

Autonomous and Mobile Robotics

Prof. Giuseppe Oriolo

Wheeled Mobile Robots

Motion Control: Introduction and Trajectory Tracking

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

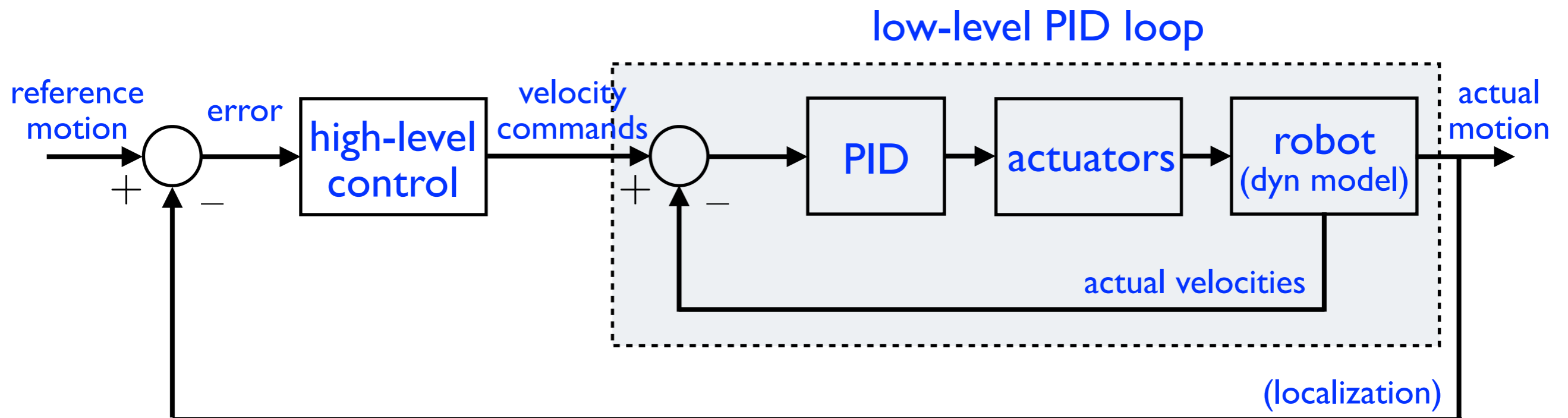


SAPIENZA
UNIVERSITÀ DI ROMA

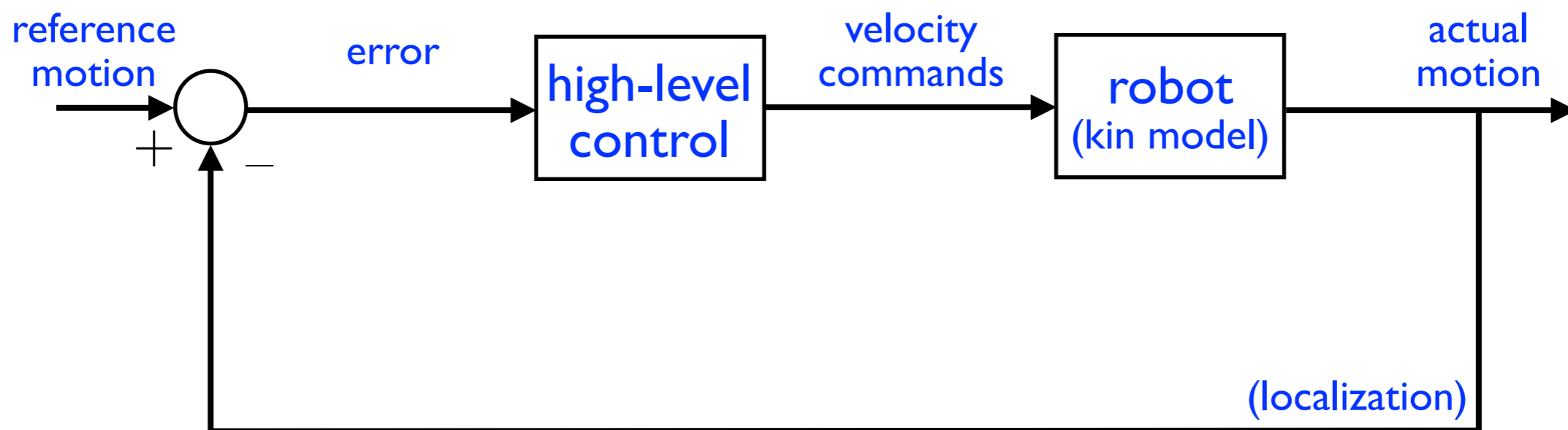
motion control

- 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

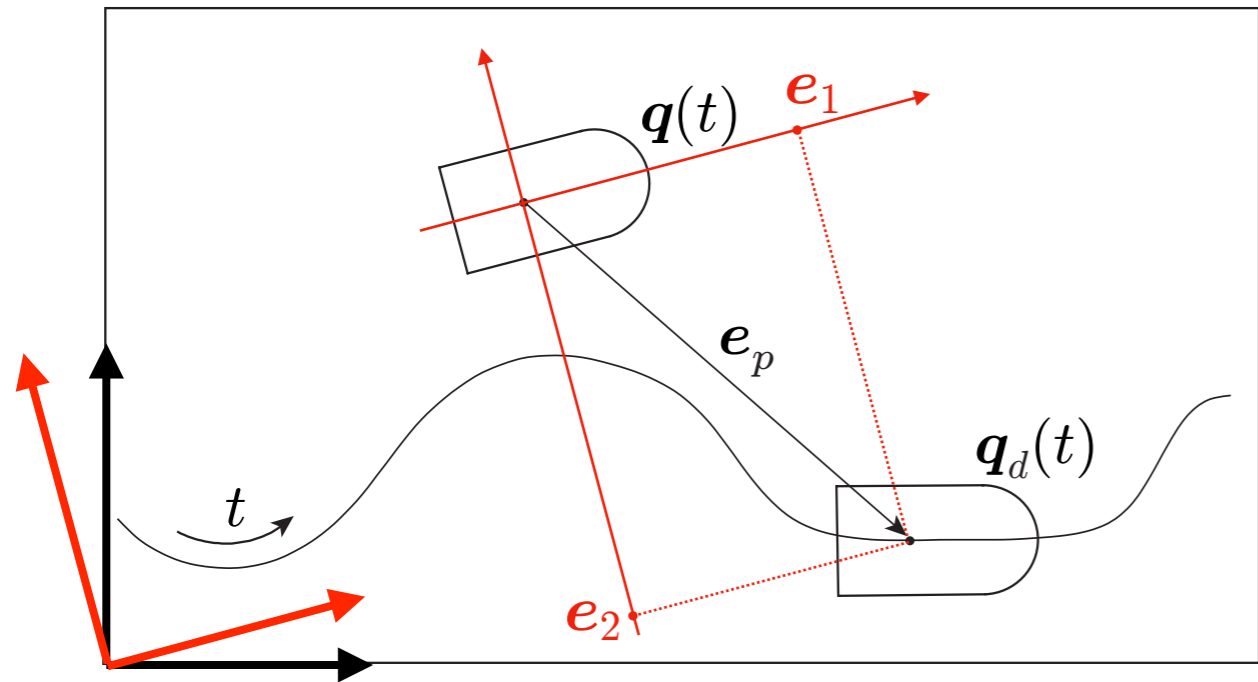
- **actual** control scheme



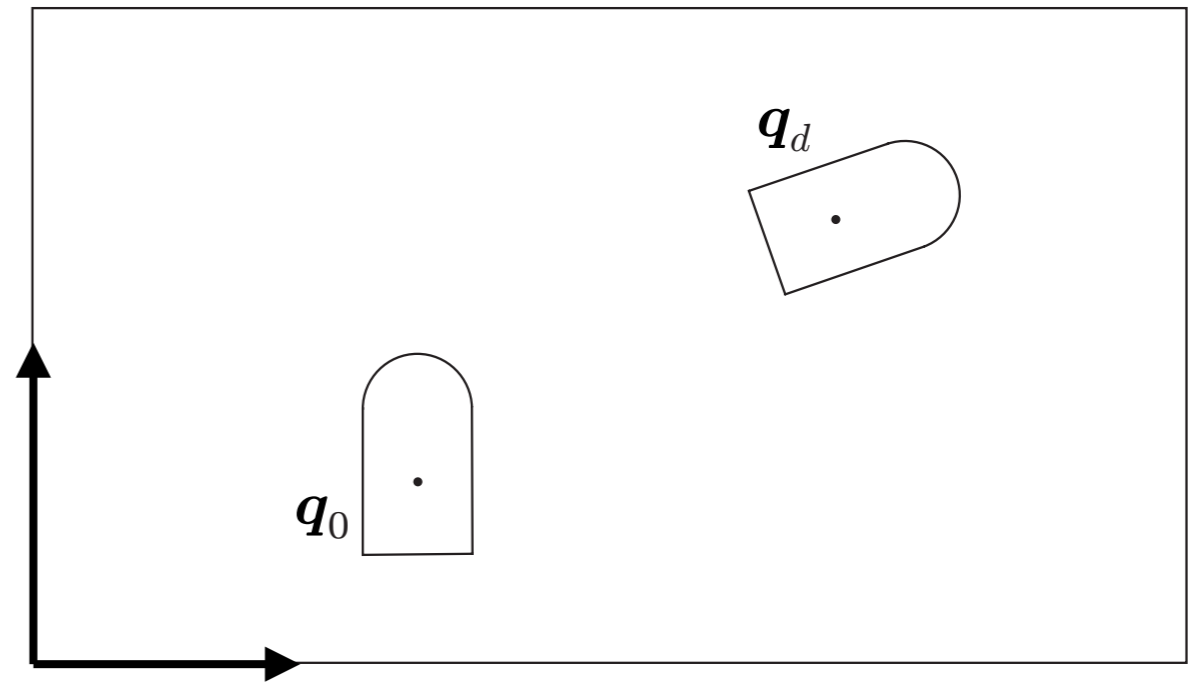
- **equivalent** control scheme (for design)



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)
- w.l.o.g., we consider a **unicycle** in the following

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

$$\dot{x}_d = v_d \cos \theta_d$$

$$\dot{y}_d = v_d \sin \theta_d$$

$$\dot{\theta}_d = \omega_d$$

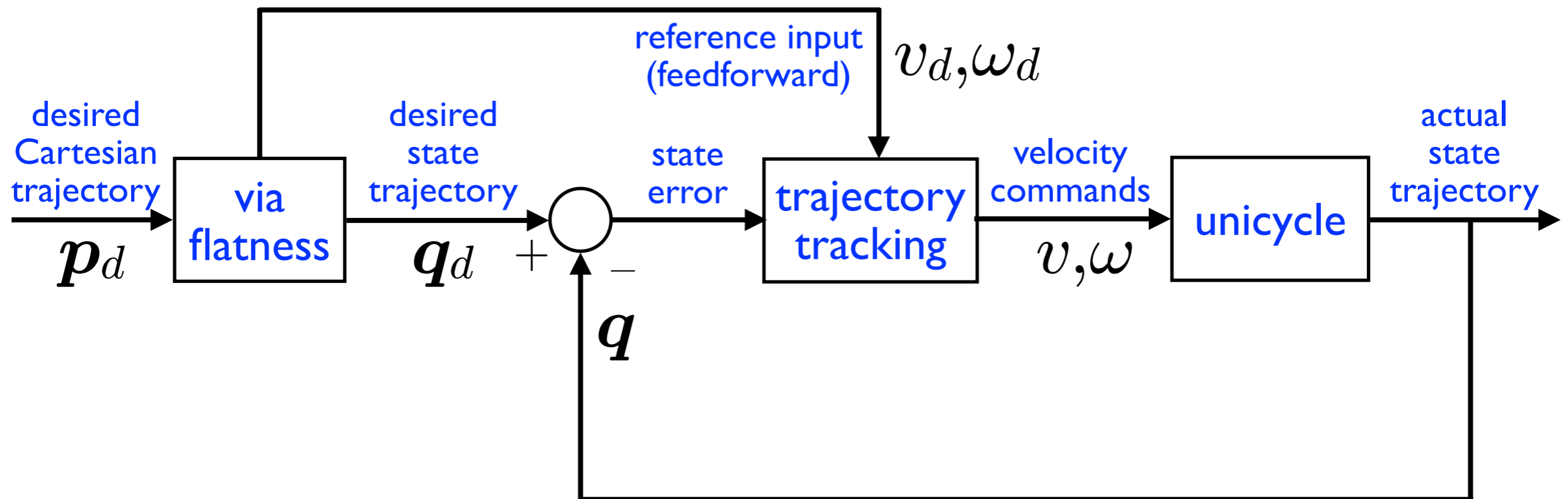
- thanks to **flatness**, from $(x_d(t), y_d(t))$ we can compute

$$\theta_d(t) = \text{Atan2}(\dot{y}_d(t), \dot{x}_d(t)) + k\pi \quad k = 0, 1$$

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

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

- 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 will be



- rather than using directly the state error $q_d - q$, use its **rotated** version defined as

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

(e_1, e_2) is the Cartesian error e_p in a frame rotated by θ (in **red** in slide 4)

- the error dynamics is nonlinear and time-varying

$$\dot{e}_1 = v_d \cos e_3 - v + e_2 \omega$$

$$\dot{e}_2 = v_d \sin e_3 - e_1 \omega$$

$$\dot{e}_3 = \omega_d - \omega$$

approximate linearization: brush-up

- idea: to stabilize a nonlinear system at an equilibrium, stabilize its **approximate linearization** around it
- for a generic nonlinear system

$$\dot{x} = \varphi(x, u) = f(x) + G(x)u \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

assume the **origin** (w.l.o.g.) is the desired **unforced** equilibrium, so that

$$\varphi(\mathbf{0}, \mathbf{0}) = \mathbf{0} \quad \text{or equivalently} \quad f(\mathbf{0}) = \mathbf{0}$$

- the Taylor expansion of φ around $x = \mathbf{0}, u = \mathbf{0}$ gives

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

- the **approximate linearization** of the nonlinear system at the origin is then defined as

$$\dot{x} = \left. \frac{\partial \varphi}{\partial x} \right|_{\substack{x=0 \\ u=0}} x + \left. \frac{\partial \varphi}{\partial u} \right|_{\substack{x=0 \\ u=0}} u = Ax + Bu$$

- now let $u = Kx$, so that

$$\dot{x} = Ax + BKx = (A + BK)x$$

if K is chosen in such a way that $A + BK$ is **Hurwitz** (certainly possible if (A, B) is controllable), then the approximate linearization is asymptotically stable

- by Lyapunov **indirect** method, this ensures the origin is **locally** asymptotically stable for the nonlinear system

via approximate linearization

- apply the approximate linearization approach to stabilize the previous error dynamics

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

- in this form, the origin is **not** an unforced equilibrium; however, this is easily rectified by using the following (invertible) **input transformation**

$$u_1 = v_d \cos e_3 - v$$

$$u_2 = \omega_d - \omega$$

- we obtain

$$\dot{e}_1 = \omega_d e_2 + u_1 - e_2 u_2$$

$$\dot{e}_2 = -\omega_d e_1 + v_d \sin e_3 + e_1 u_2$$

$$\dot{e}_3 = u_2$$

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

- note that

$$\varphi(e, u) = \overbrace{\begin{pmatrix} \omega_d e_2 \\ -\omega_d e_1 + v_d \sin e_3 \\ 0 \end{pmatrix}}^{f(e)} + \overbrace{\begin{pmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{pmatrix}}^{G(e)} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

- the linear approximation of the error dynamics is

$$\dot{e} = \left. \frac{\partial \varphi}{\partial e} \right|_{\substack{e=0 \\ u=0}} e + \left. \frac{\partial \varphi}{\partial u} \right|_{\substack{e=0 \\ u=0}} u = A e + B u$$

- one easily finds

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

- letting $e = \mathbf{0}$, $u = \mathbf{0}$ gives

$$A = \begin{pmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

note that A is actually $A(t)$ due to the dependence of the references inputs v_d and ω_d on time \Rightarrow the linear approximation will be a **time-varying** system!

- wrapping up, the linearized approximation of the error dynamics around the reference trajectory is

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

- now define the **linear** feedback

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

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

$$\dot{e} = (A(t) + BK)e = A_{cl}(t)e = \begin{pmatrix} -k_1 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & -k_2 & -k_3 \end{pmatrix} e$$

- letting

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

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

$$p(\lambda) = (\lambda + 2\zeta a)(\lambda^2 + 2\zeta a\lambda + a^2)$$

real
negative
eigenvalue

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)

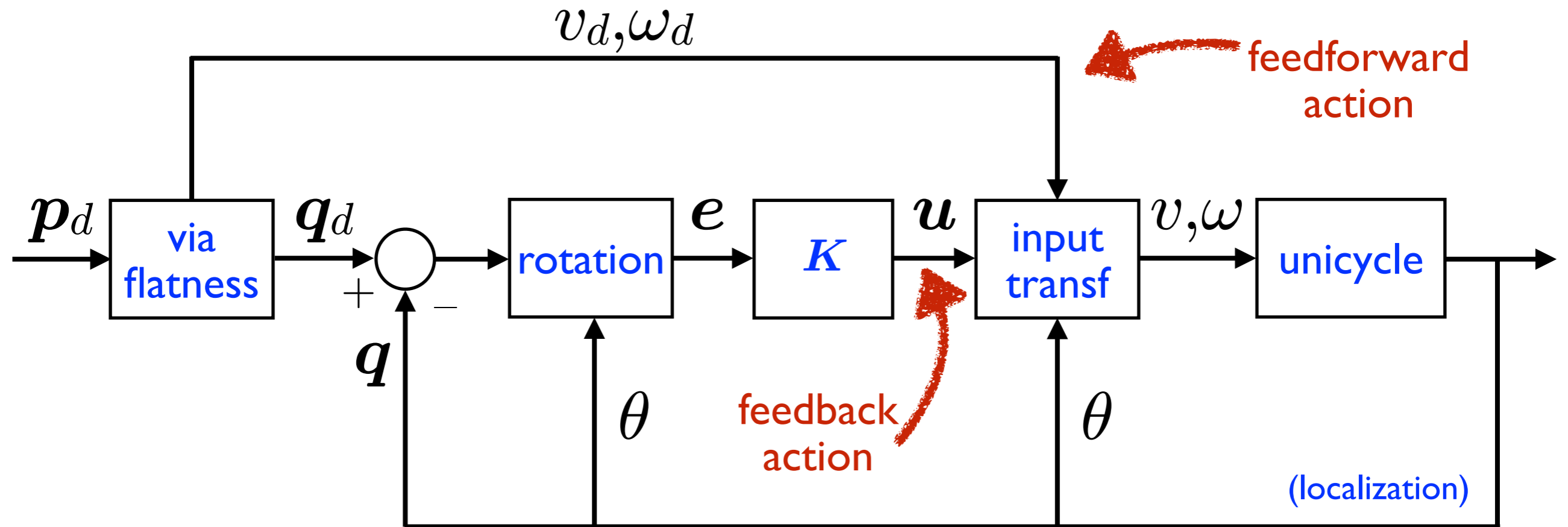
- the **actual** velocity inputs v, ω are obtained plugging the feedback controls u_1, u_2 in the input transformation
- note: $(v, \omega) \rightarrow (v_d, \omega_d)$ as $e \rightarrow 0$ (**pure feedforward**)
- note: $k_2 \rightarrow \infty$ as $v_d \rightarrow 0$, hence this controller can only 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

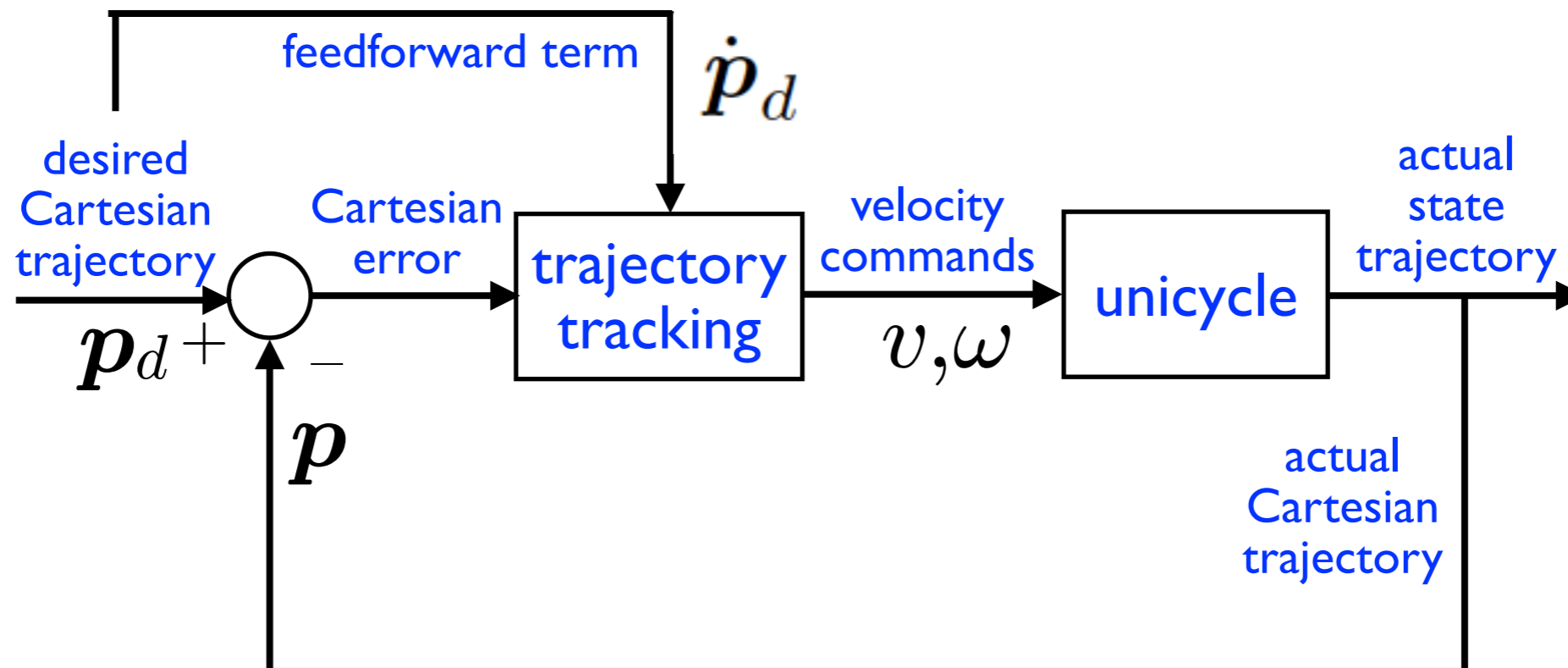
- the final block scheme for trajectory tracking via state error feedback and approximate linearization is



- 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 velocity along the reference trajectory
- the resulting block scheme will be

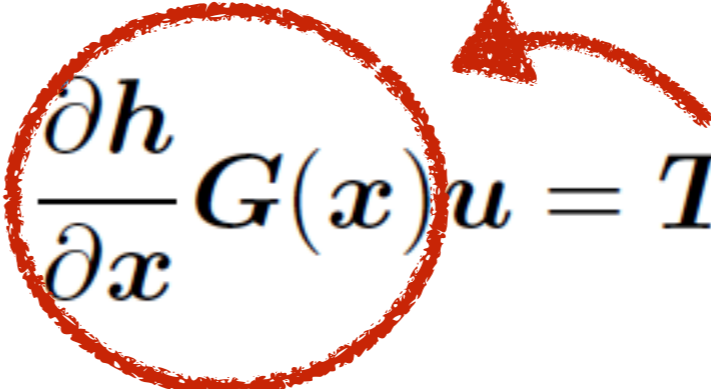


exact i/o linearization: brush-up

- consider a driftless nonlinear system

$$\begin{aligned}\dot{x} &= G(x)u \\ y &= h(x)\end{aligned}\quad x \in \mathbb{R}^n, u \in \mathbb{R}^m, y \in \mathbb{R}^m$$

- being

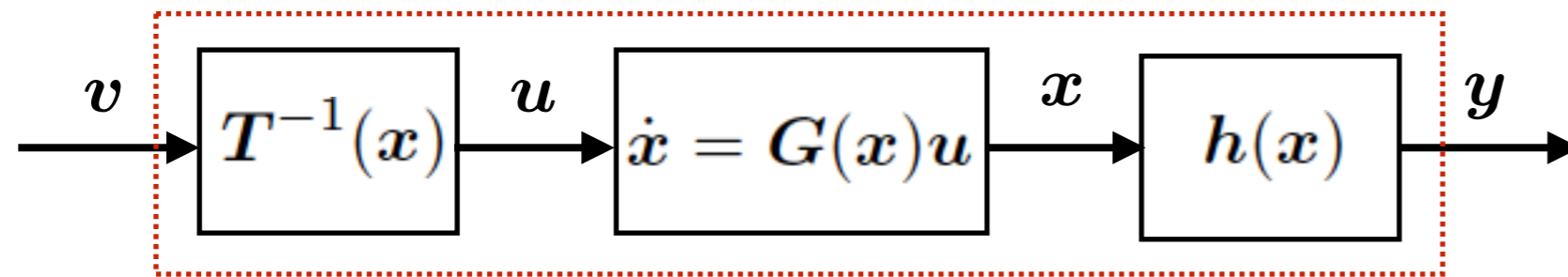
$$\dot{y} = \frac{\partial h}{\partial x} \dot{x} = \frac{\partial h}{\partial x} G(x)u = T(x)u$$


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

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

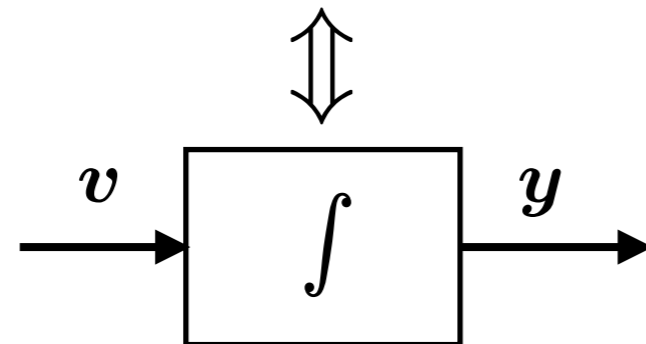
i.e., an exactly linear map between the new inputs v and (the time derivative of) the outputs

- in pictures



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

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 $u = T^{-1}(x)(\dot{y}_d + Ke)$

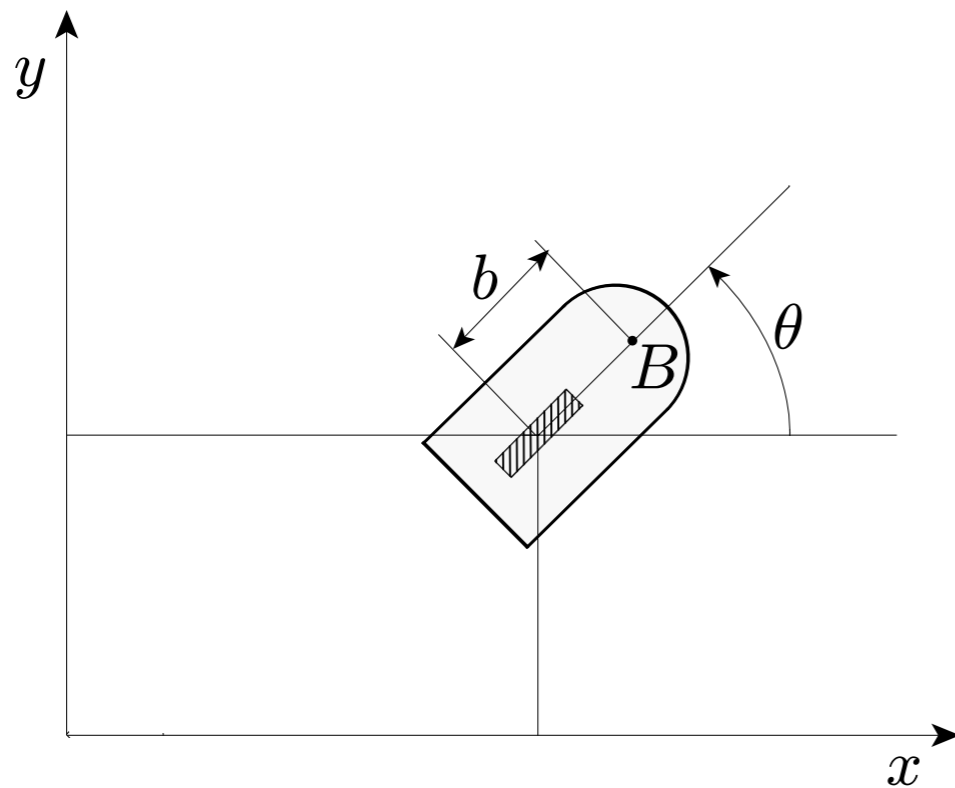
via i/o linearization (static feedback)

- let us adopt the exact i/o linearization approach to design a Cartesian trajectory tracking controller for the unicycle
- however, in this case the decoupling matrix associated to the Cartesian position turns out to be **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, exact input-output linearization is **not possible** for the output (x, y)

- 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

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} \cos \theta & -b \sin \theta \\ \sin \theta & b \cos \theta \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} = \mathbf{T}(\theta) \begin{pmatrix} v \\ \omega \end{pmatrix}$$

$$\det = b$$

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

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

obtaining an **input-output linearized** system

$$\dot{y}_1 = u_1$$

$$\dot{y}_2 = u_2$$

$$\dot{\theta} = \frac{u_2 \cos \theta - u_1 \sin \theta}{b}$$

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

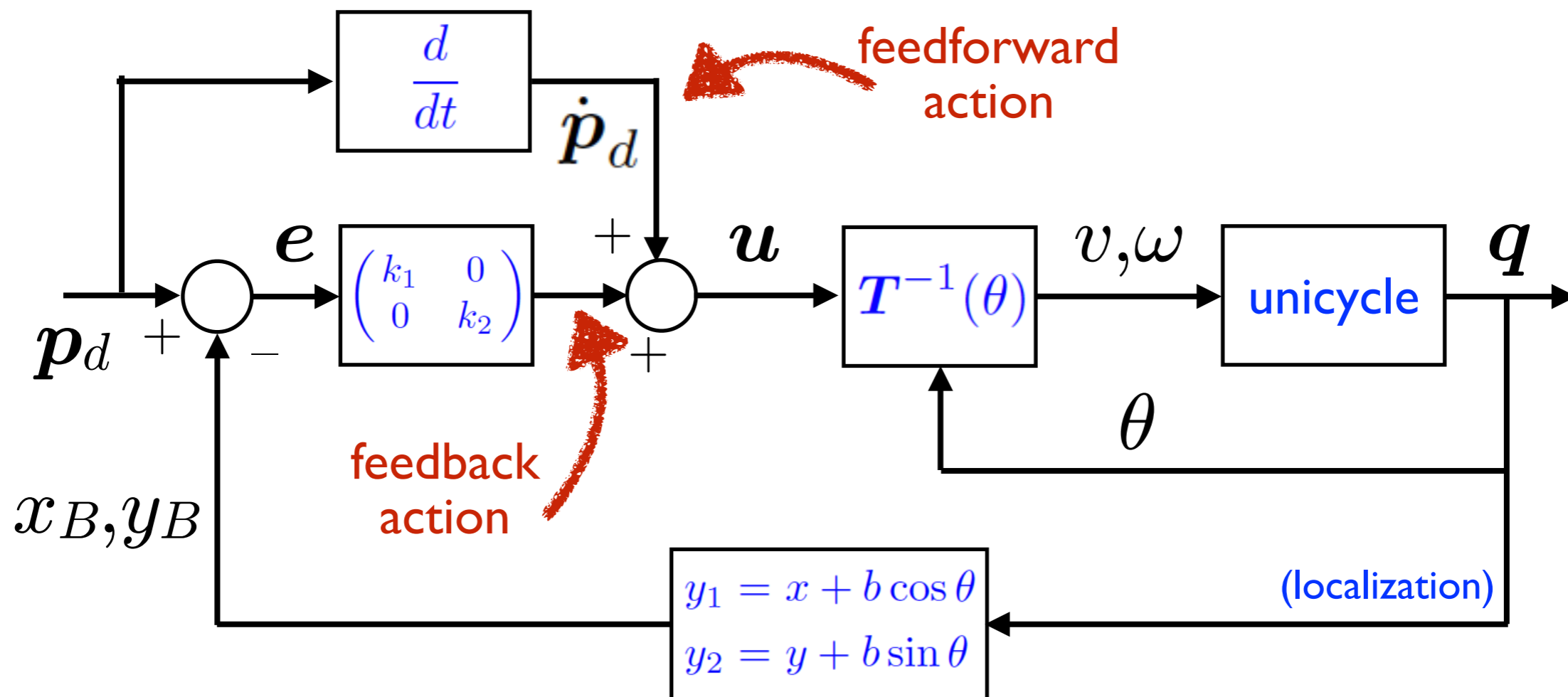
$$u_1 = \dot{x}_d + k_1(x_d - x_B)$$

$$u_2 = \dot{y}_d + k_2(y_d - y_B)$$

with $k_1, k_2 > 0$

- θ is **not** controlled with this scheme, which is based on **output error** feedback (there is a **zero dynamics**); still, it must evolve as dictated by flatness...
- the desired trajectory for B can be **arbitrary**; in particular, square corners may be included

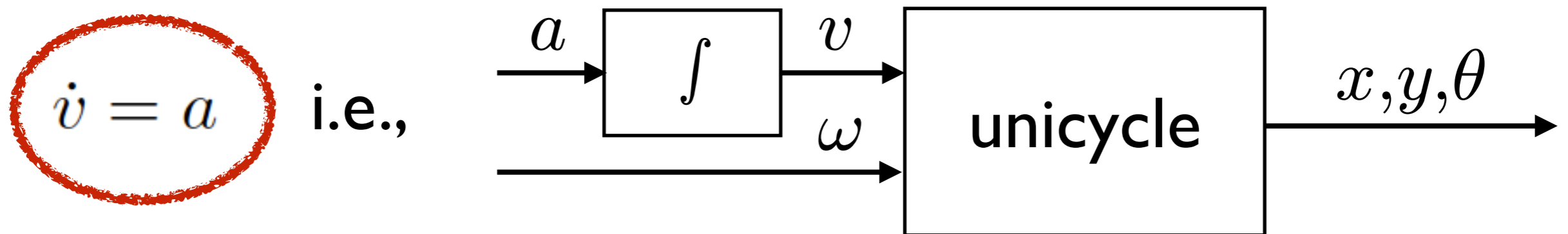
- the final block scheme for trajectory tracking via output error feedback + static i/o linearization is



- a **static** controller based on **output error**
- needs \dot{p}_d
- needs x, y, θ for output reconstruction and θ also for input transformation

via i/o linearization (dynamic feedback)

- rather than displacing the controlled output to B , we can **keep the (flat) output** (x, y) and achieve exact linearization by using a **dynamic** compensator
- to do this, perform a **dynamic extension** on the driving velocity channel ('slow down' the input)




- we can now differentiate further the output

$$\ddot{x} = a \cos \theta - v \sin \theta \omega$$

$$\ddot{y} = a \sin \theta + v \cos \theta \omega$$

- the new decoupling matrix is nonsingular if $v \neq 0$

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \cos \theta & -v \sin \theta \\ \sin \theta & v \cos \theta \end{pmatrix} \begin{pmatrix} a \\ \omega \end{pmatrix} = \mathbf{T}(v, \theta) \begin{pmatrix} a \\ \omega \end{pmatrix}$$



 $\det = v$

- under this assumption, we may set

$$\begin{pmatrix} a \\ \omega \end{pmatrix} = \mathbf{T}^{-1}(v, \theta) \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\frac{\sin \theta}{v} & \frac{\cos \theta}{v} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

obtaining a **fully linearized** system

$$\ddot{x} = u_1$$

$$\ddot{y} = u_2$$

- achieve **global exponential convergence** of x, y to the desired trajectory by letting

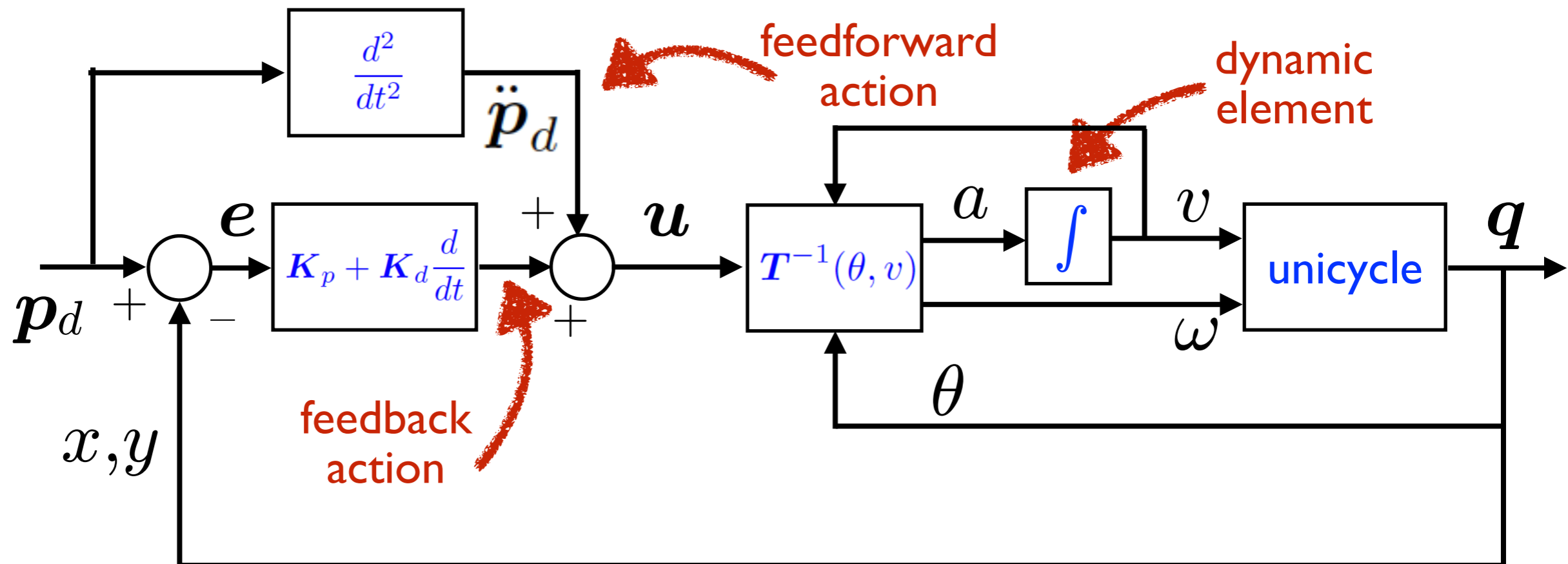
$$u_1 = \ddot{x}_d + k_{p1}(x_d - x) + k_{d1}(\dot{x}_d - \dot{x})$$

$$u_2 = \ddot{y}_d + k_{p2}(y_d - y) + k_{d2}(\dot{y}_d - \dot{y})$$

with $k_{p1}, k_{p2}, k_{d1}, k_{d2} > 0$

- since we are controlling the original flat output, there is no **zero dynamics** with this approach
- the desired trajectory must be **twice differentiable** and **persistent**, i.e., it must be $v_d \neq 0$ always (this singularity is structural in nonholonomic systems)

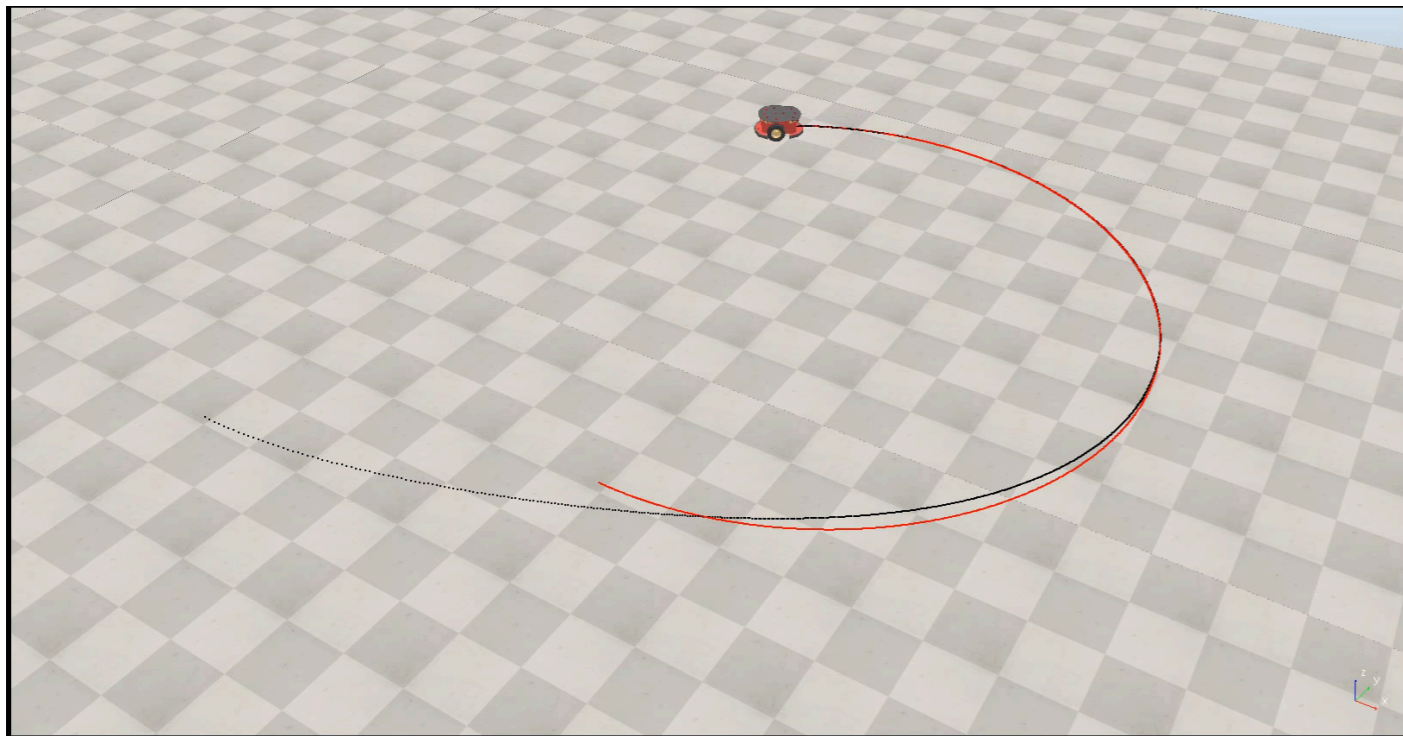
- the final block scheme for trajectory tracking via output error feedback + dynamic i/o linearization is



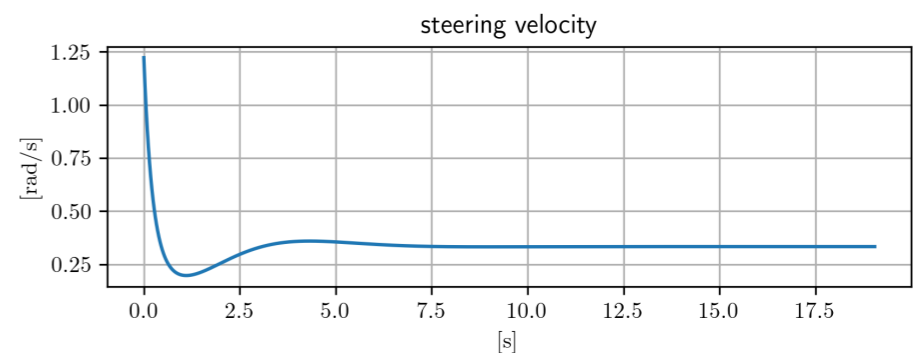
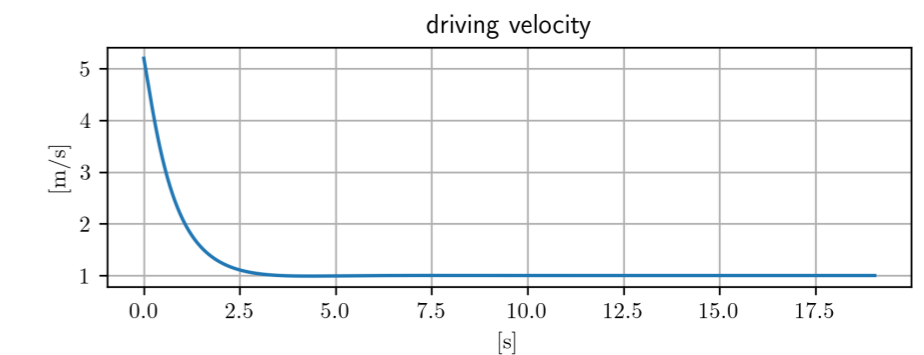
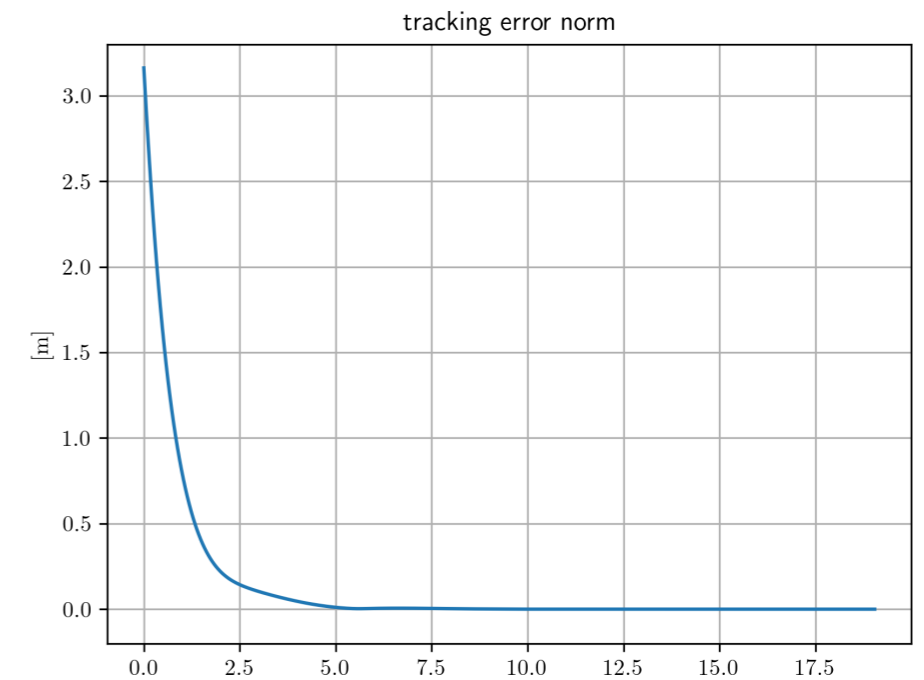
- a **dynamic** controller based on **output error**
- needs \dot{p}_d and \ddot{p}_d
- needs x, y for error computation and θ for input transformation

simulations

tracking a circle via approximate linearization

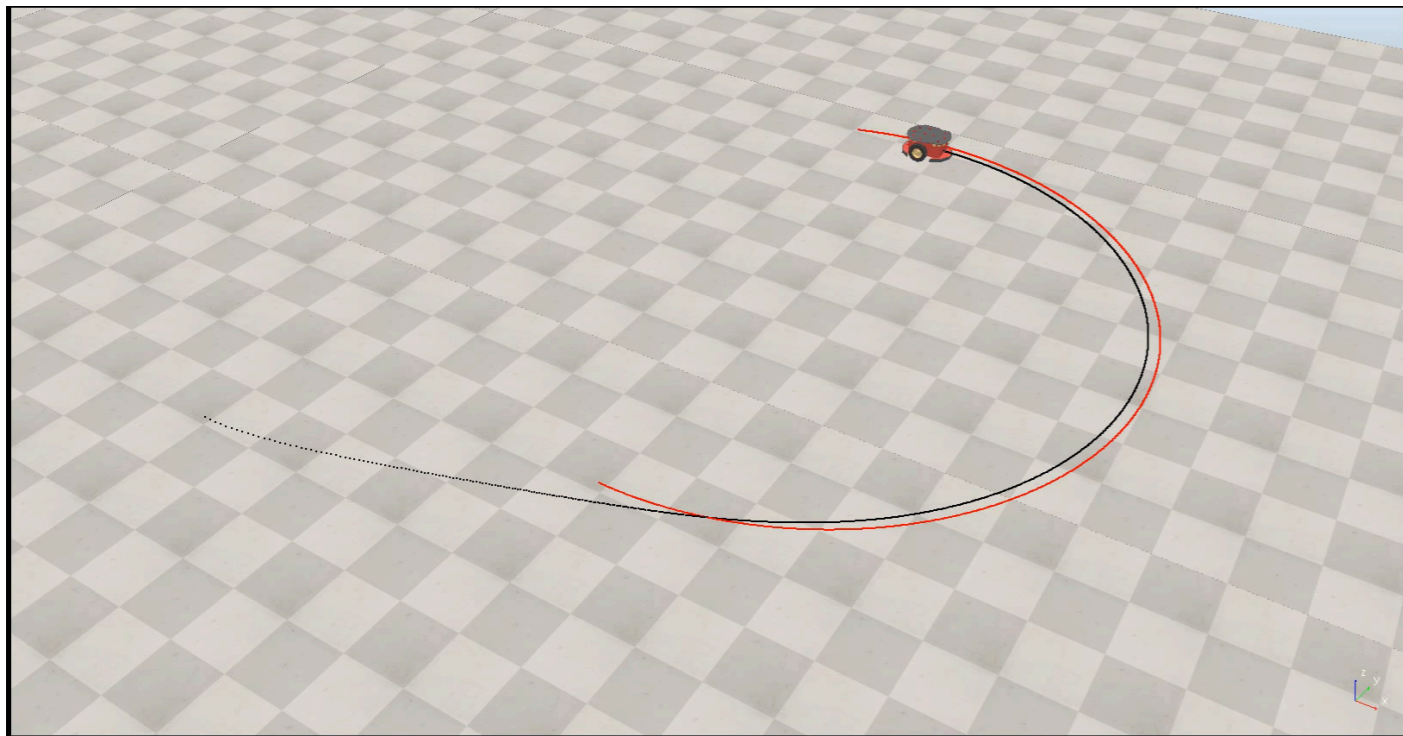


- only local stability is guaranteed

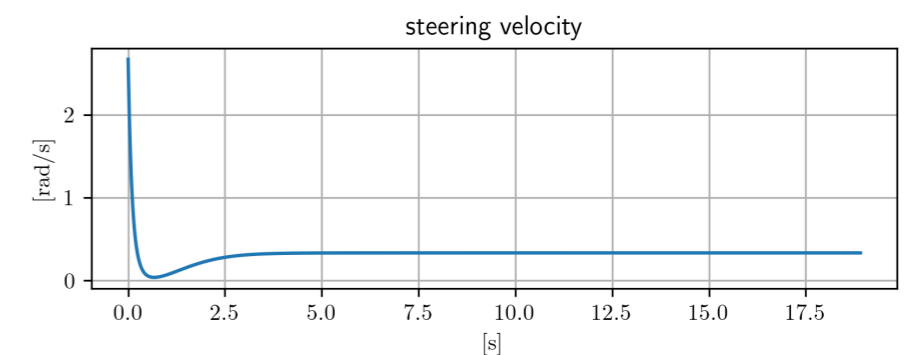
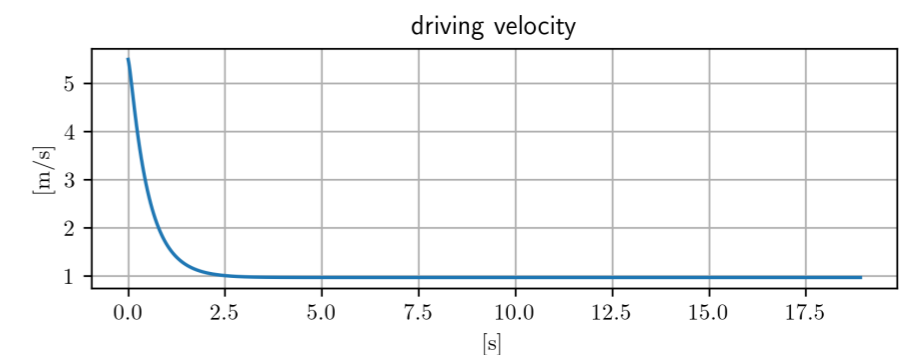
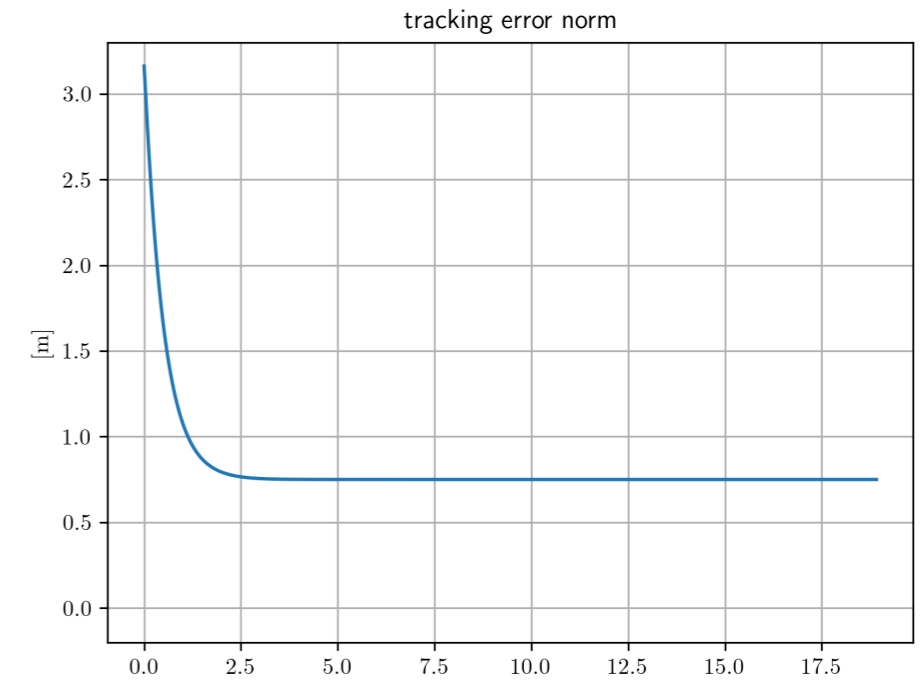


simulations

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

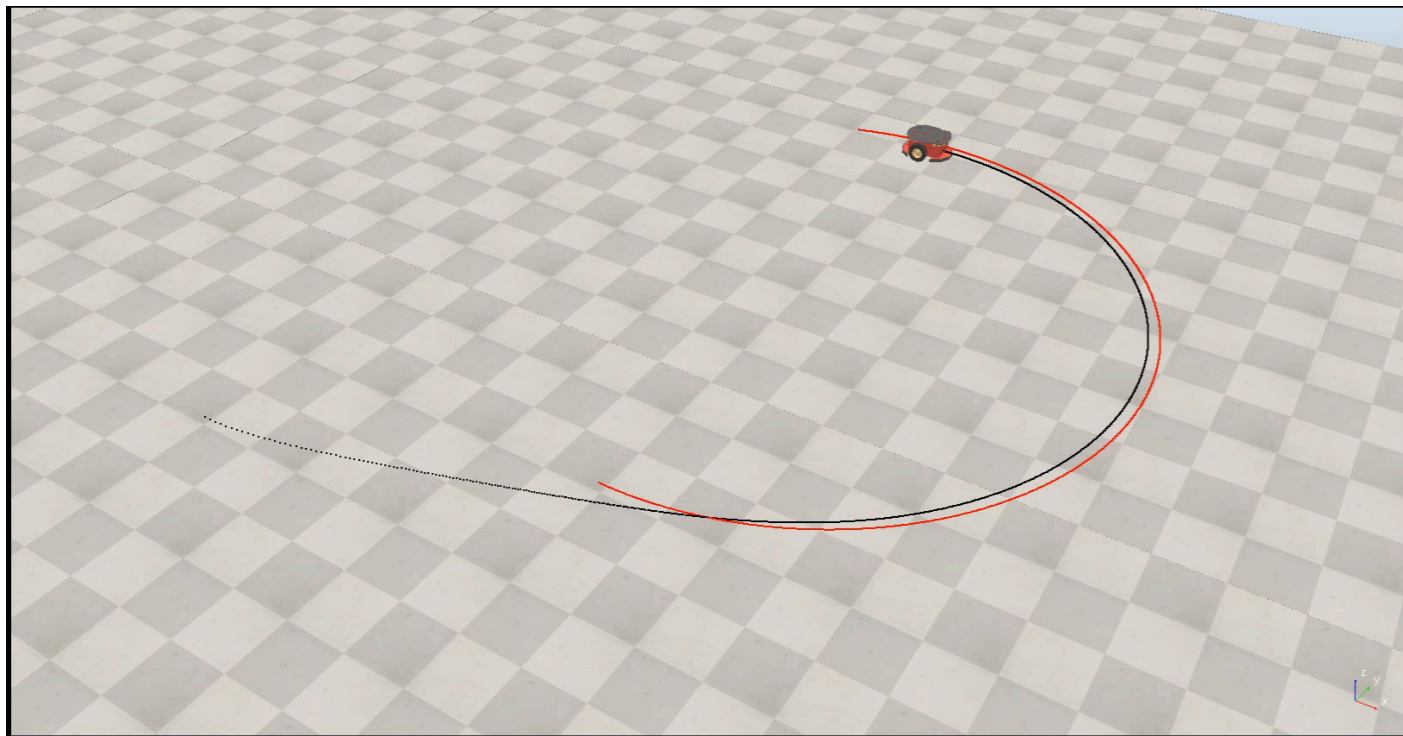


- steady-state error = b

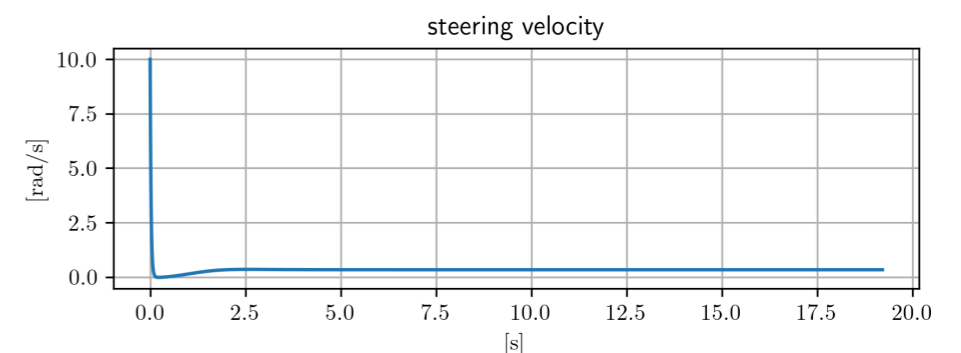
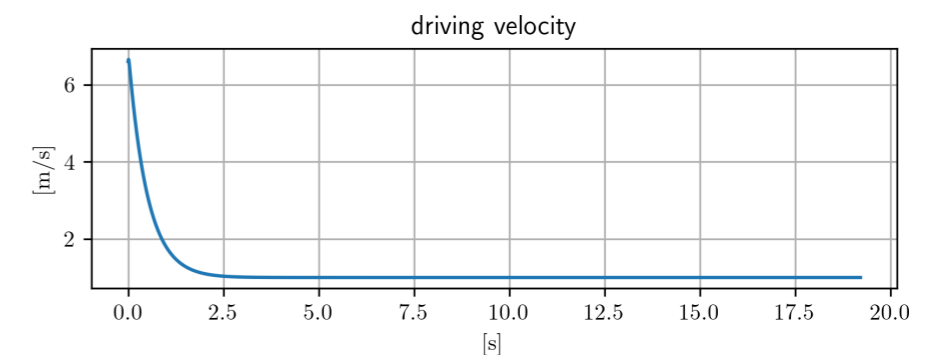
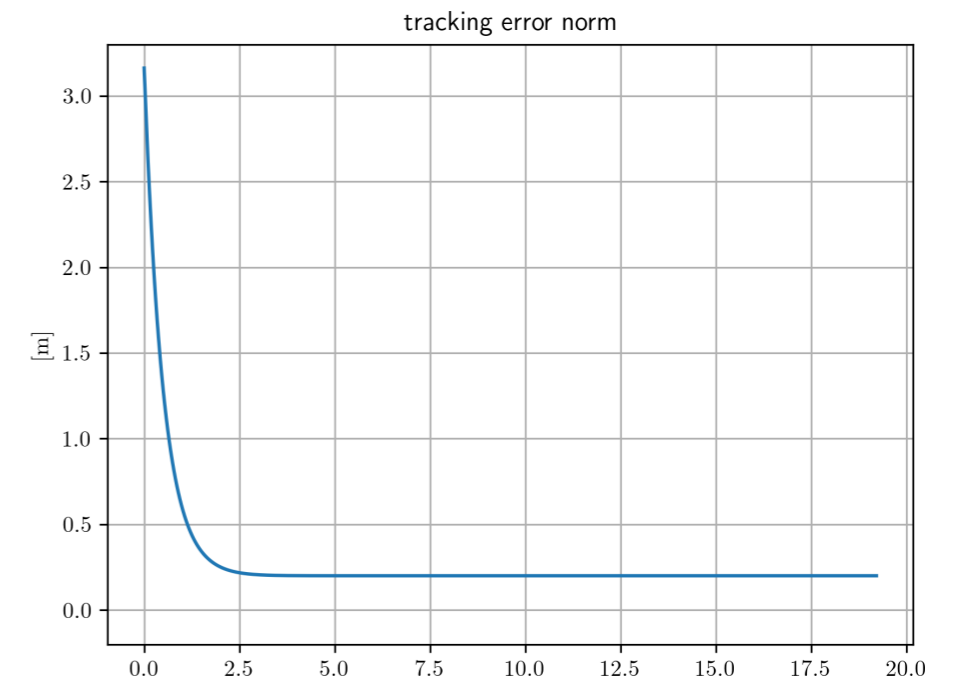


simulations

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

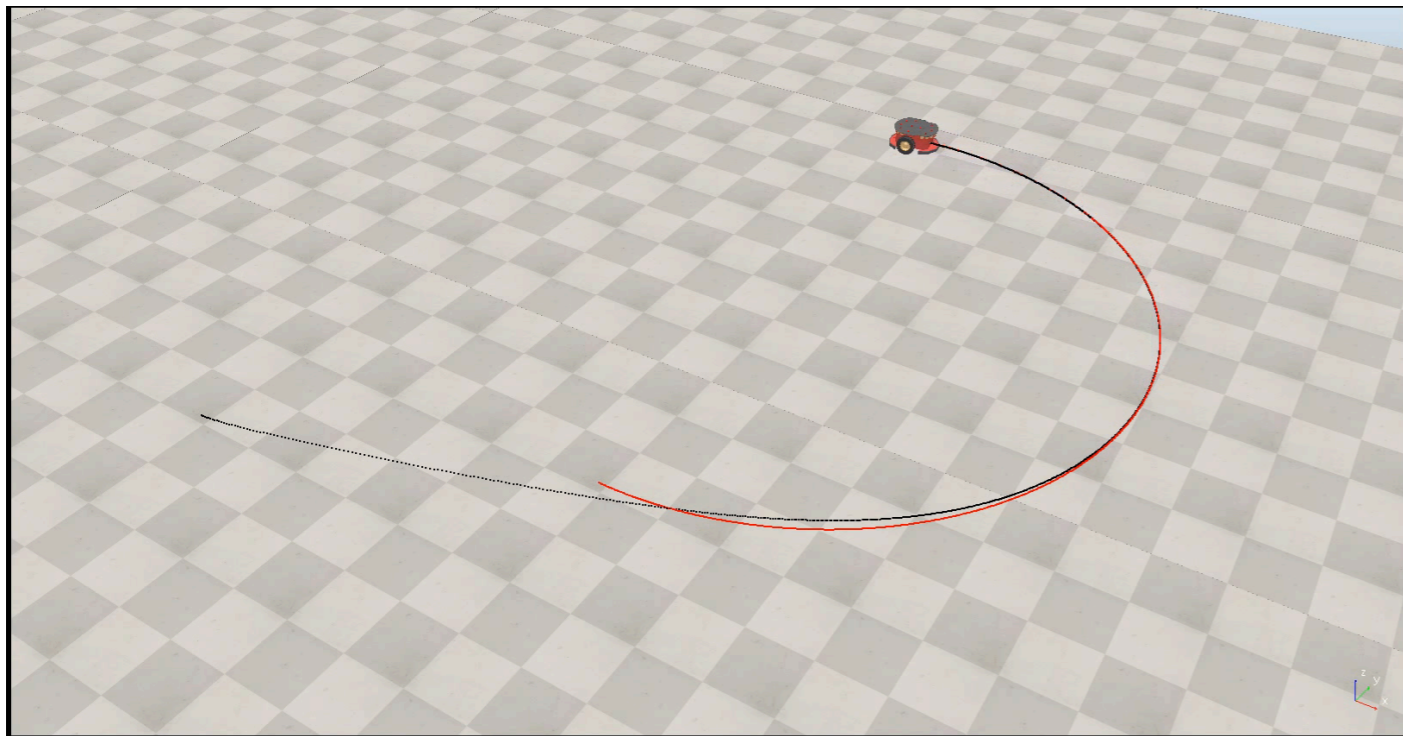


- steady-state error is now reduced but steering velocity increases

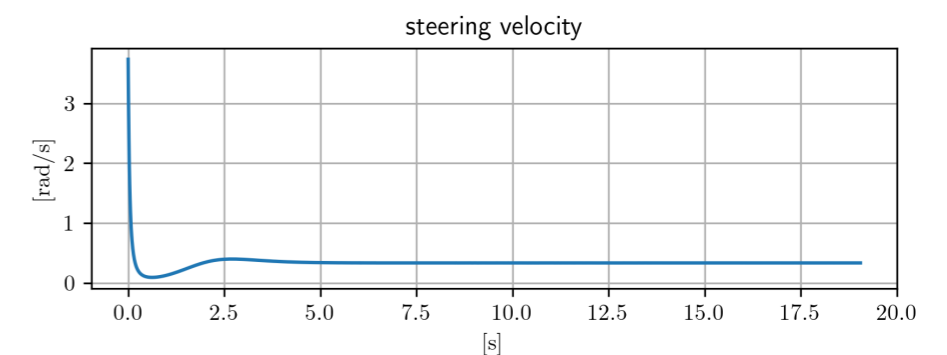
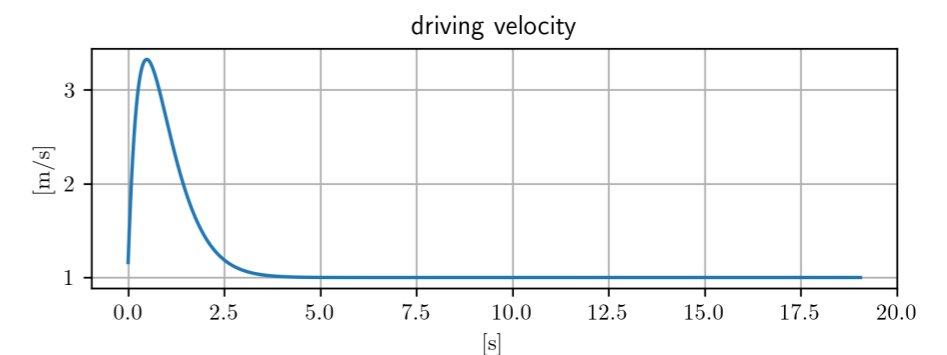
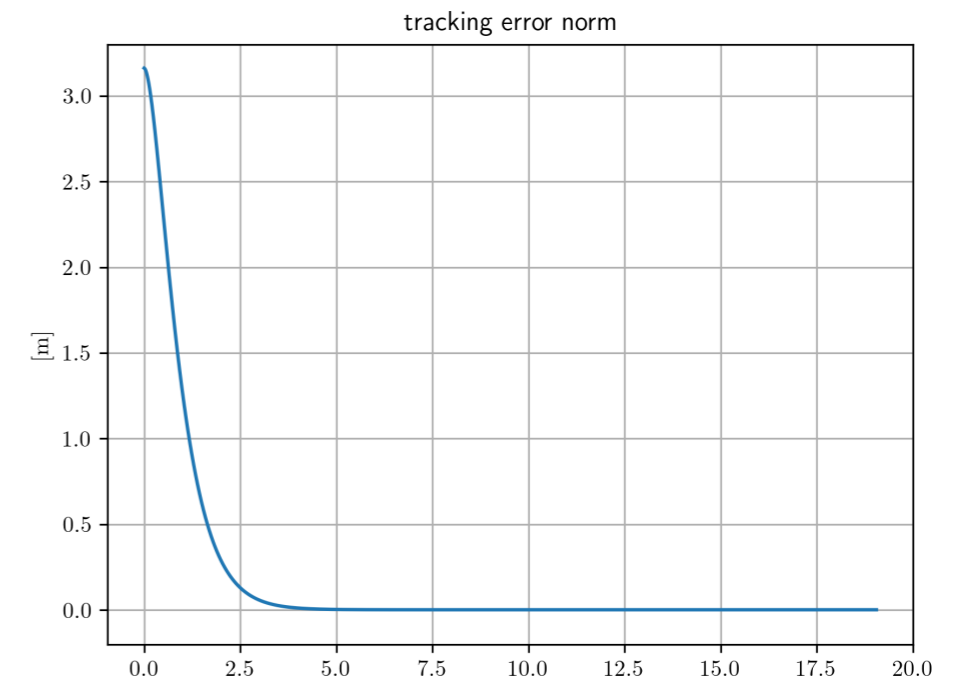


simulations

tracking a circle via dynamic i/o linearization

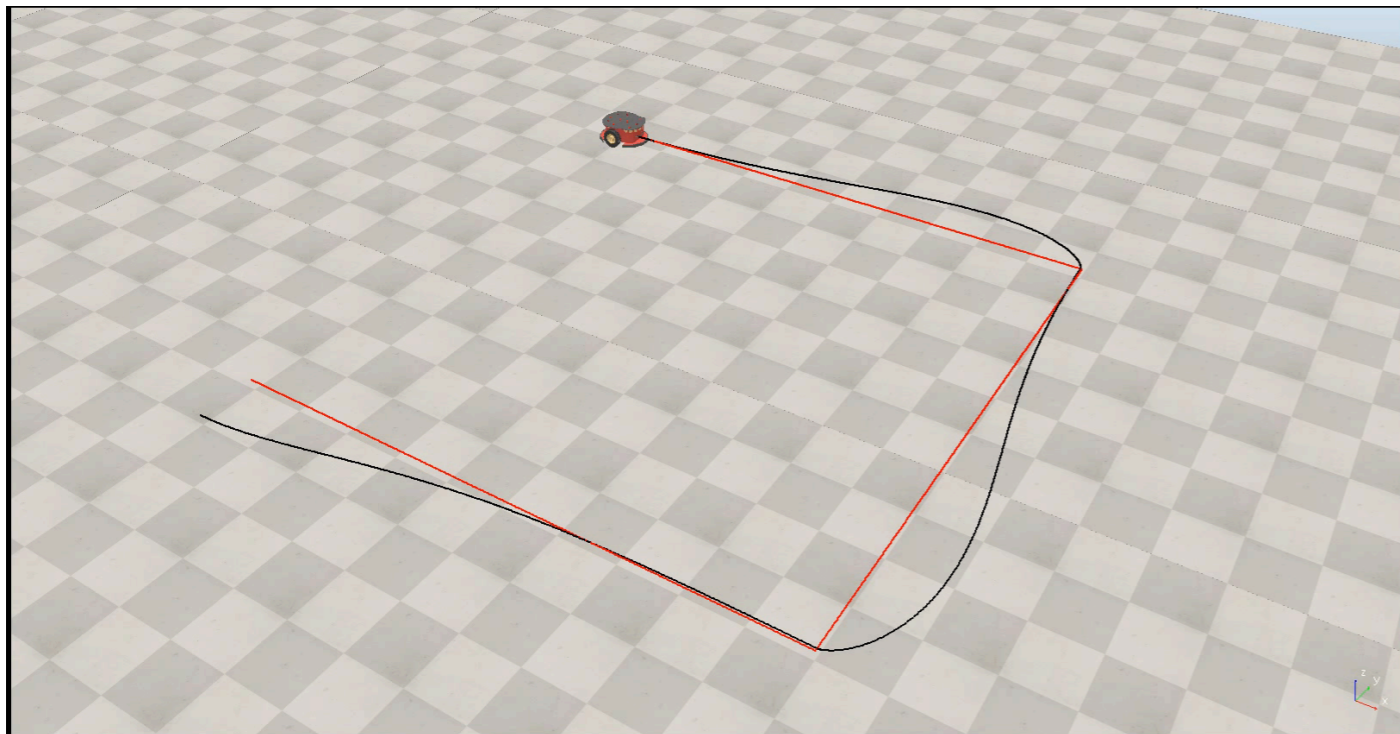


- zero steady-state error and reasonable velocities

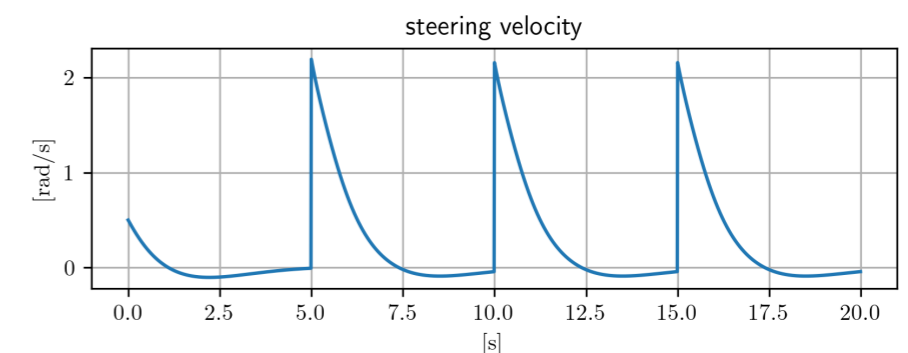
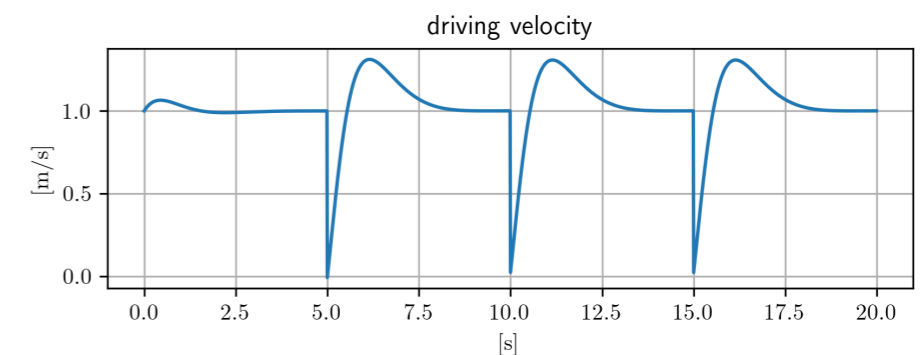
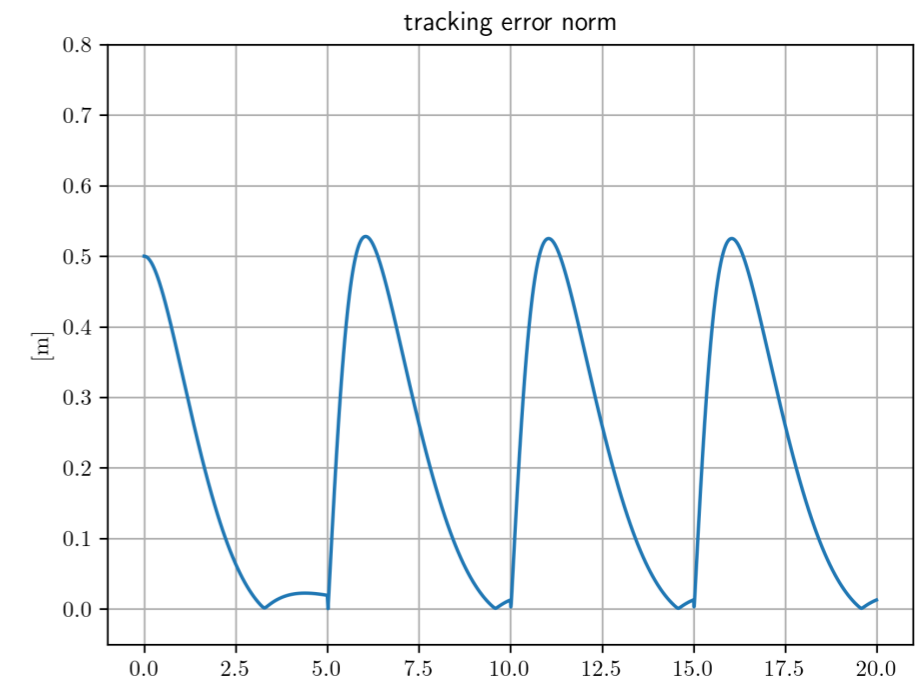


simulations

tracking a square via approximate linearization

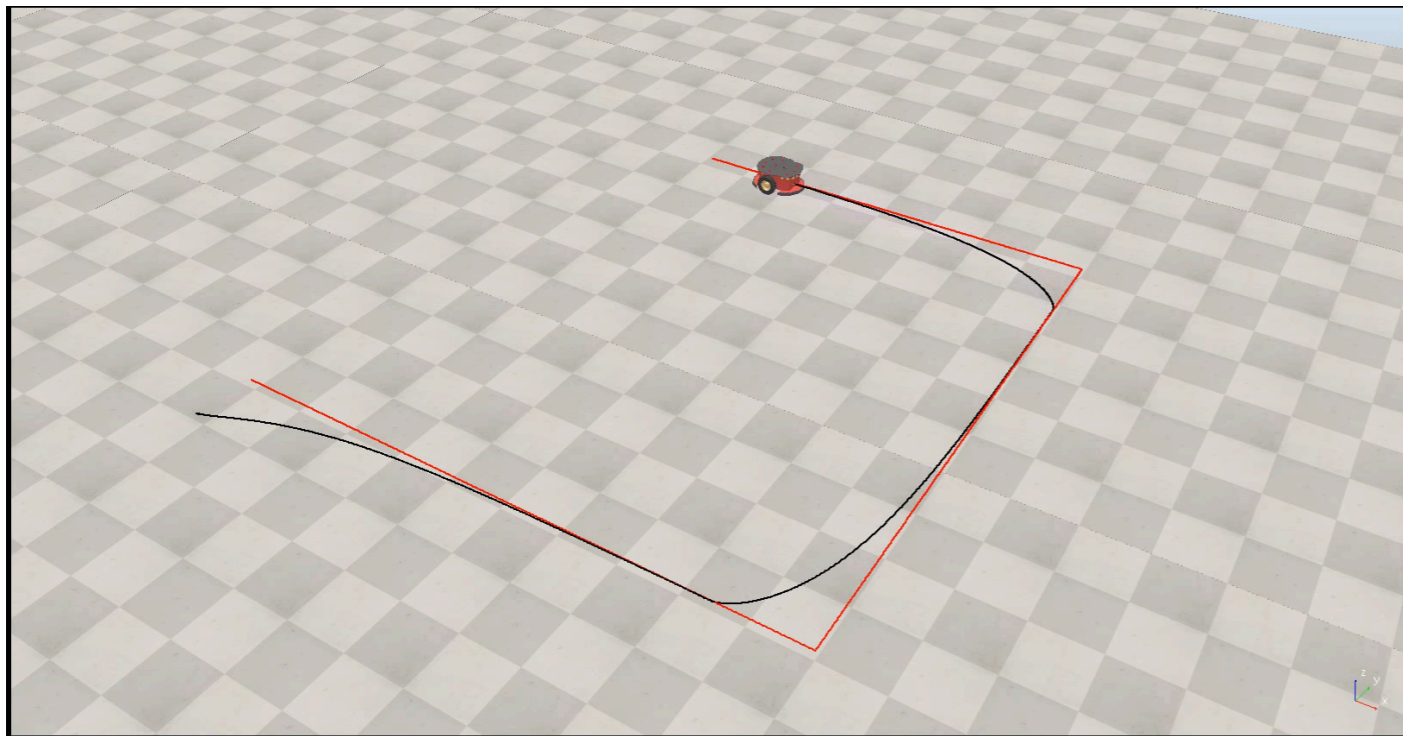


- only local stability is guaranteed
- a new transient at each corner

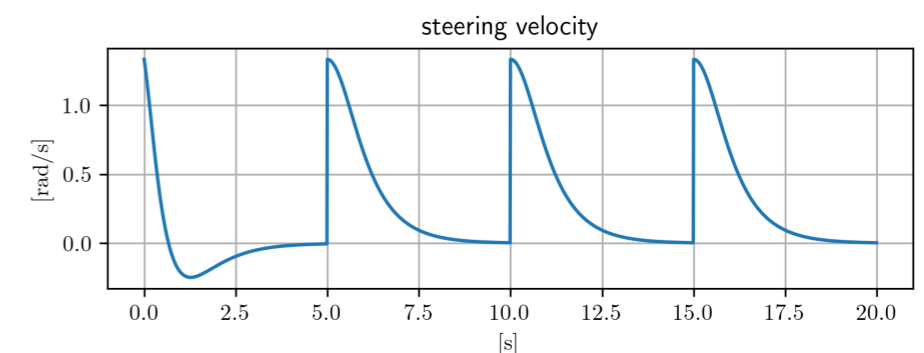
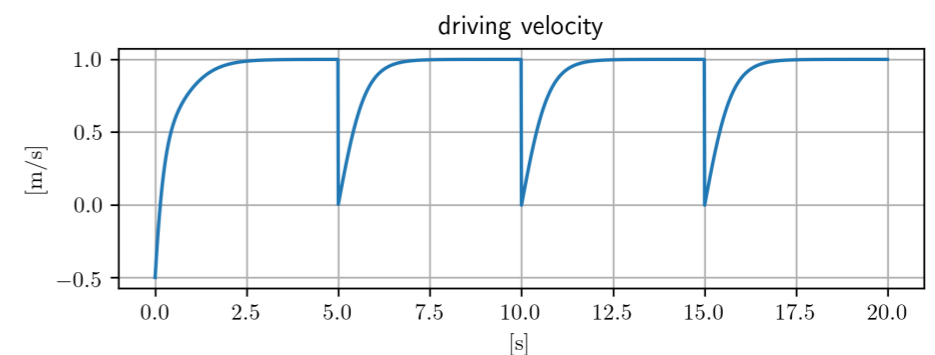
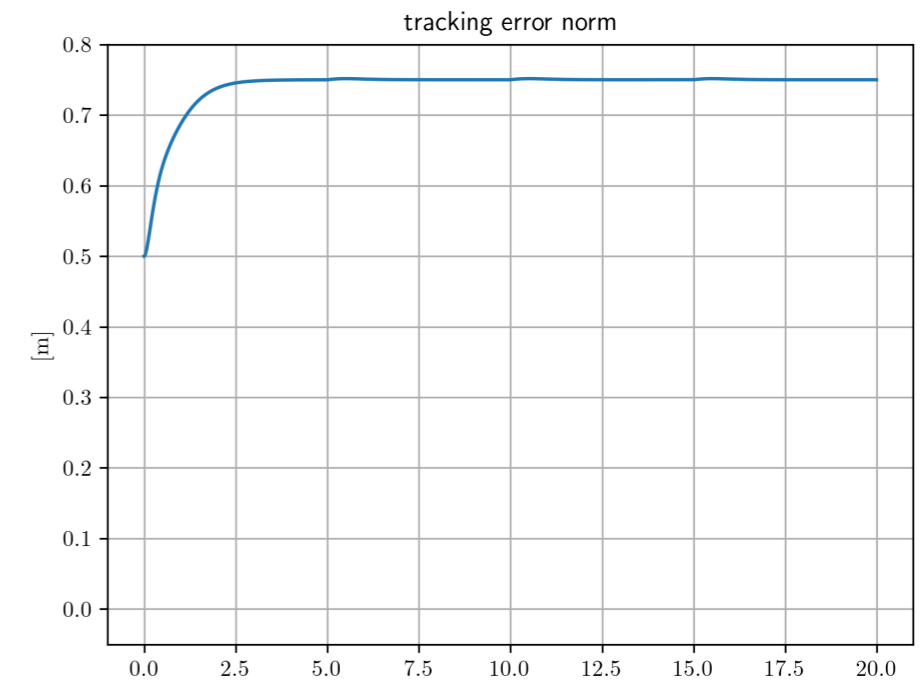


simulations

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

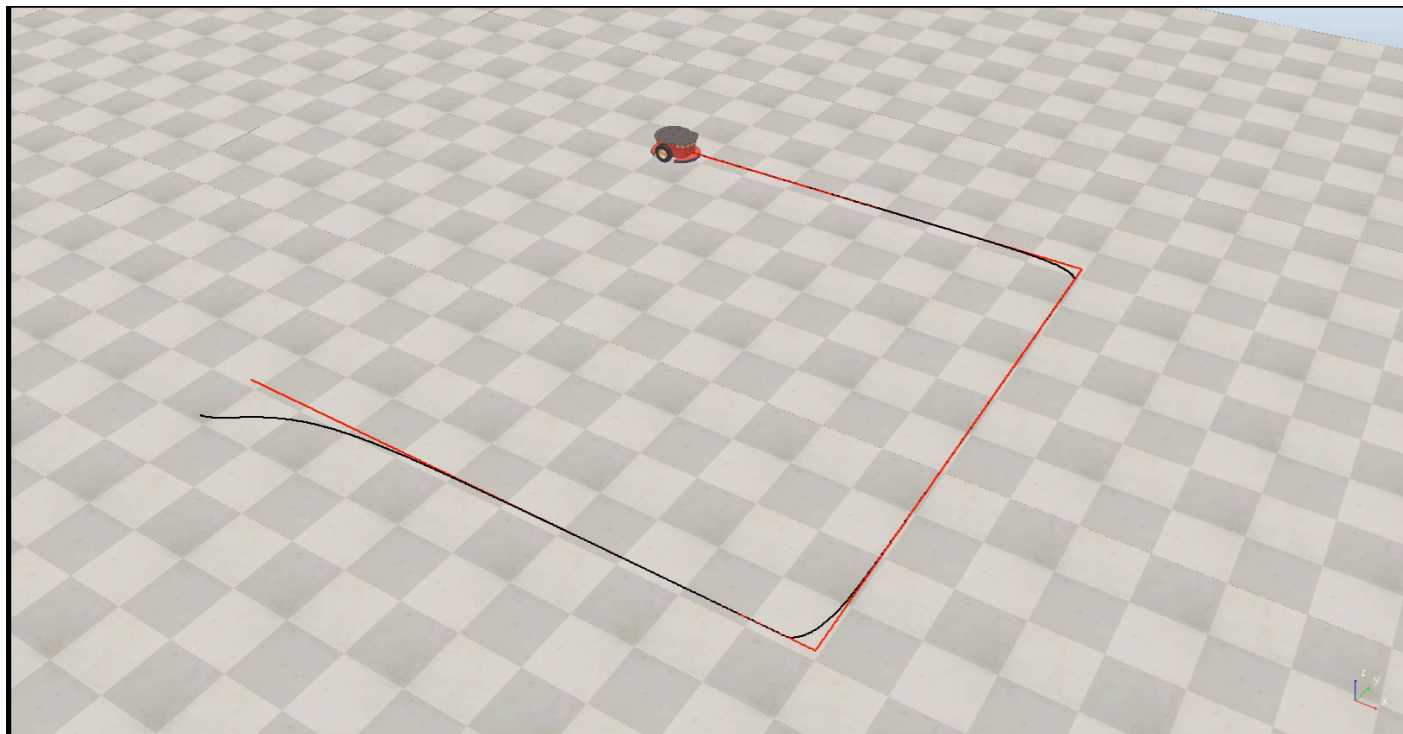


- steady-state error = b
- the displaced output provides a **lookahead** behavior

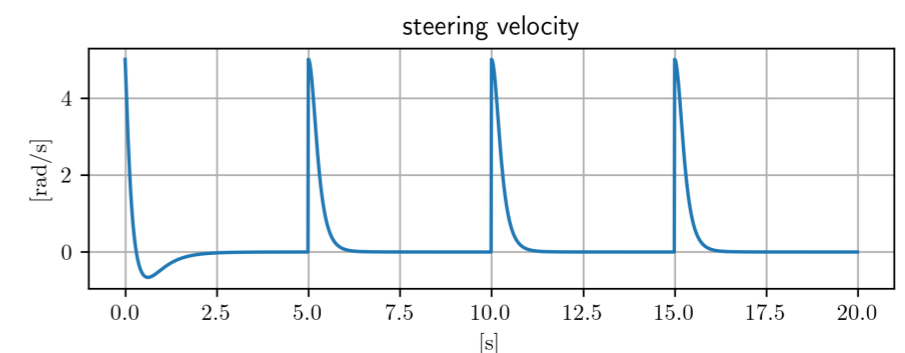
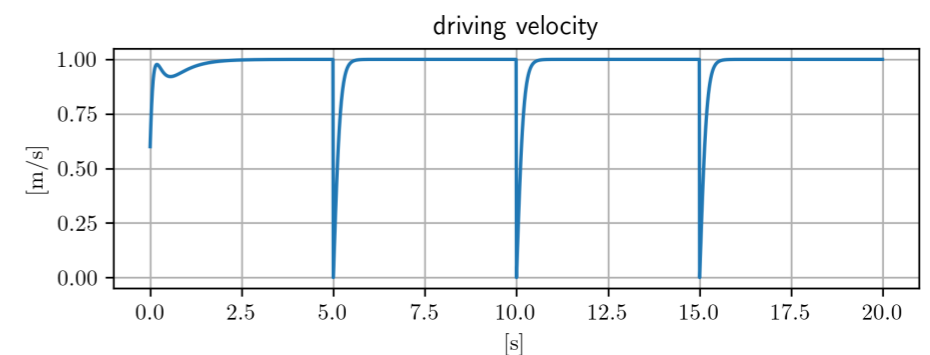
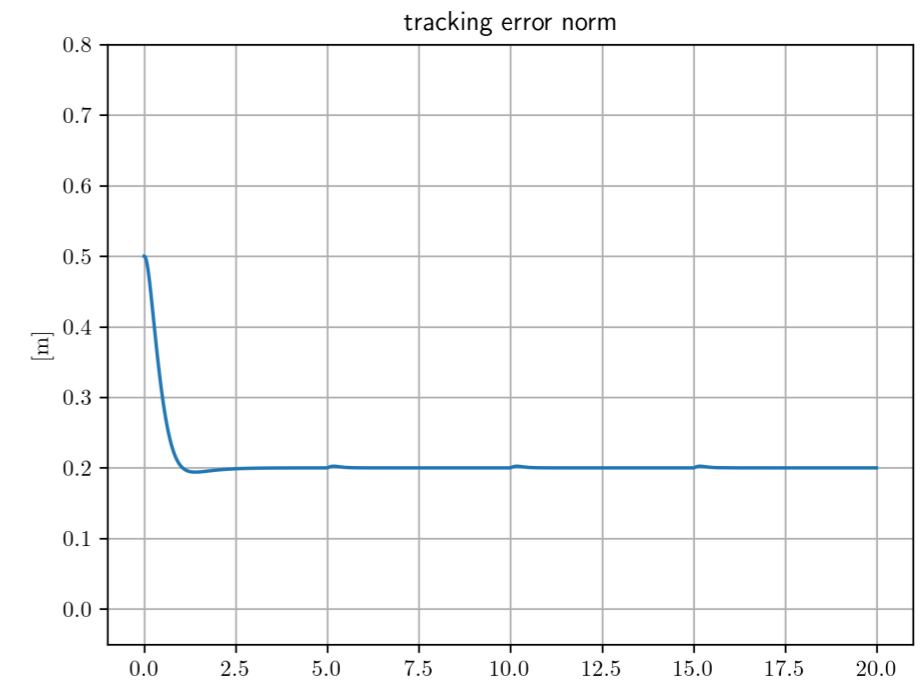


simulations

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

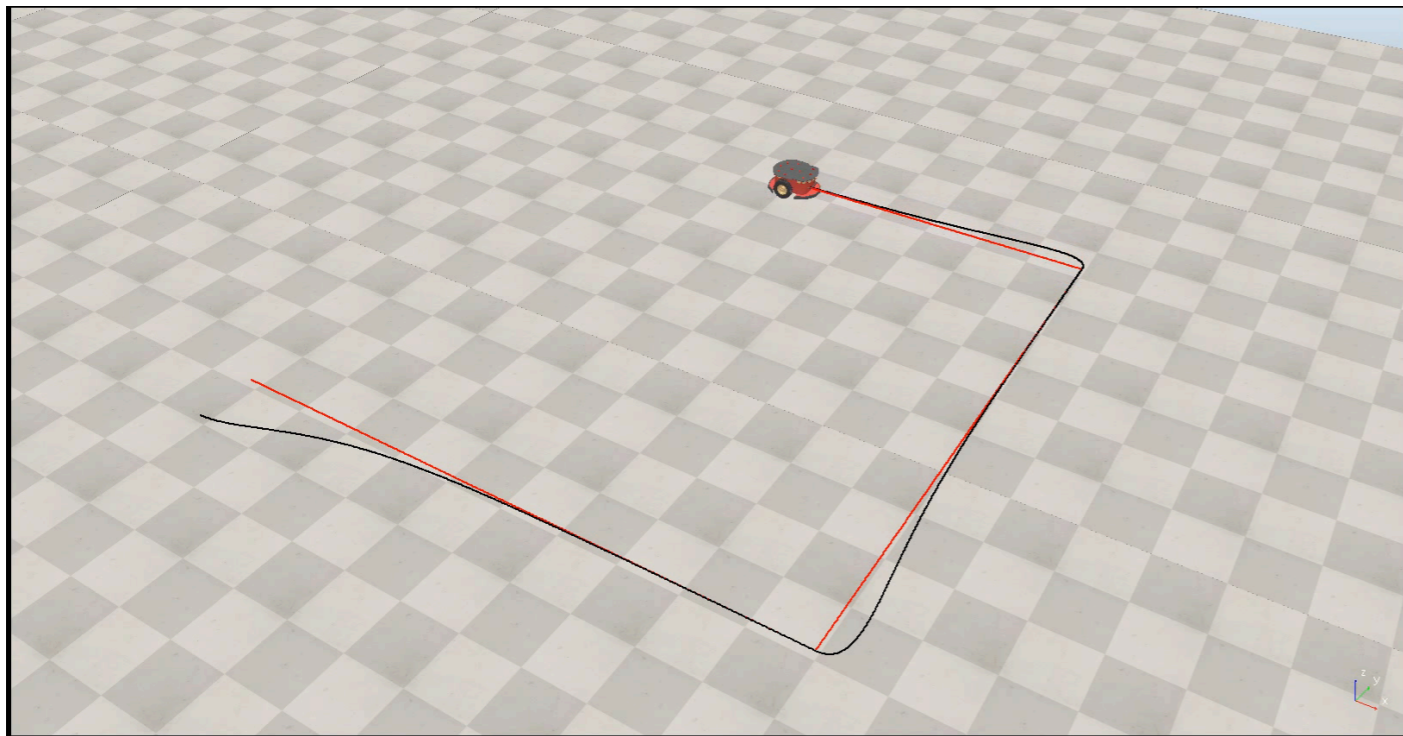


- steady-state error is now reduced but steering velocity increases

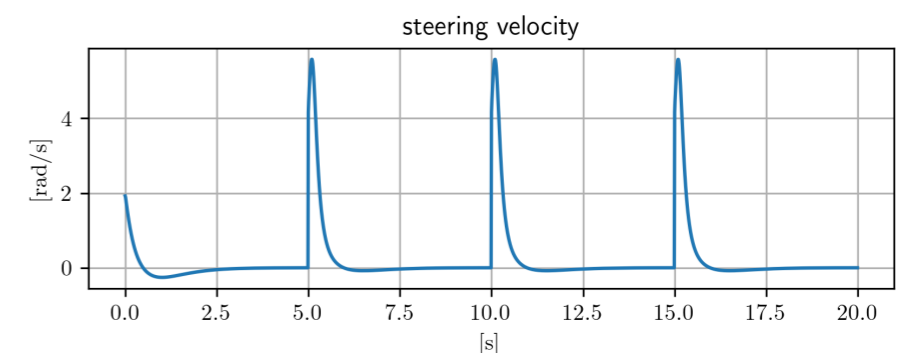
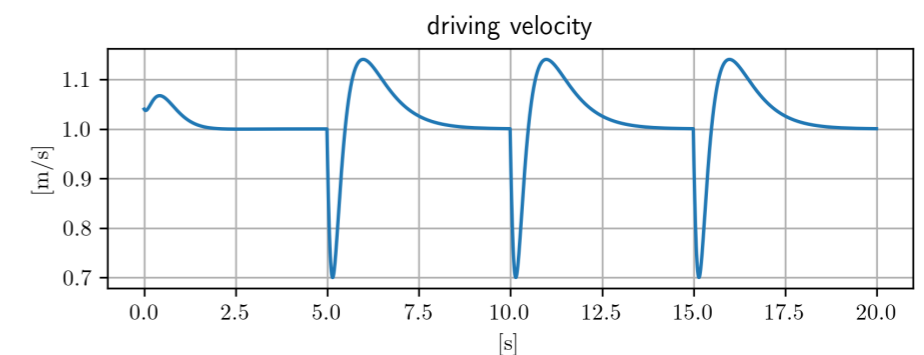
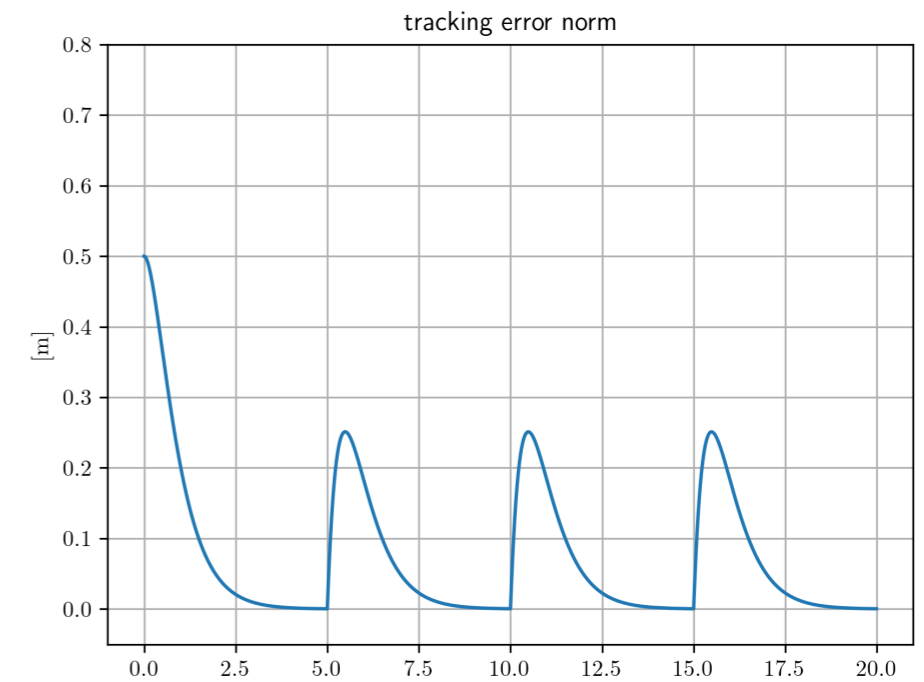


simulations

tracking a square via dynamic i/o linearization

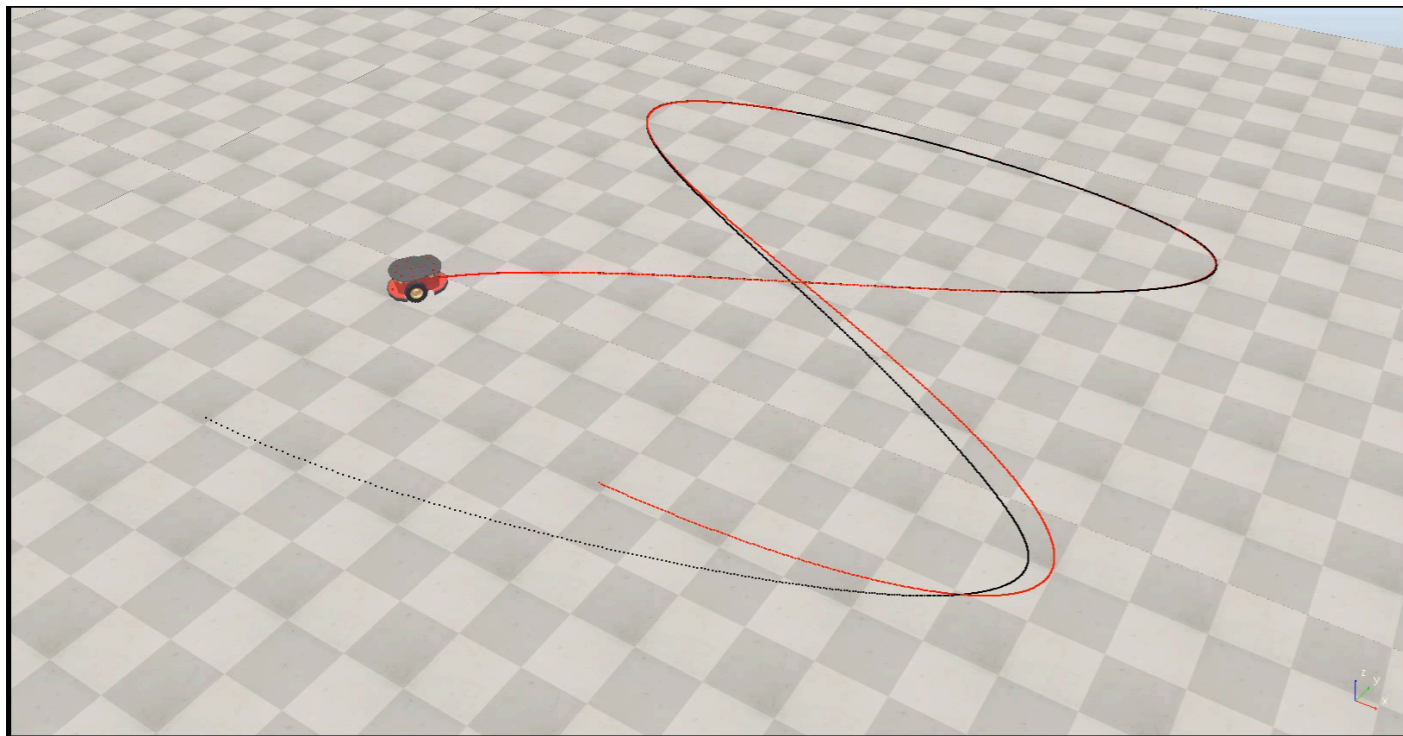


- faster transients and reasonable velocities

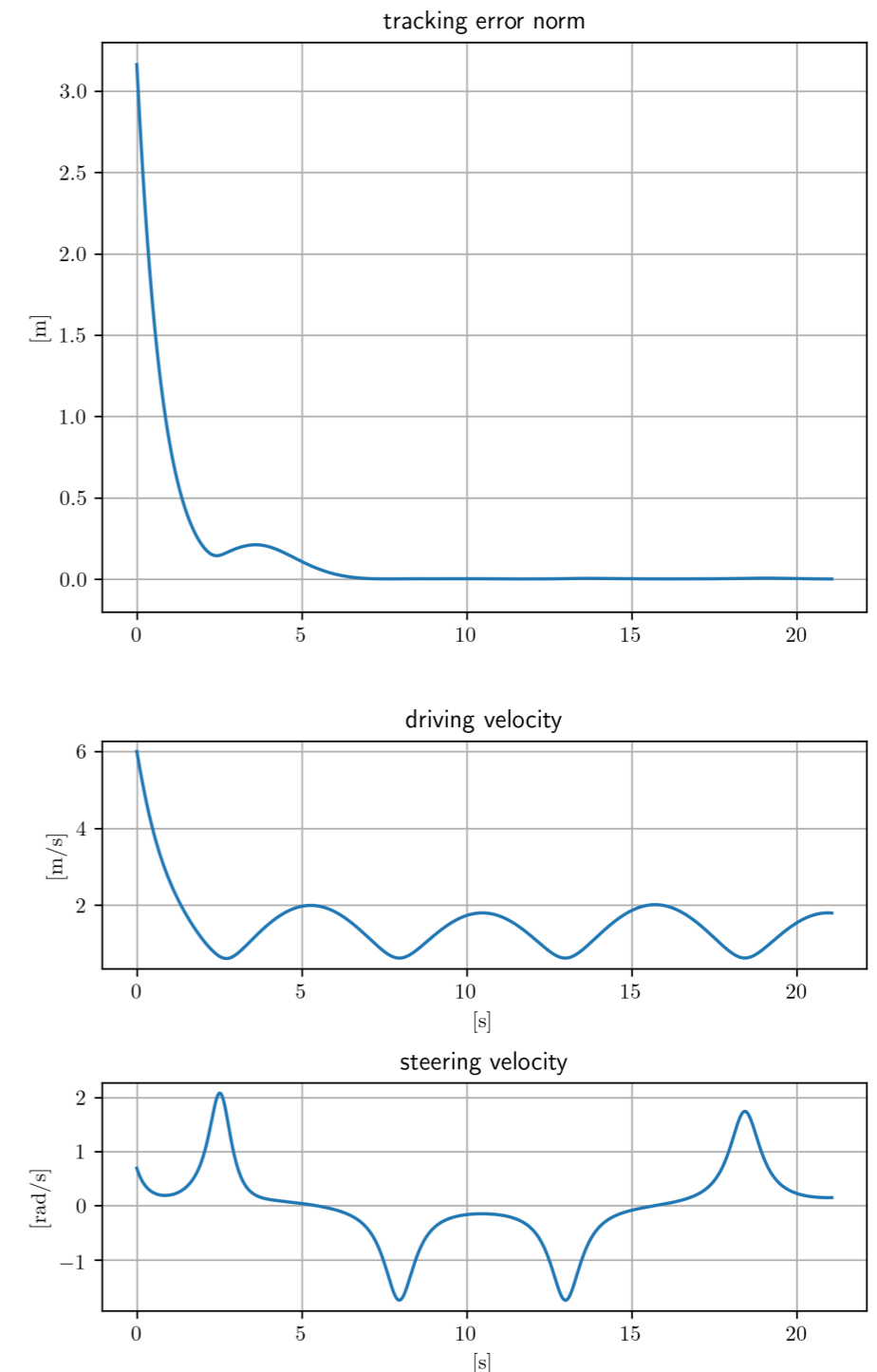


simulations

tracking a figure 8 via approximate linearization

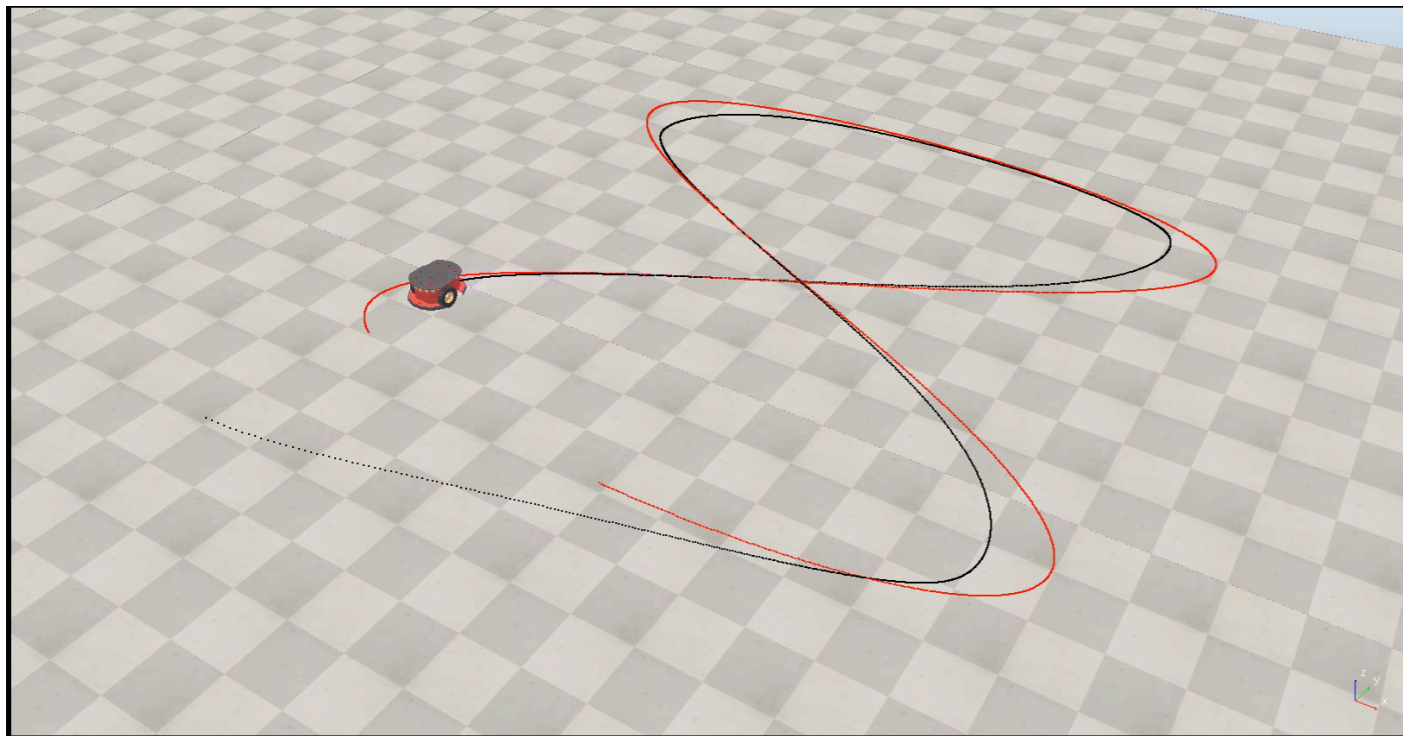


- local stability is not guaranteed, but performance is good

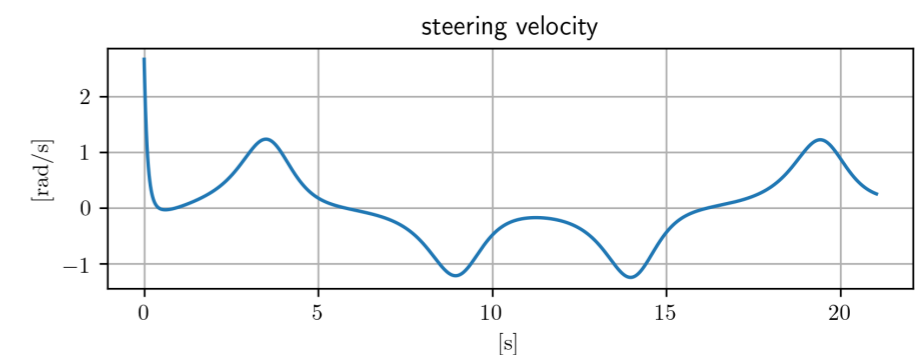
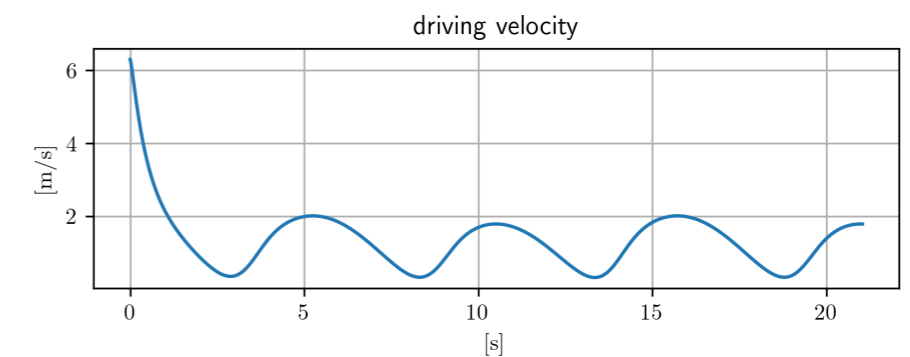
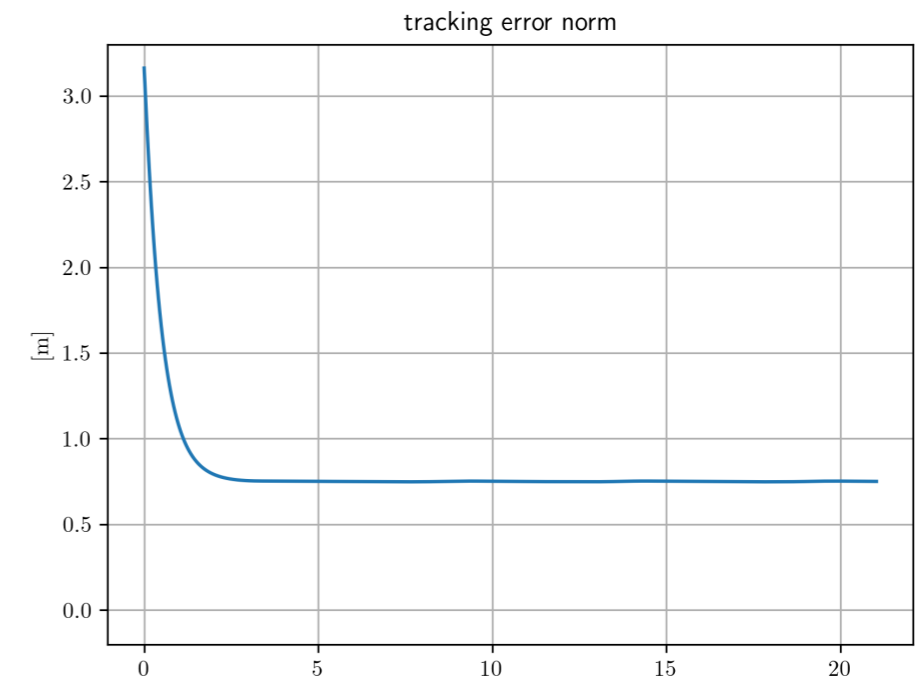


simulations

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

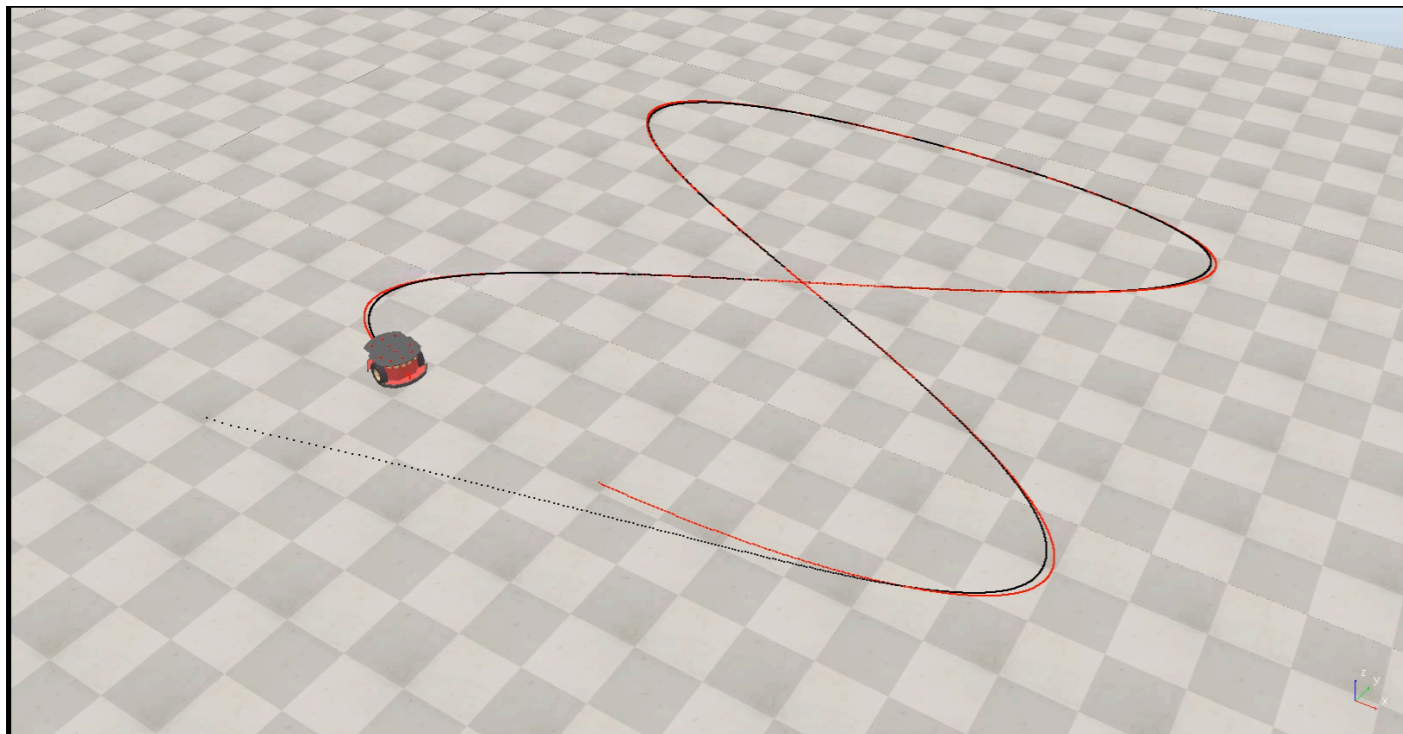


- steady-state error = b

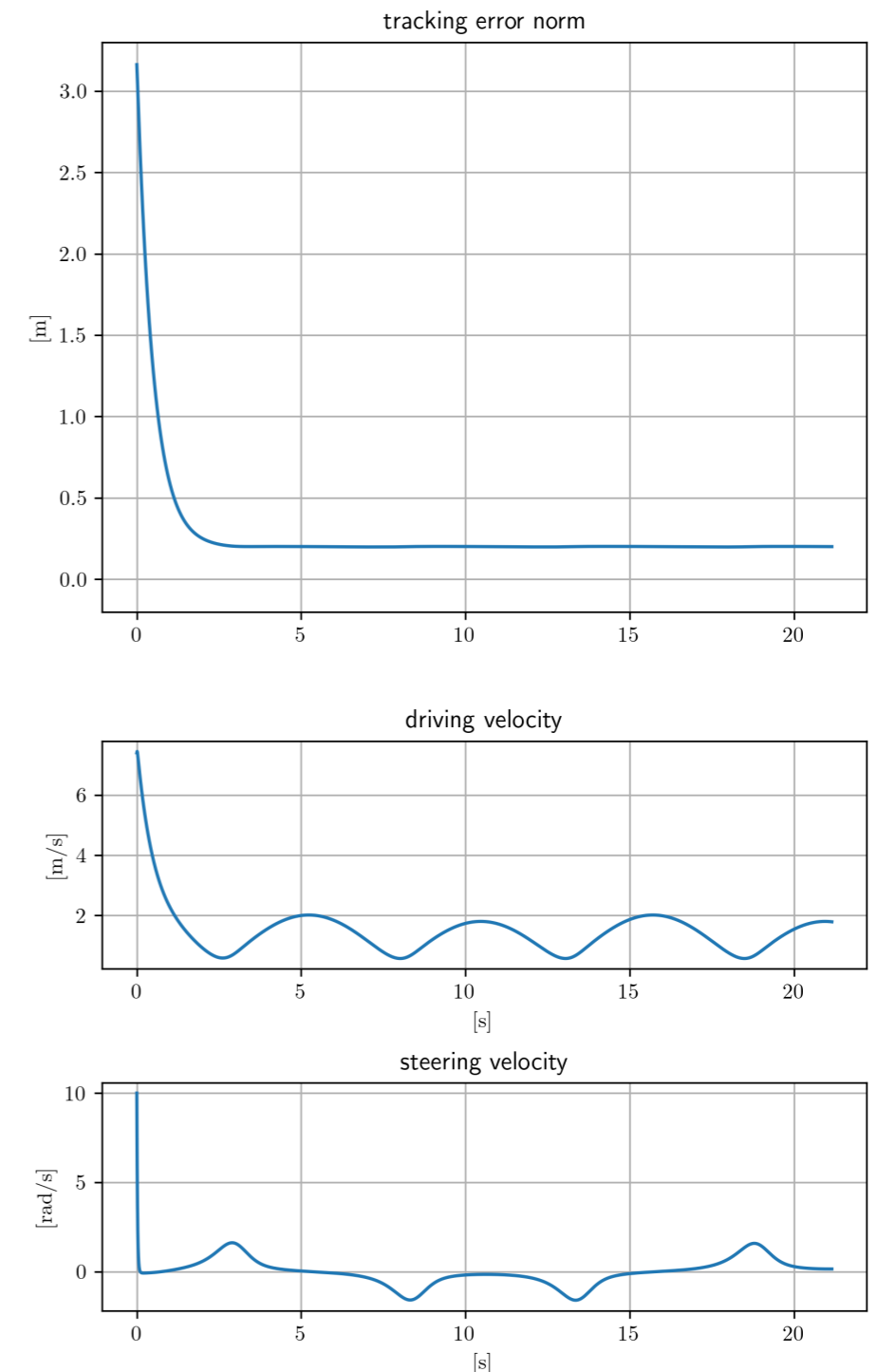


simulations

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

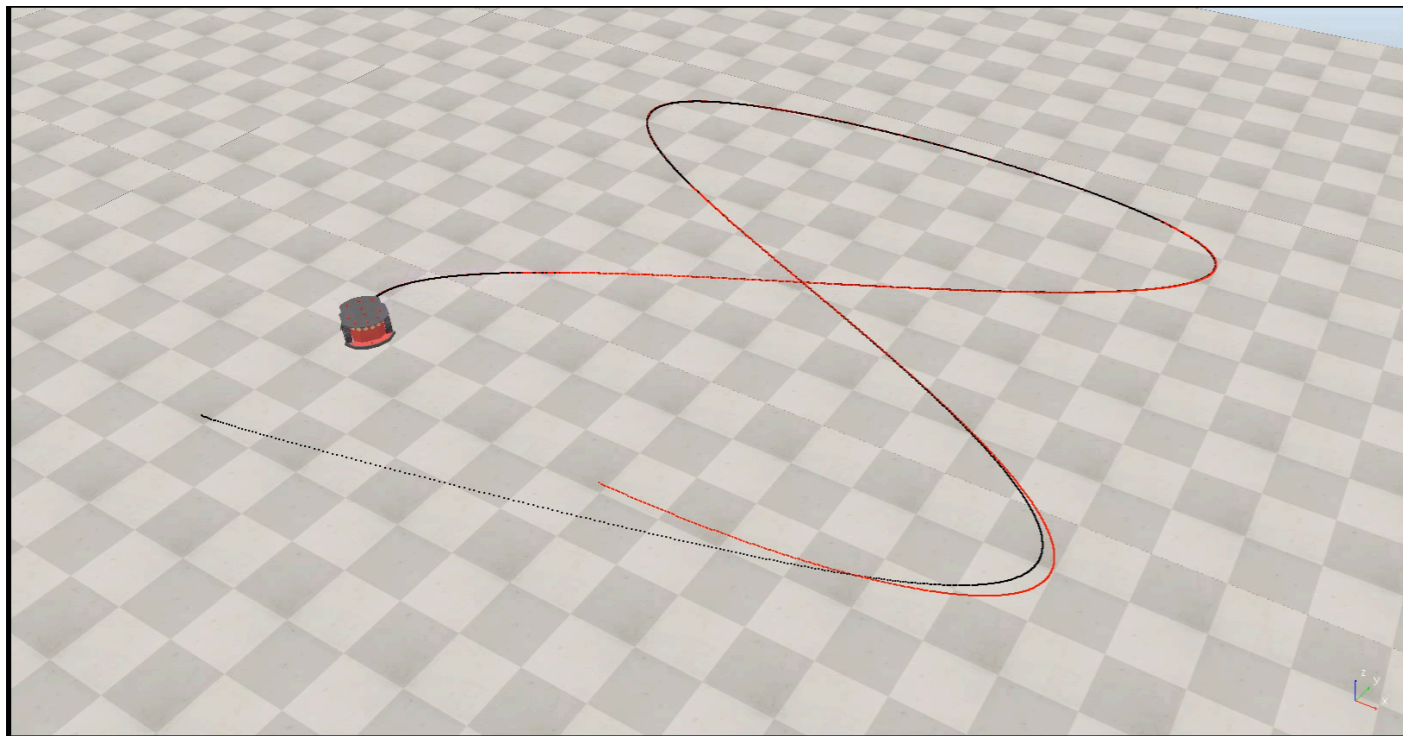


- steady-state error is now reduced but steering velocity increases



simulations

tracking a figure 8 via dynamic i/o linearization



- zero steady-state error and reasonable velocities

