Autonomous and Mobile Robotics

Prof. Giuseppe Oriolo

Wheeled Mobile Robots 4

Motion Control of WMRs: Trajectory Tracking

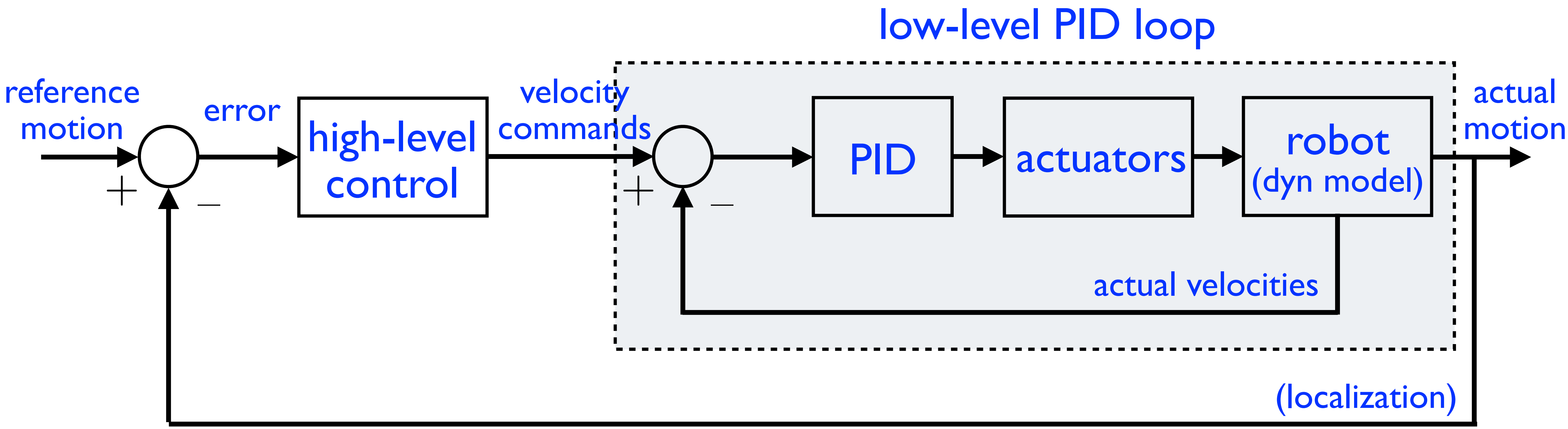
Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

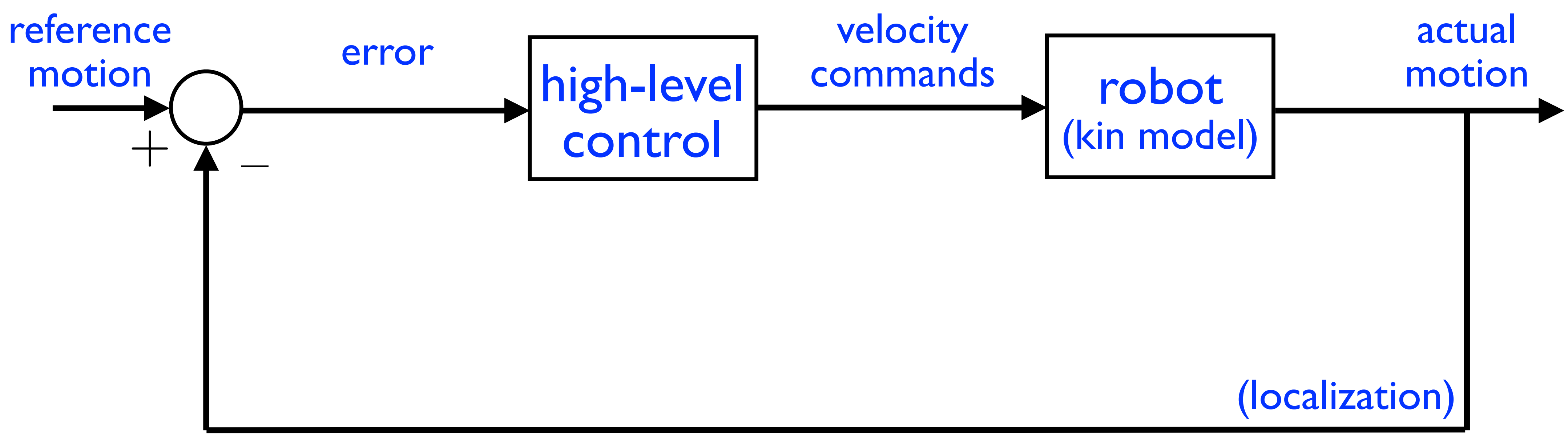
motion control

- 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
- dynamic models are generally used in robotics to compute commands at the generalized force level
- kinematic models are used to design WMR feedback laws because (1) dynamic terms can be canceled via feedback (2) wheel actuators are equipped with low-level PID loops that accept velocities as reference

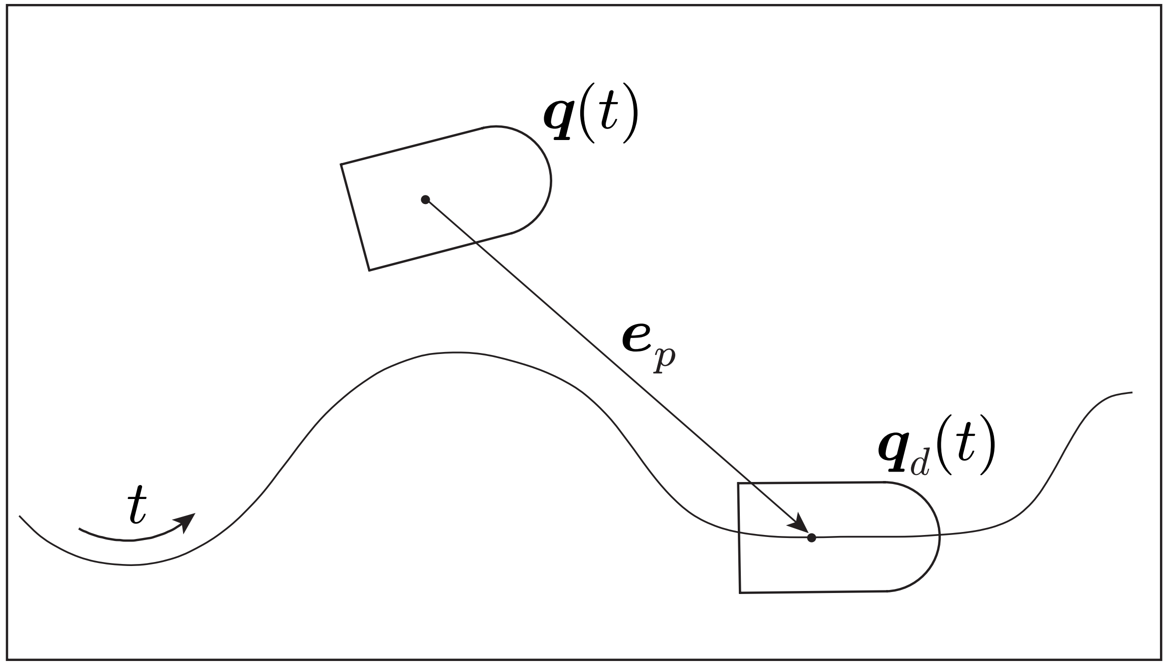
- actual control scheme



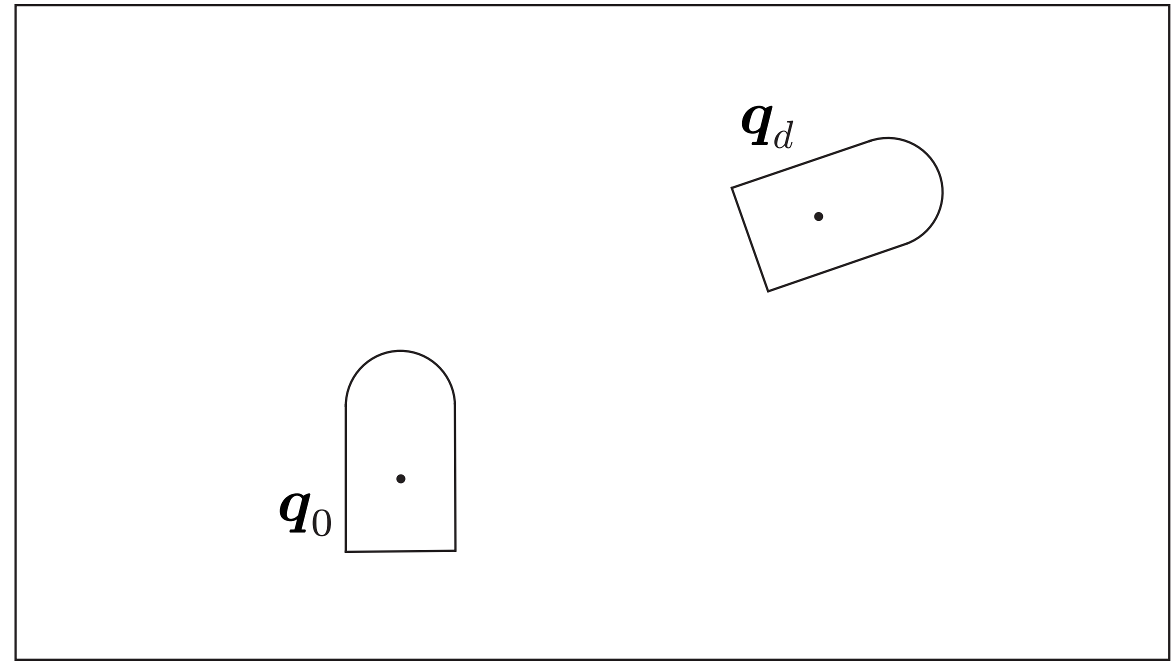
- equivalent control scheme (for design)



motion control problems



trajectory tracking
(predictable transients)



posture regulation
(no prior planning)

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

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

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

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

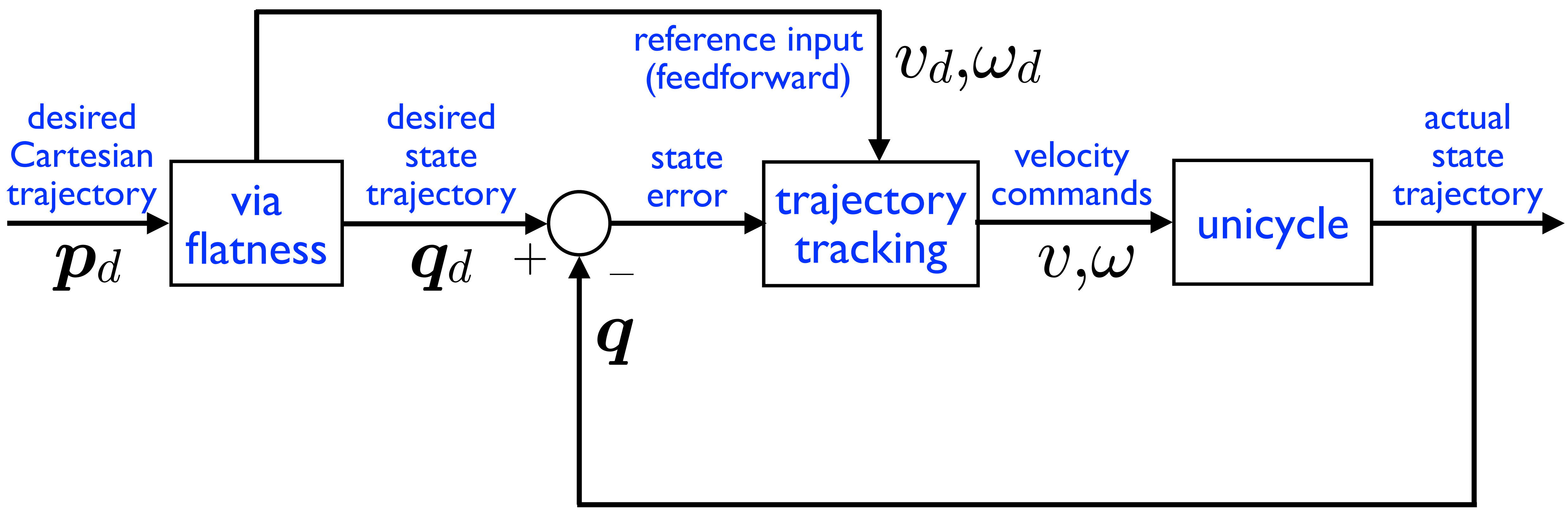
- thanks to flatness, from $(x_d(t), y_d(t))$ we can compute

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

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

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

- 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 $\boldsymbol{e}_p$ (previous figure) in a frame rotated by θ

- 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

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

use 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

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

drift term
nonlinear, time-varying

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

- 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

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

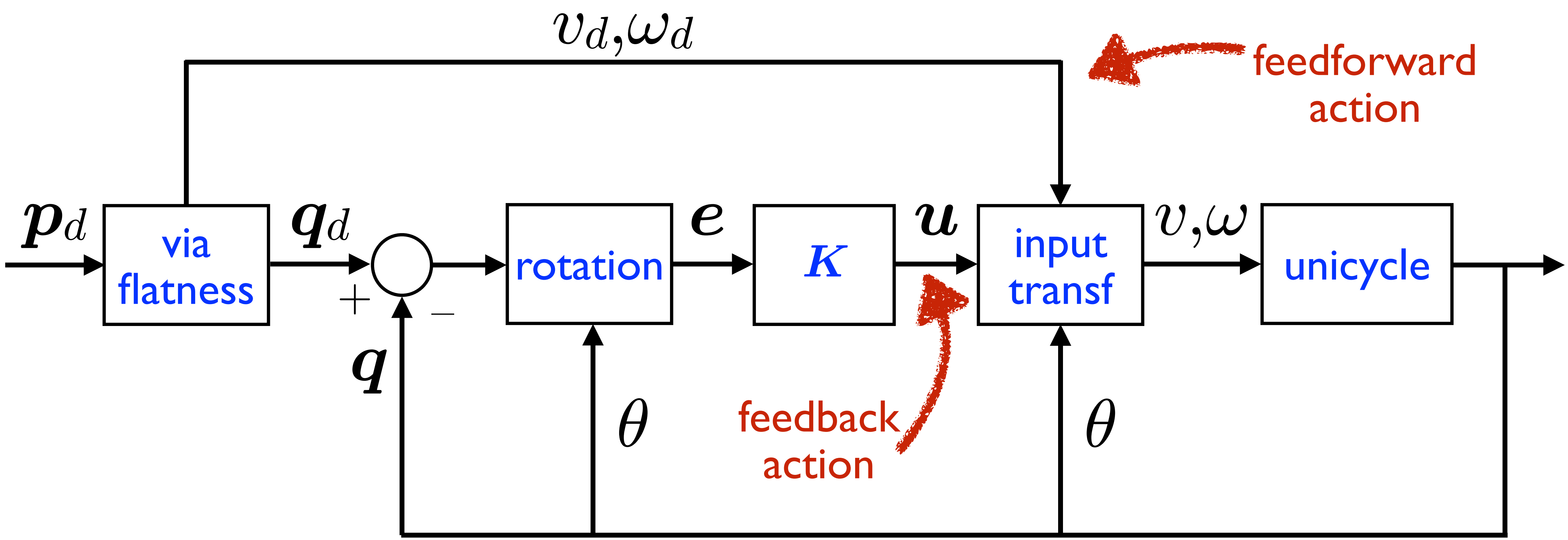
- caveat: this does not guarantee asymptotic stability, unless v_d and ω_d are constant (rectilinear and circular trajectories); even in this case, asymptotic stability of the unicycle is not global (indirect Lyapunov method)

- 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 $\boldsymbol{e} \to \boldsymbol{0}$ (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

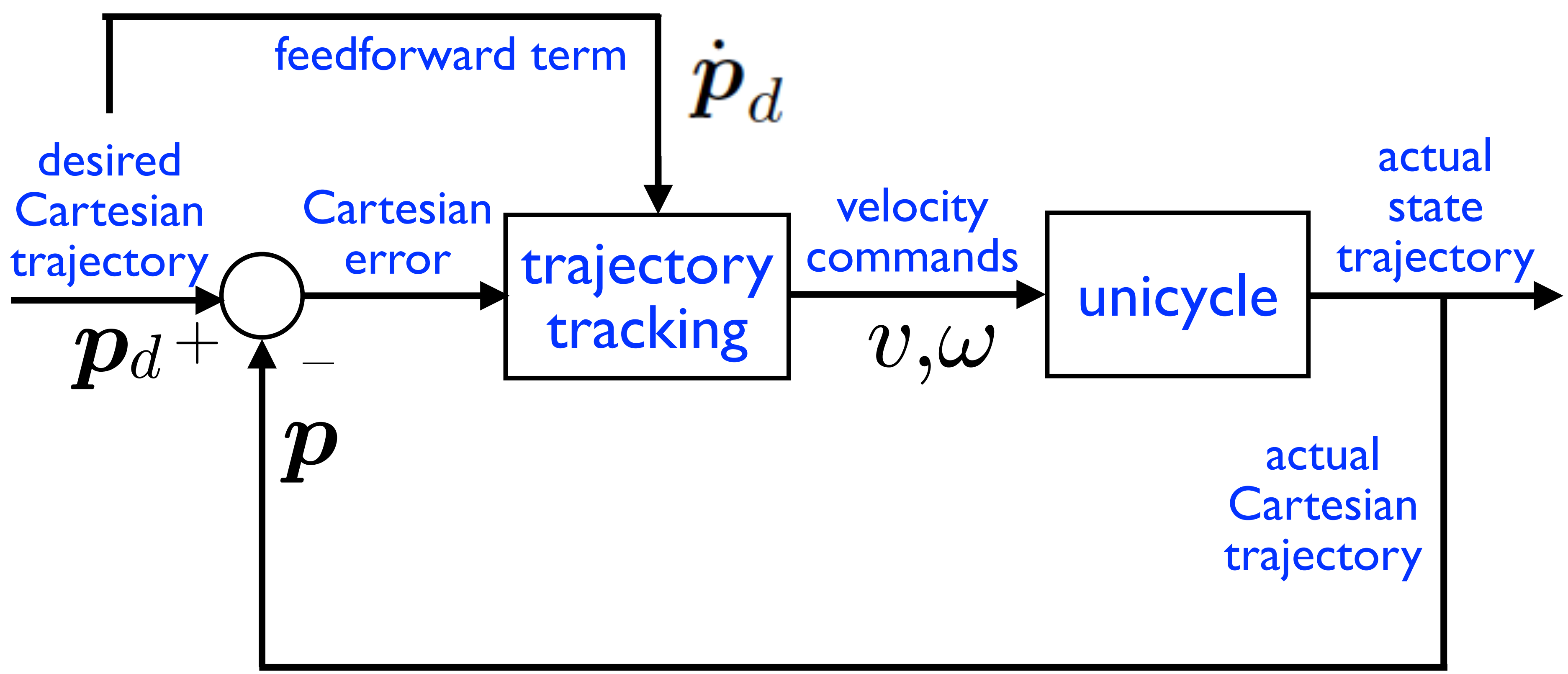


- based on state error
- needs v_d, ω_d
- needs θ also for error rotation + input transformation

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 is



via exact input/output linearization

- idea: (1) if the map between the available inputs and some derivative of the output is invertible, then (2) by inverting this map the system can be made linear

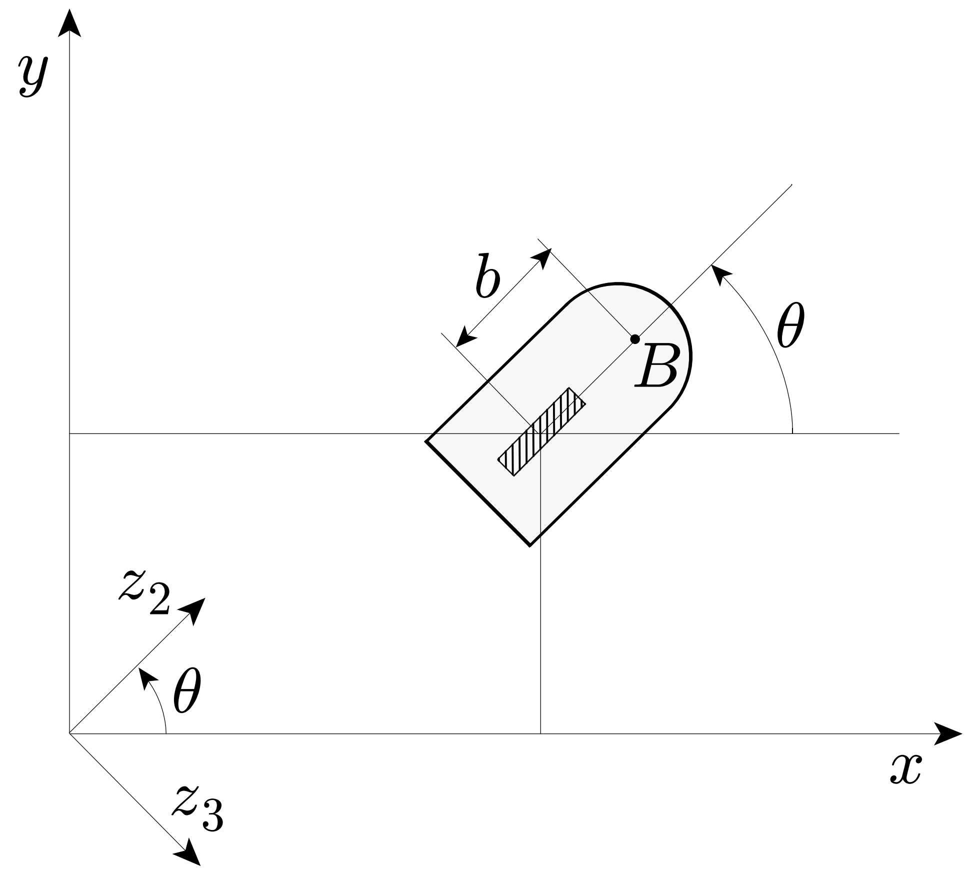
- however, for the unicycle the map between the velocity inputs and the Cartesian output is singular

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

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



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

- differentiating wrt time

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

- 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

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

- achieve global exponential convergence of y_1, y_2 to the desired trajectory letting

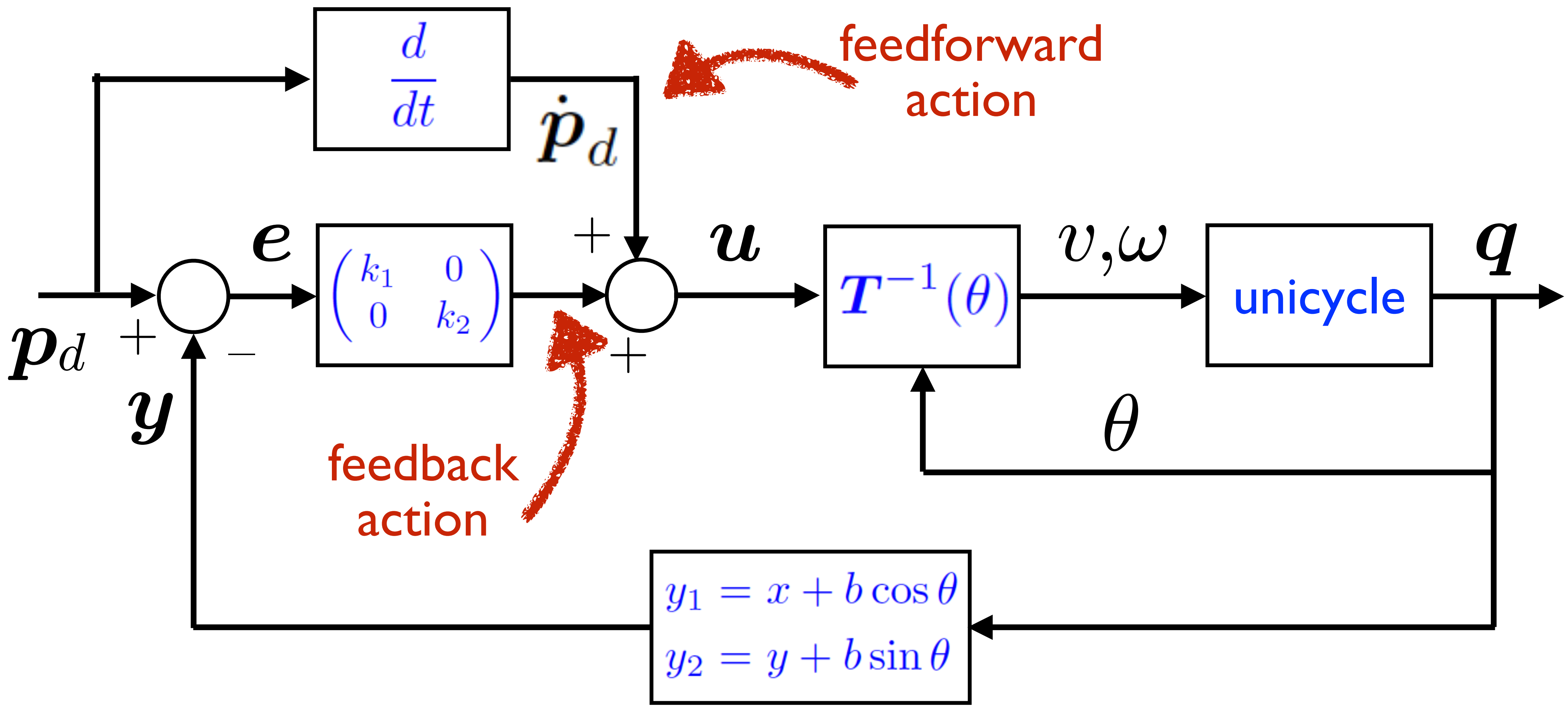
$$u_1 = \dot{y}_{1d} + k_1(y_{1d} - y_1)$$
$$u_2 = \dot{y}_{2d} + k_2(y_{2d} - y_2)$$

  with $k_1, k_2 > 0$

- θ is not controlled with this scheme, which is based on output error feedback (compare with the previous)

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

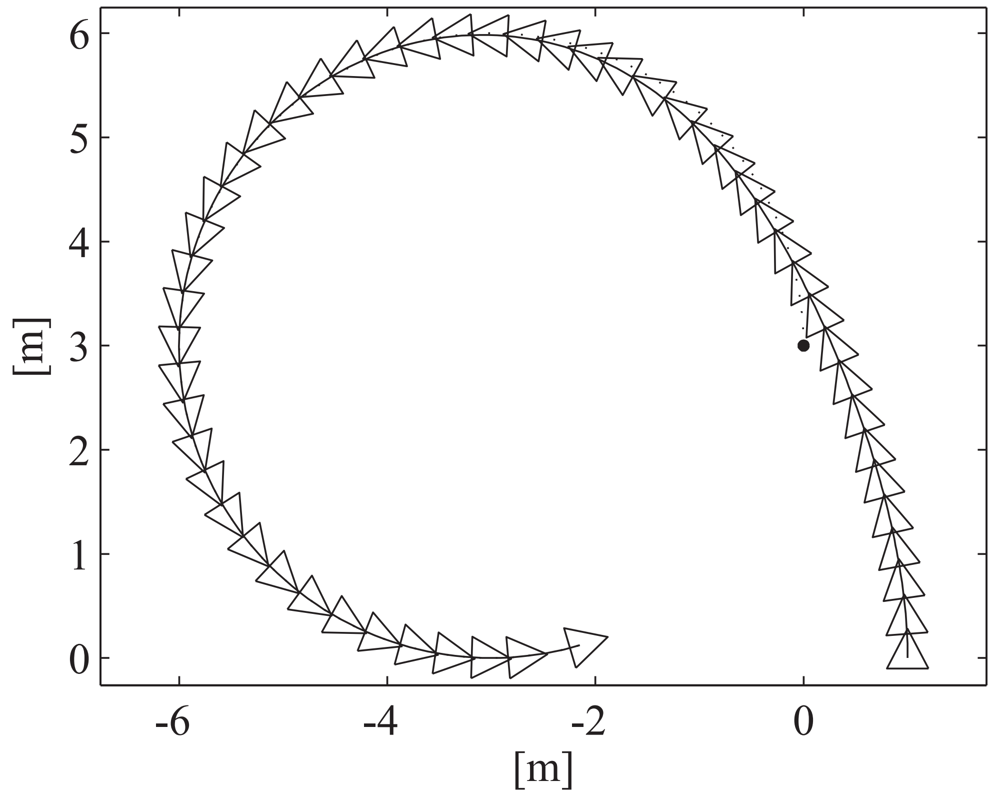
- the final block scheme for trajectory tracking via output error feedback + input-output linearization is

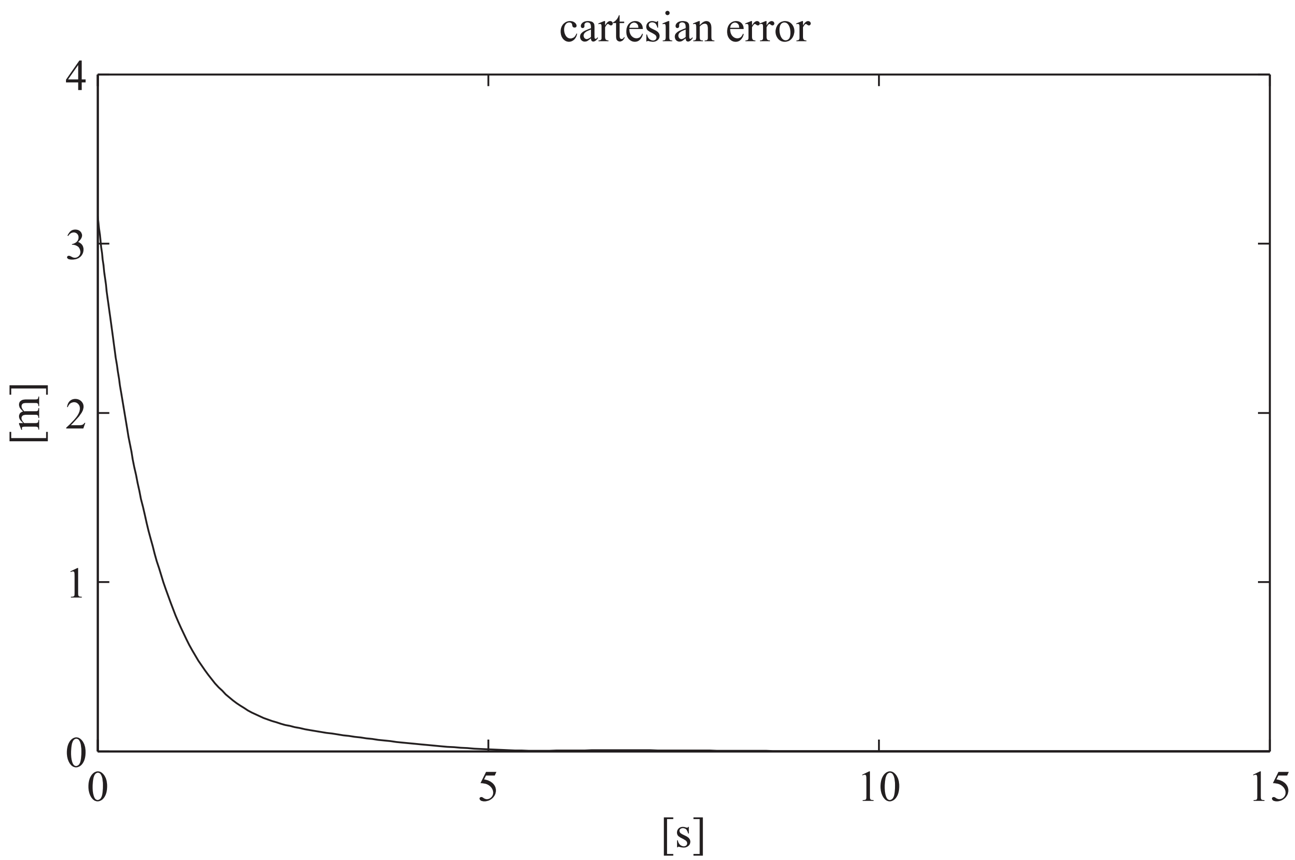


- based on output error
- needs $\dot{\boldsymbol{p}}_d$
- needs x,y,θ for output reconstruction and θ also for input transformation

simulations

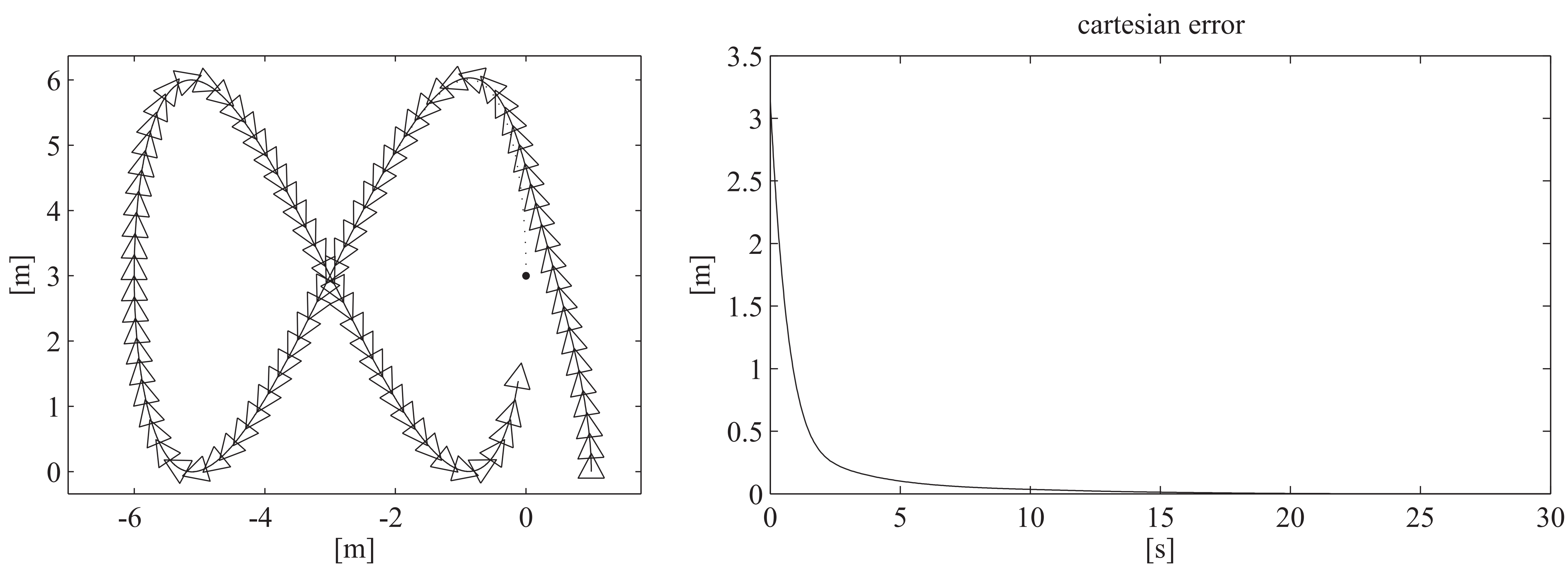
tracking a circle via approximate linearization





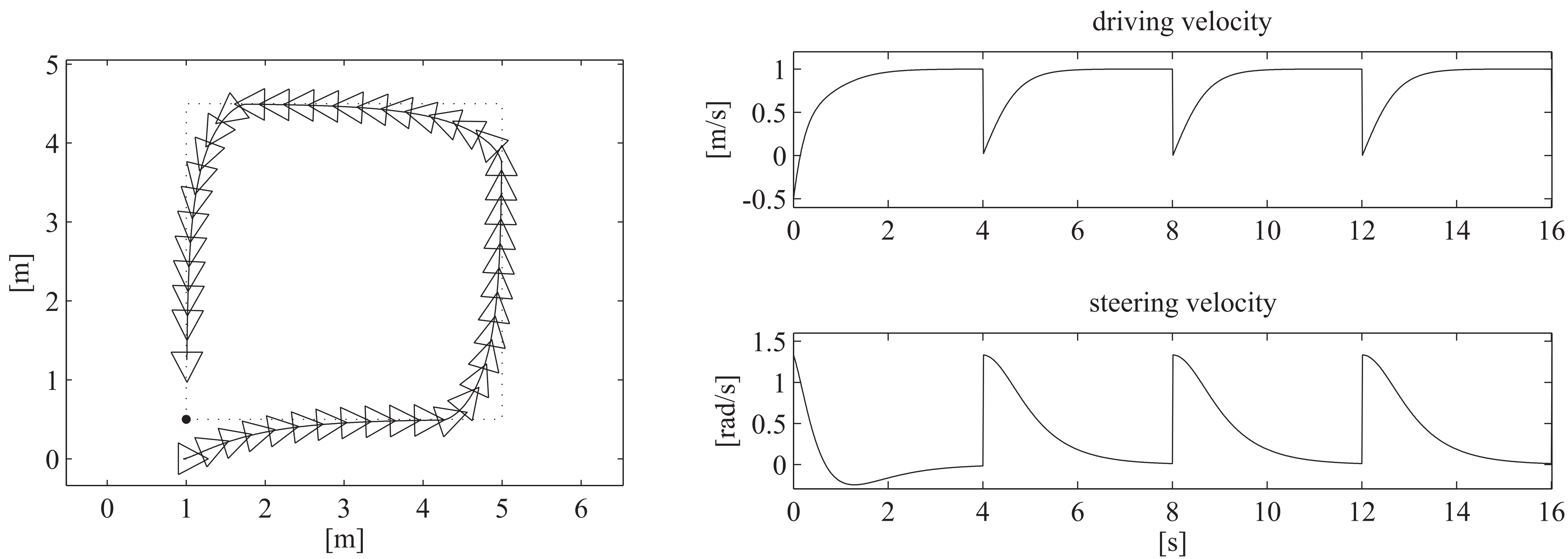
simulations

tracking an 8-figure via nonlinear feedback



simulations

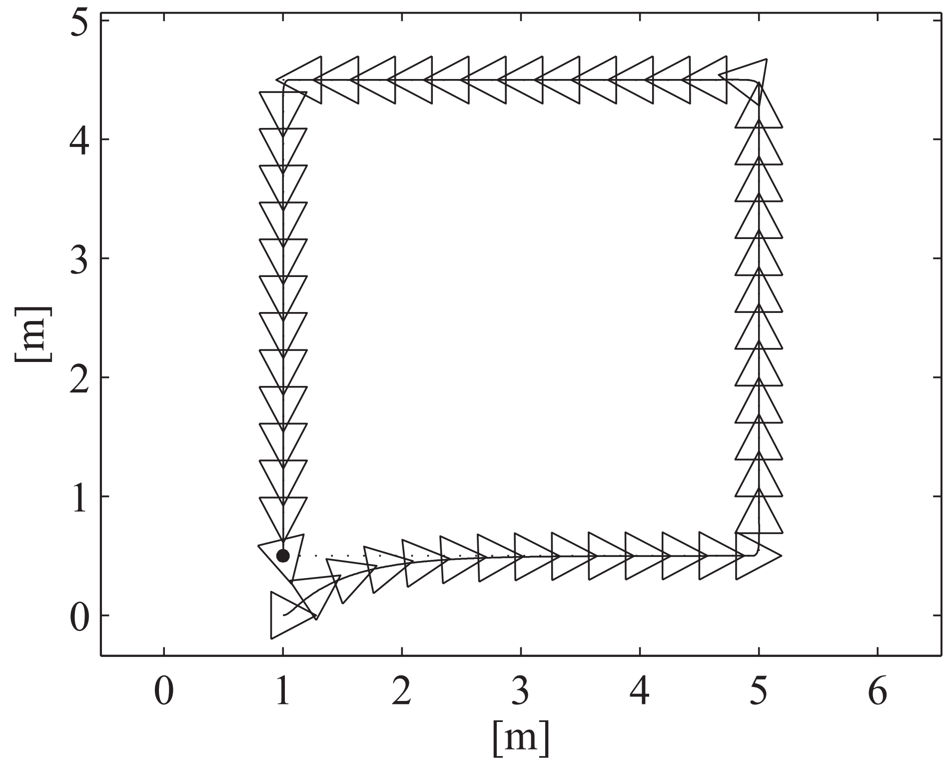
tracking a square via i/o linearization

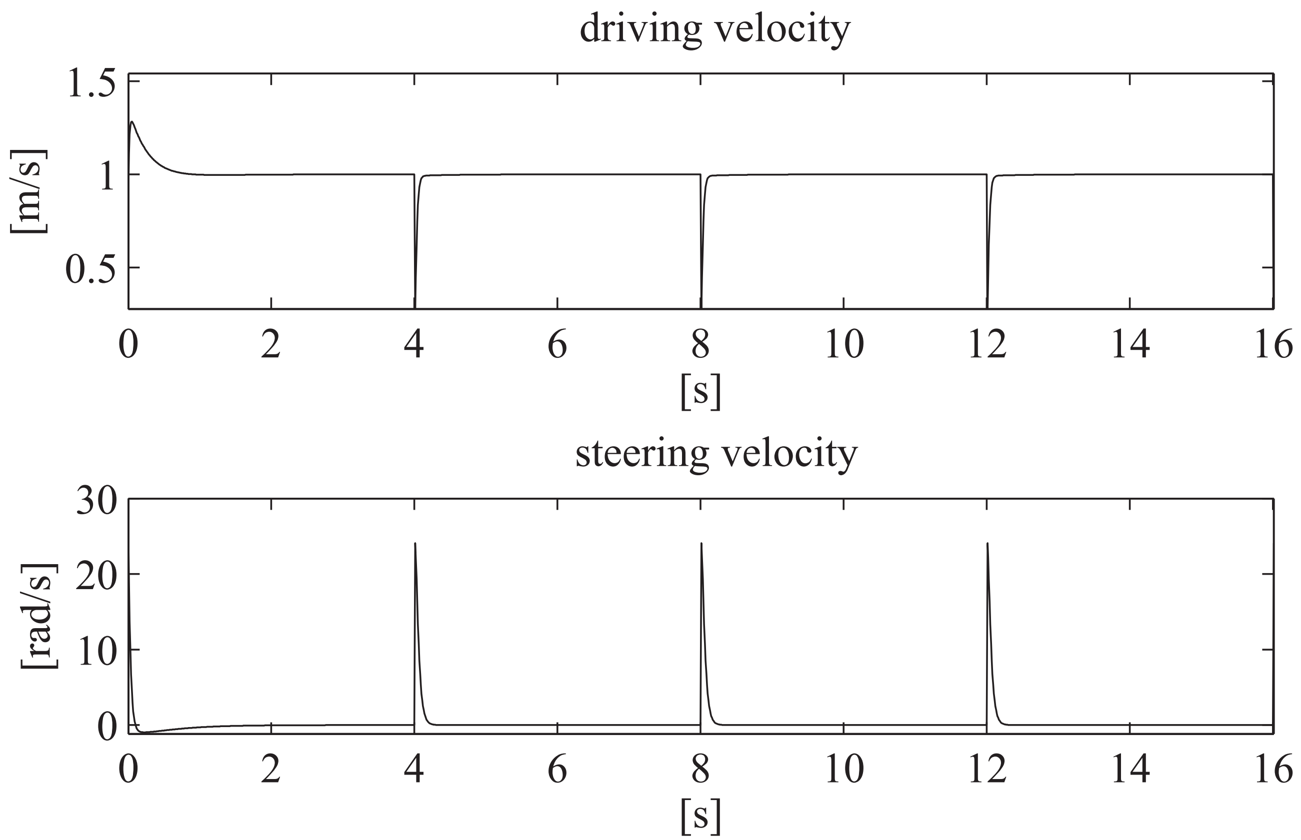


$b=0.75 \Rightarrow$ the unicycle rounds the corners

simulations

tracking a square via i/o linearization





$b = 0.2 \Rightarrow$ accurate tracking but velocities increase