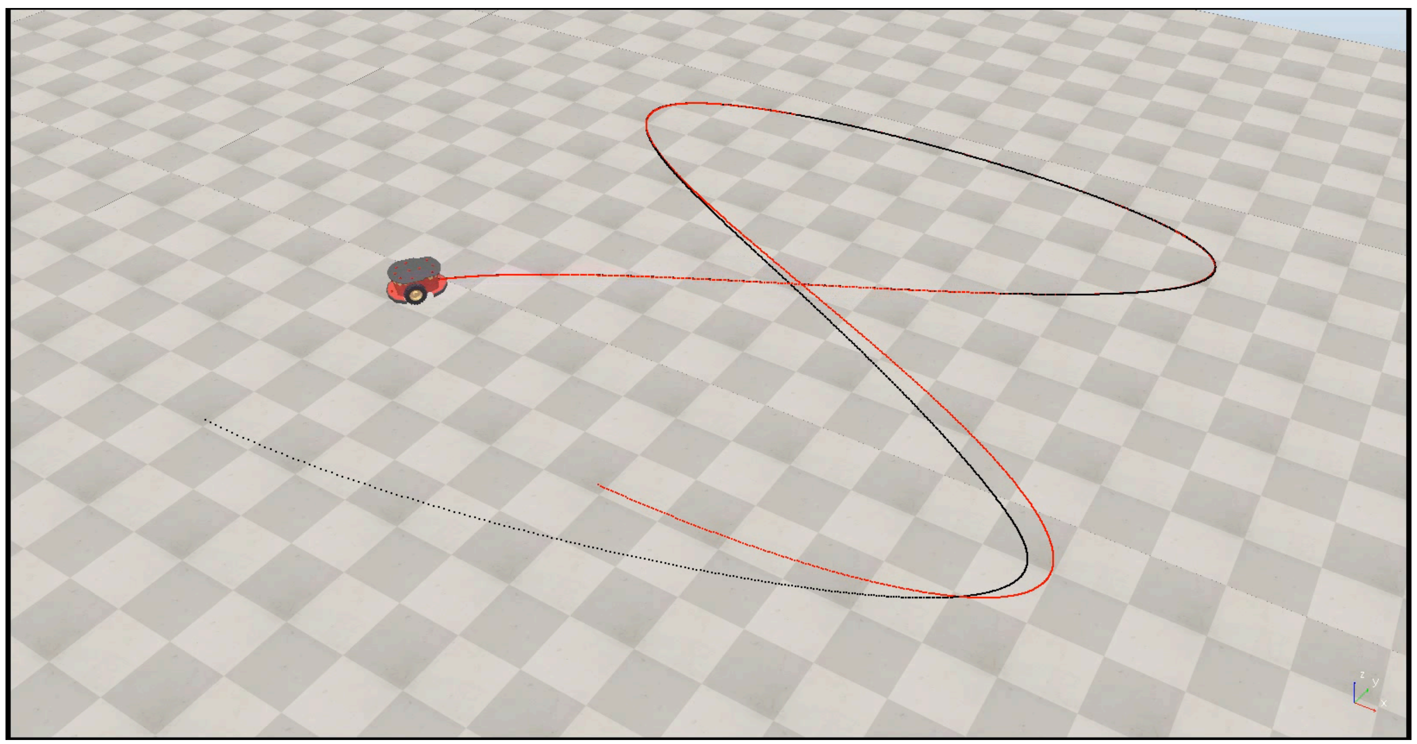


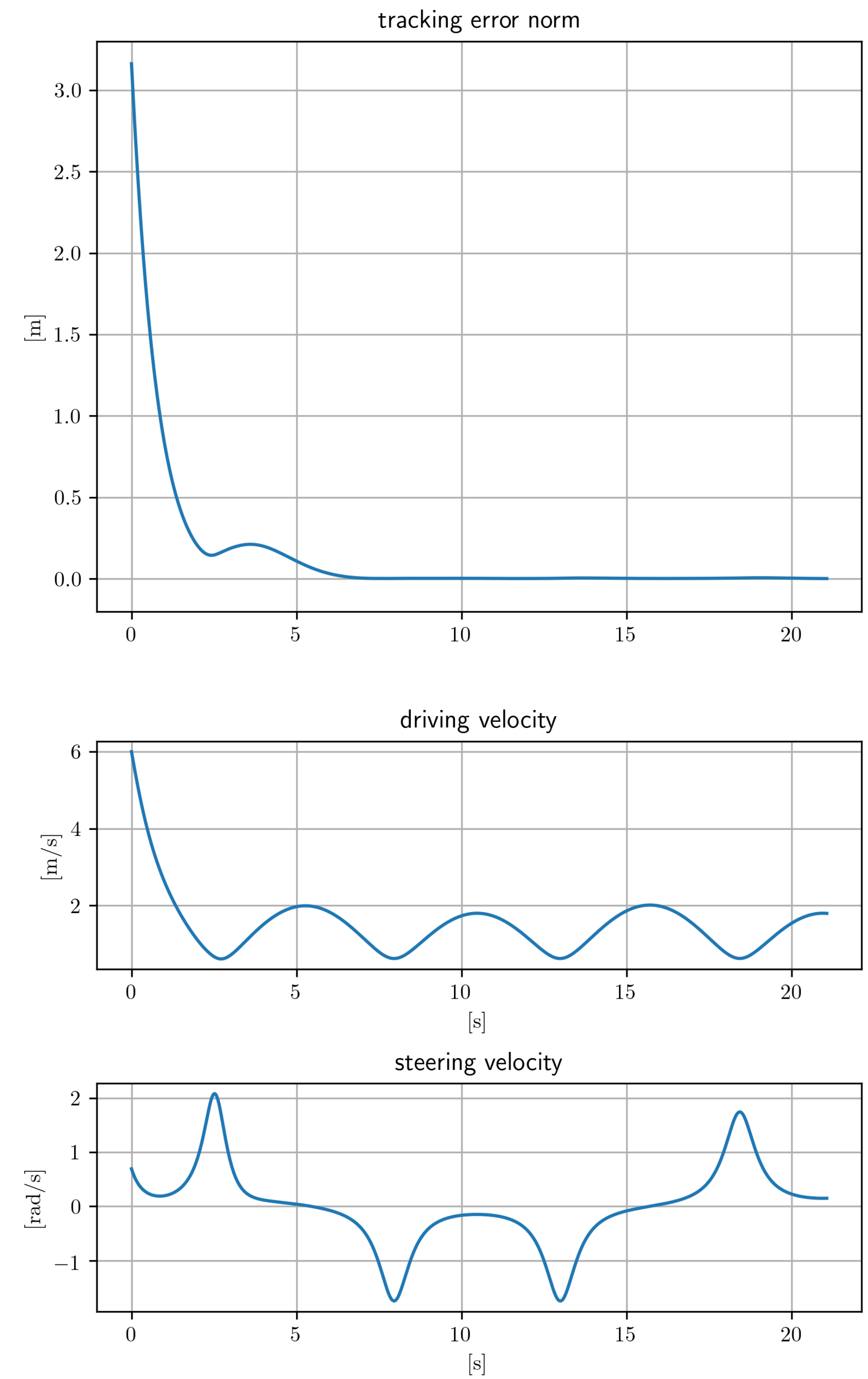


- steady-state error is now reduced but steering velocity increases

simulations

tracking a figure 8 via approximate linearization

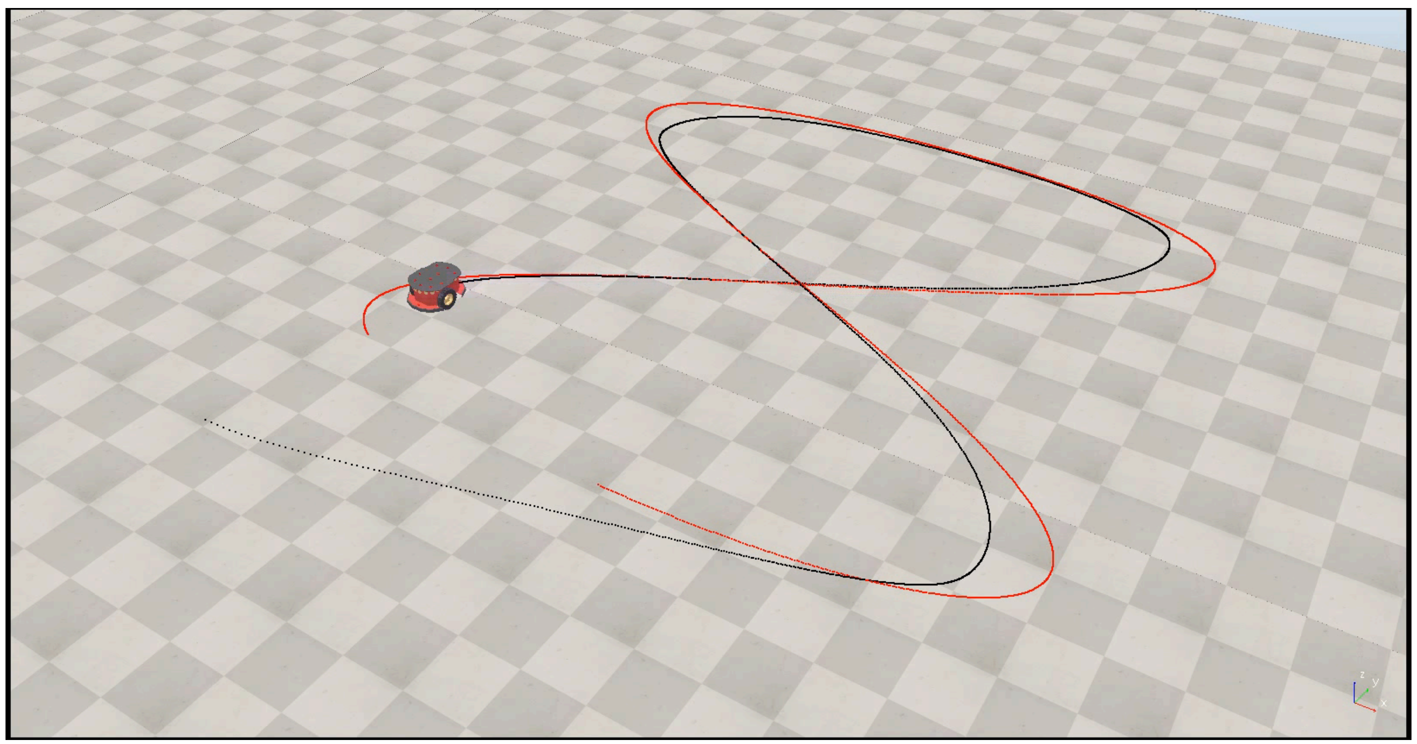


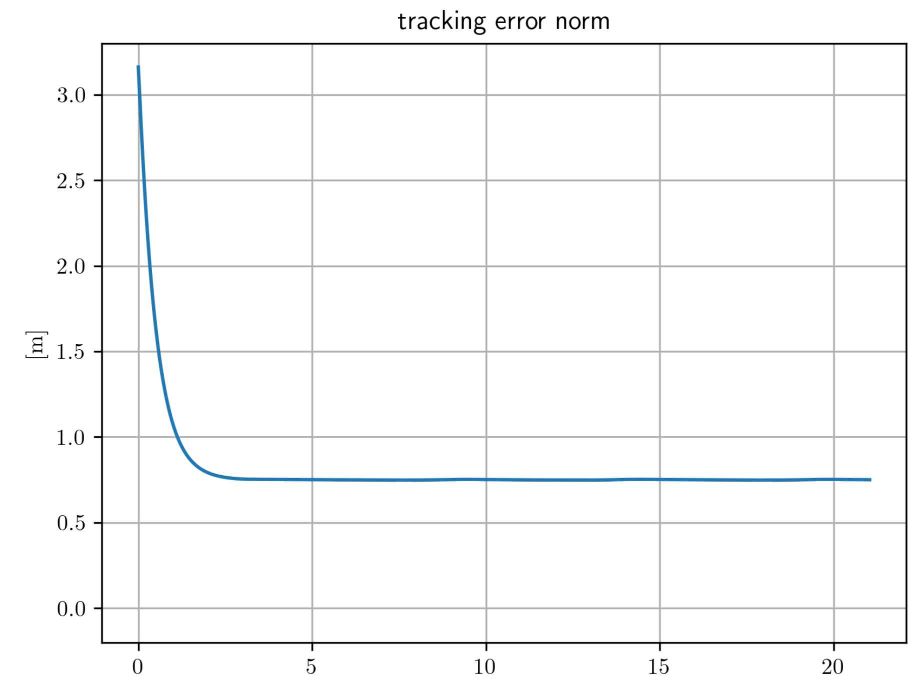


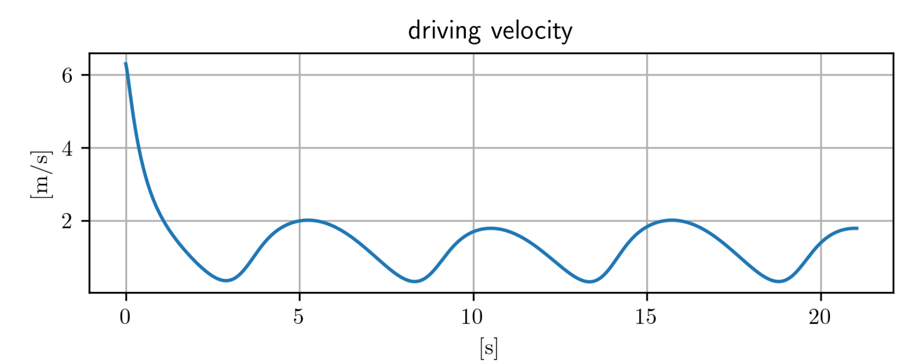
- local stability is not guaranteed, but performance is good

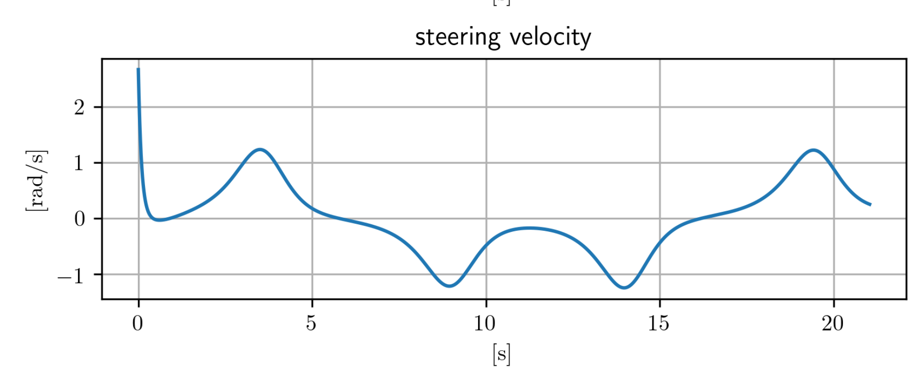
simulations

tracking a figure 8 via exact i/o linearization (b=0.75)





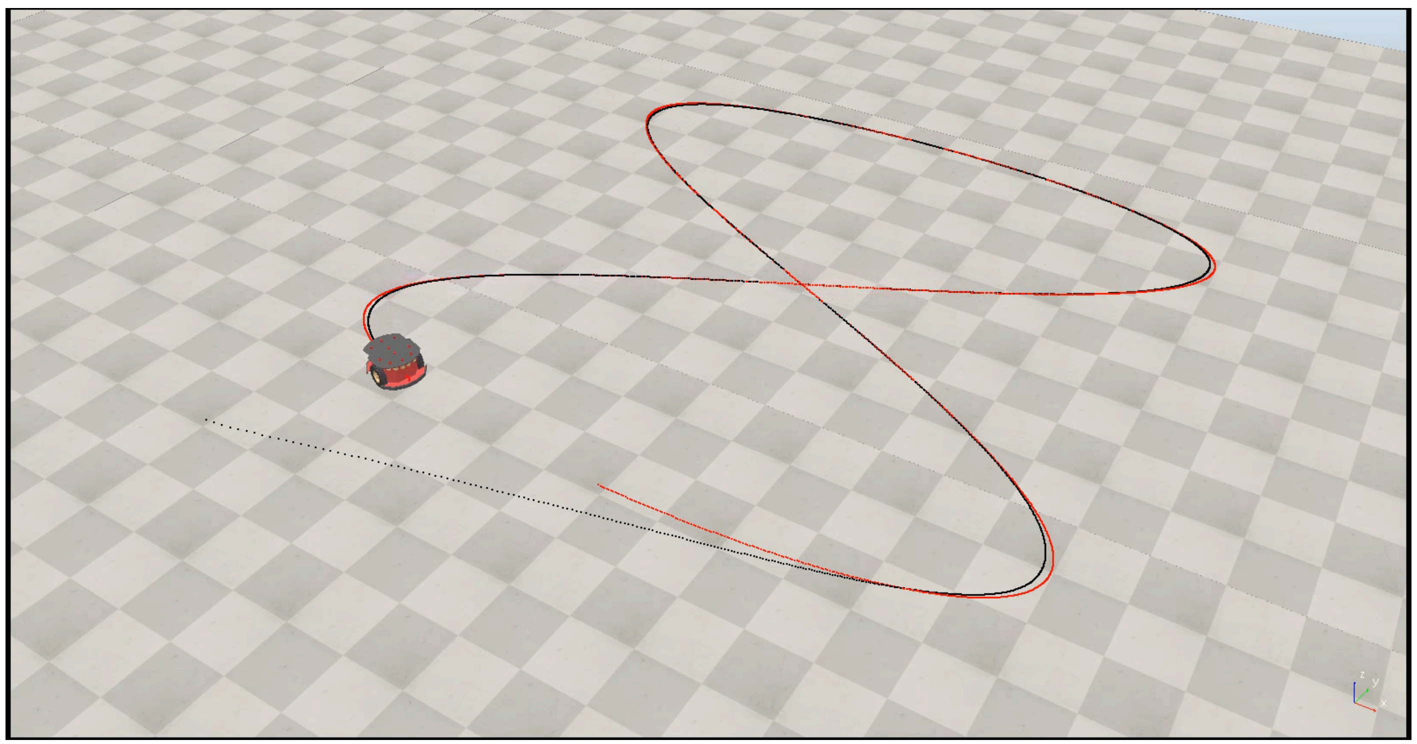


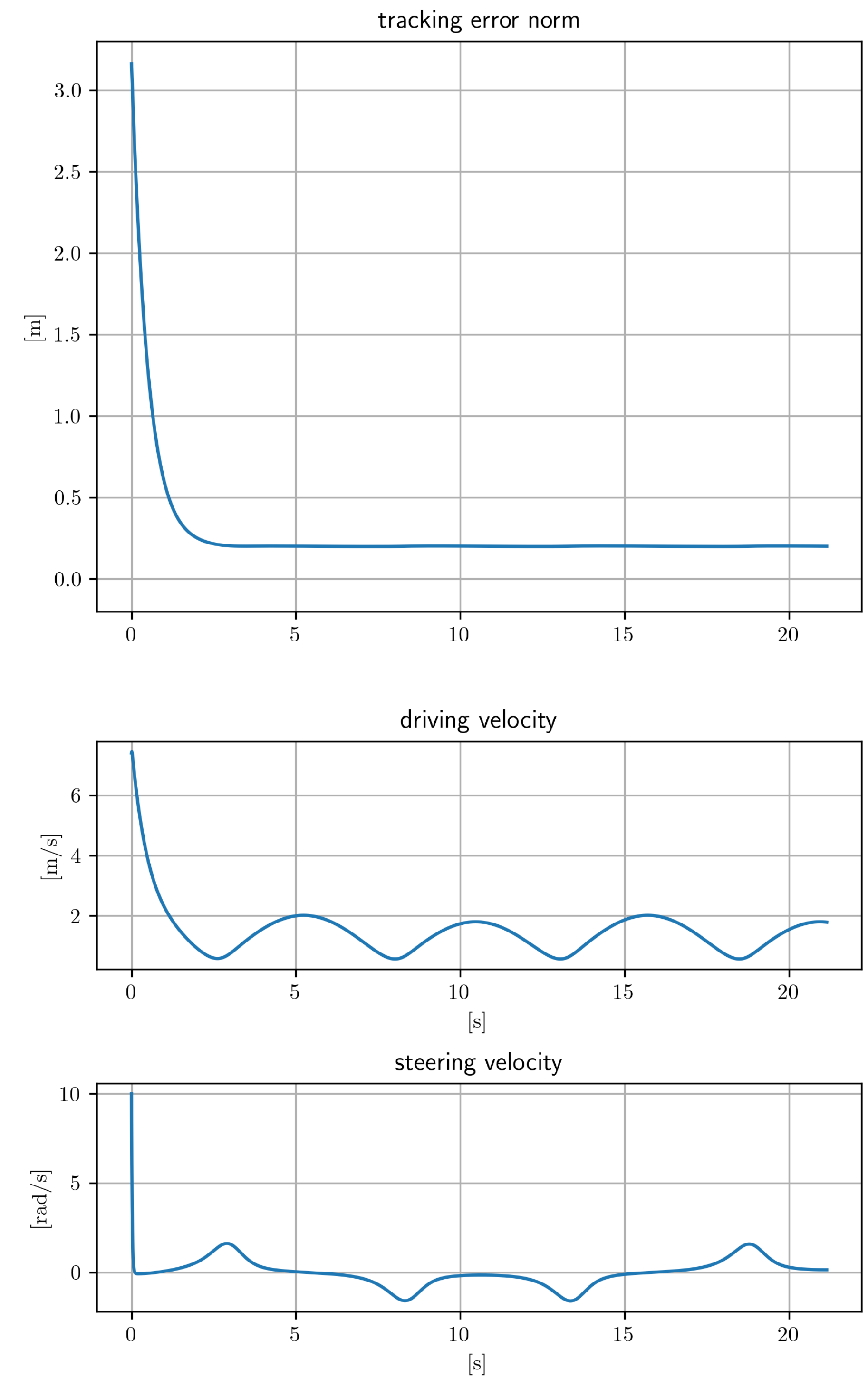


- steady-state error $= b$

simulations

tracking a figure 8 via exact i/o linearization (b=0.2)





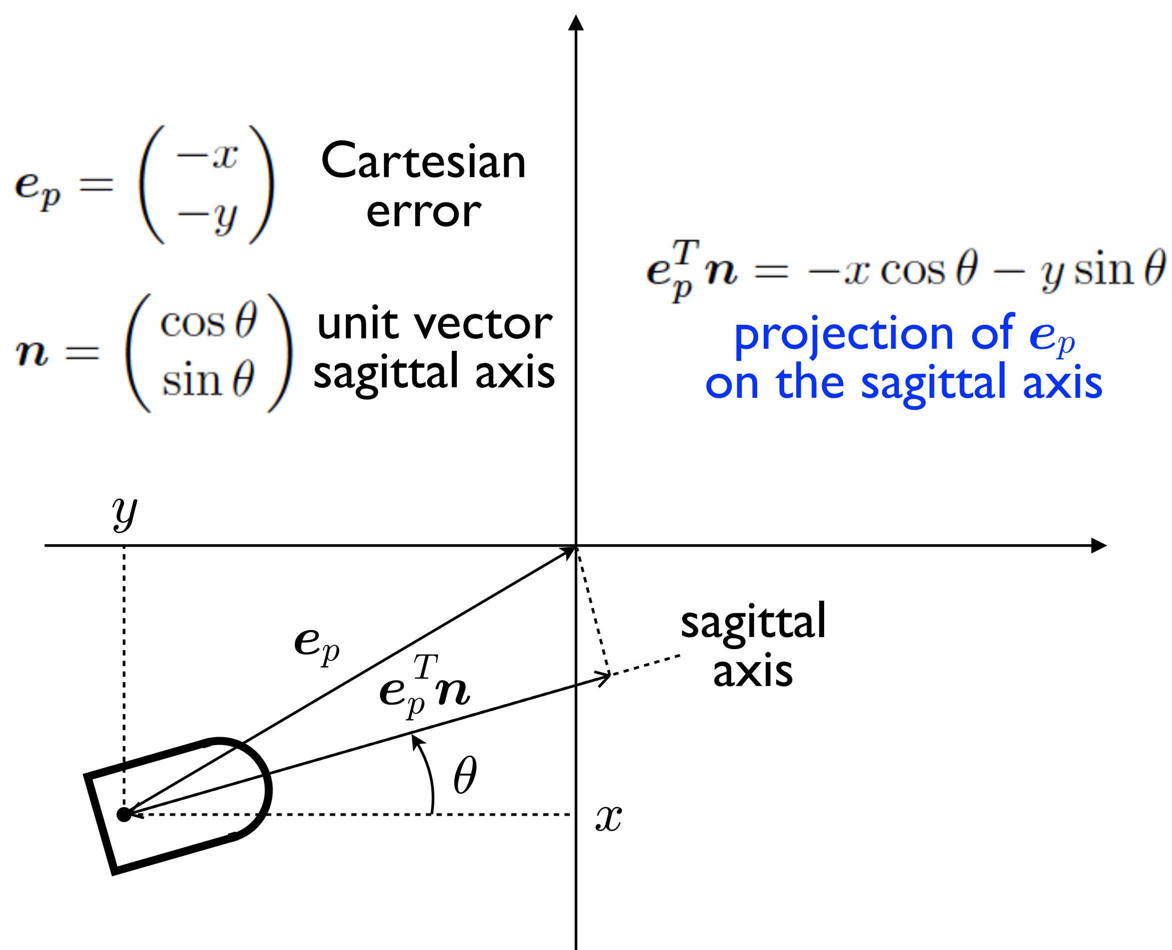
- steady-state error is now reduced but steering velocity increases

regulation

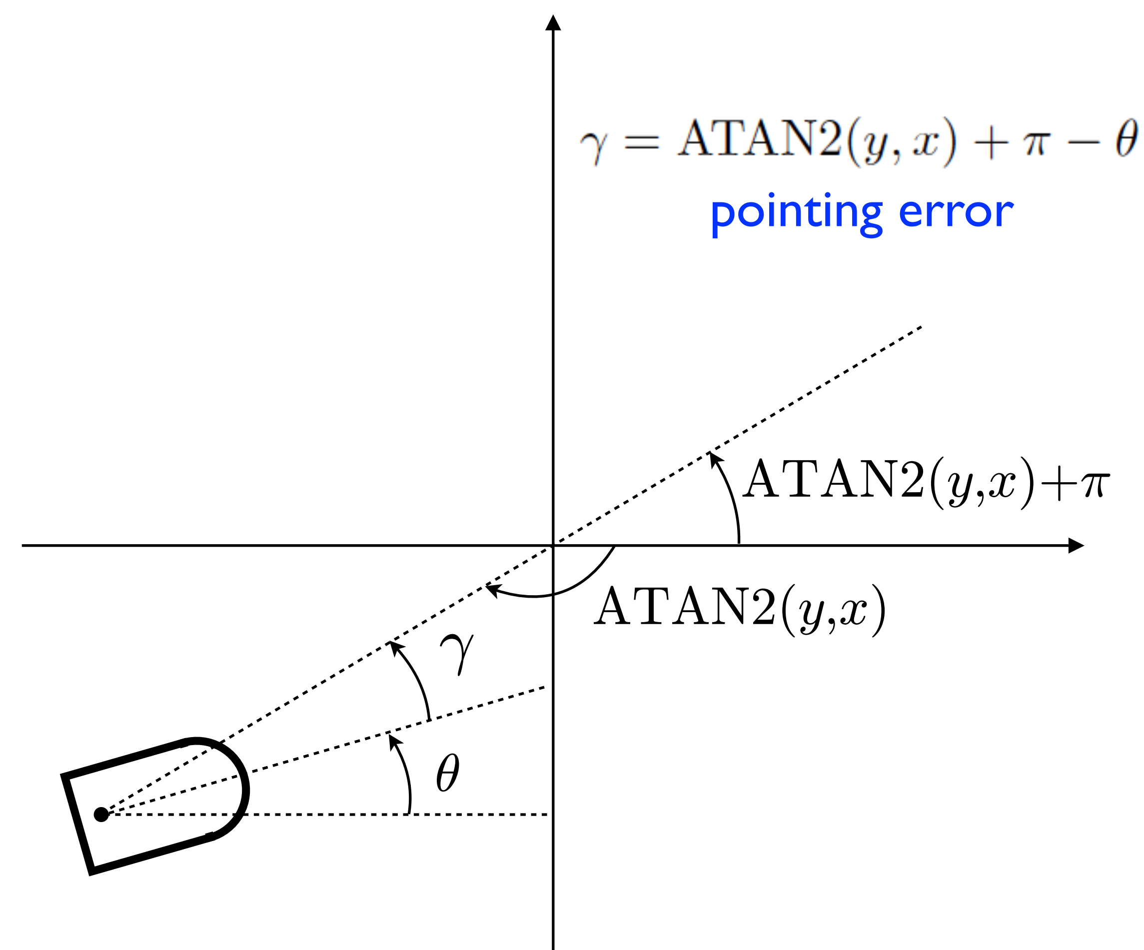
- drive the unicycle to a desired configuration $\boldsymbol{q}_d$
- the obvious approach (choose a path/trajectory that stops in $\boldsymbol{q}_d$, then track it via feedback) does not work:
  - linear/nonlinear controllers based on the error dynamics require persistent trajectories
  - i/o linearization leads point B to the destination rather than the representative point of the unicycle
- being nonholonomic, WMRs (unlike manipulators) do not admit universal controllers, i.e., controllers that can stabilize arbitrary trajectories, persistent or not

Cartesian regulation

- drive the unicycle to a given Cartesian position (w.l.o.g., the origin $(0\ 0)^T$), regardless of orientation
- geometry of the problem:



(distances)



(angles)

Cartesian regulation

- consider the feedback control law

$$v = -k_1(x\cos\theta + y\sin\theta)$$
$$\omega = k_2(\mathrm{Atan2}(y,x) - \theta + \pi)$$

- geometrical interpretation:
  - v is proportional to the orthogonal projection of the Cartesian error e_p on the sagittal axis
  - ω is proportional to the pointing error (i.e., the difference between the orientation of e_p and that of the unicycle)

- does it work? consider the Lyapunov-like function

$$V = \frac{1}{2}(x^2 + y^2)$$ positive semidefinite (PSD)

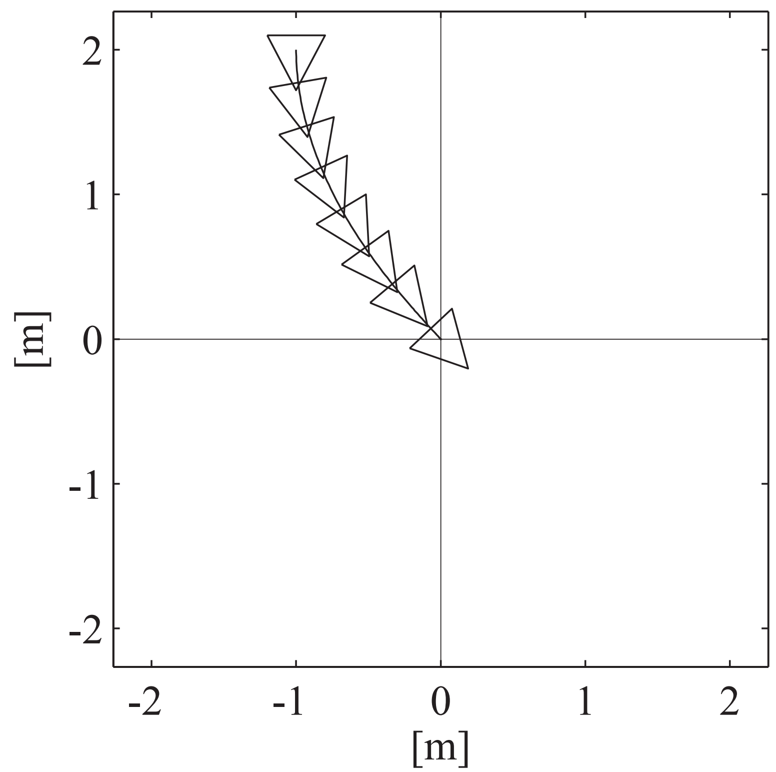
$$\dot{V} = -k_1(x\cos\theta + y\sin\theta)^2$$ negative semidefinite (NSD)

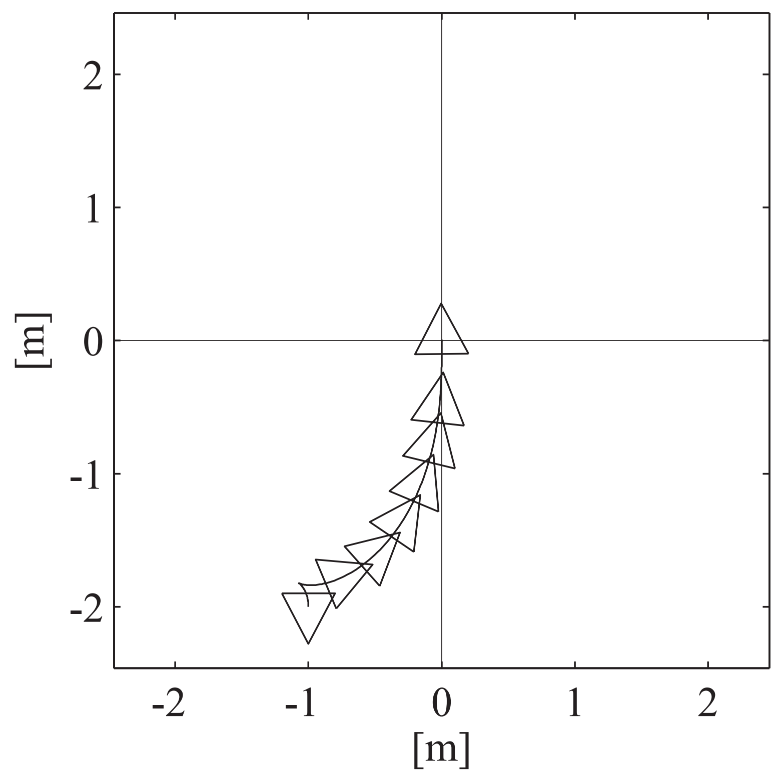
- cannot use LaSalle theorem, but being V PSD, $\dot{V}$ NSD and $\ddot{V}$ bounded (can be shown) we can use Barbalat lemma to infer that $\dot{V}$ tends to zero, i.e.

$$\lim_{t\to\infty}(x\cos\theta + y\sin\theta) = 0$$

- this implies that the Cartesian error goes to zero (the other possibility would be e_p becoming orthogonal to $\boldsymbol{n}$, but this cannot be steady-state since in such configuration it would be $v=0$ and $\omega = k_2\pi/2$)

simulation





- final orientation is not controlled
- at most one backup maneuver