Ground vehicle motion control

TSFS12: Autonomous Vehicles – planning, control, and learning systems

Lecture 6: Erik Frisk <erik.frisk@liu.se>

LINKÖPINGS UNIVERSITET

Scope of this lecture

- Introduce control structure and architecture for path and trajectory following
- Basic approaches to motion control
- Basic notion of trajectories, paths and curvature
- Two methods for path following controllers
  - Pure-pursuit
  - Stabilization by state-feedback control
- Comment on properties of linear and non-linear controllers

Reading instructions

- Main texts
  - Paden, Brian, et al. "*A survey of motion planning and control techniques for self-driving urban vehicles*." IEEE Transactions on intelligent vehicles 1.1 (2016): 33-55.

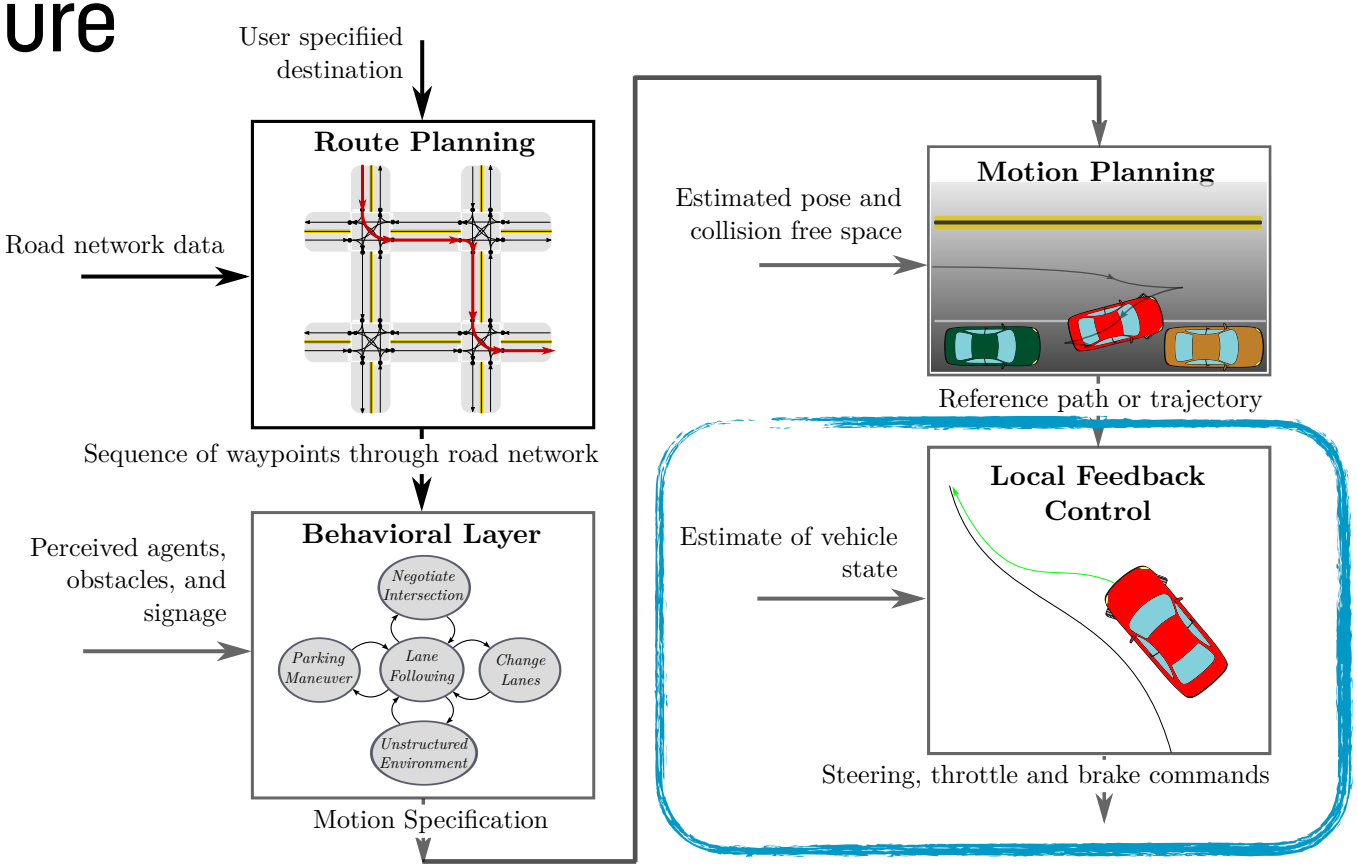
    A good but slightly advanced text. Therefore many details are included in these lecture notes
  - Coulter, R. Craig. "*Implementation of the Pure Pursuit Path Tracking Algorithm*" (1992).
    [But don't look at the figure in this paper, it is badly scaled and difficult to understand; use figures in these slides instead]

Reading material, cont'

- Want to dig a little deeper? Here's some extra reading…
  - Siciliano, B., and Oussama K., eds. "*Springer handbook of robotics*". Springer, 2016. Part E, Chapters 49 (wheeled robots), 51 (underwater), and 52 (aerial).
  - Werling, M., Gröll, L., and Bretthauer, G.. "*Invariant trajectory tracking with a full-size autonomous road vehicle*". IEEE Transactions on Robotics 26.4 (2010): 758-765.

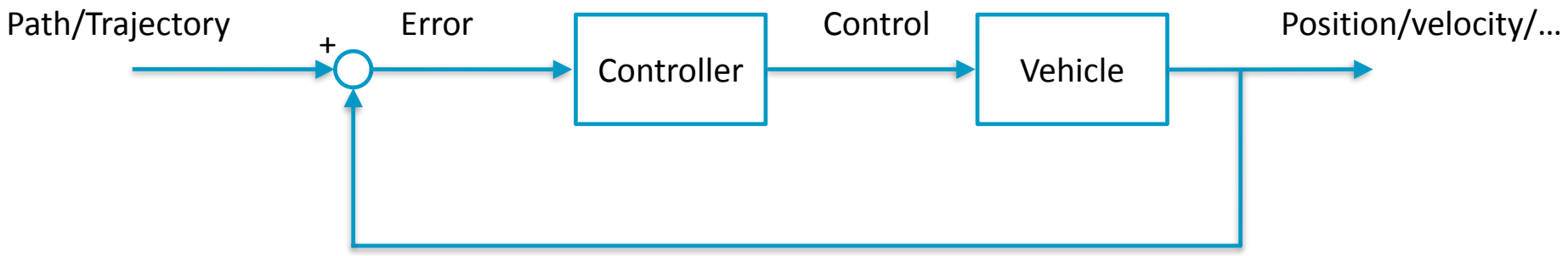
    An advanced, control oriented, text on nonlinear trajectory stabilizing controllers.

Control architecture



Control architecture from Paden et al., IEEE Trans. Intell. Vehicles, 1:1, 2016.

Basic motion controller



- Paths/trajectories from a high-level planner
- Obstacle detections
- What happens if you get away from intended path; how to get back on track
- Receding horizon control, MPC, and replanning

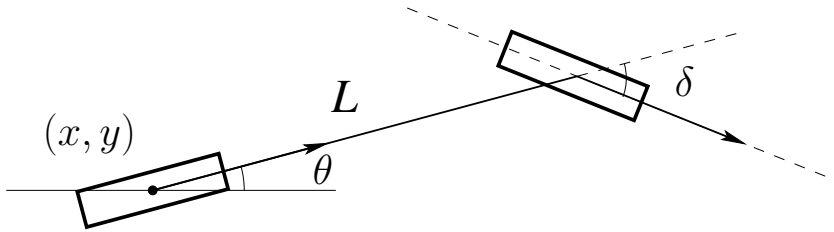
Approaches

Controller		Model	Stability	Time Complexity	Comments/Assumptions
★ Pure Pursuit	(V-A1)	Kinematic	LES* to ref. path	$O(n)^*$	No path curvature
Rear wheel based feedback	(V-A2)	Kinematic	LES* to ref. path	$O(n)^*$	$C^2(\mathbb{R}^n)$ ref. paths
Front wheel based feedback	(V-A3)	Kinematic	LES* to ref. path	$O(n)^*$	$C^1(\mathbb{R}^n)$ ref. paths; Forward driving only
Feedback linearization	(V-B2)	Steering rate controlled kinematic	LES* to ref. traj.	$O(1)$	$C^1(\mathbb{R}^n)$ ref. traj.; Forward driving only
≈★ Control Lyapunov design	(V-B1)	Kinematic	LES* to ref. traj.	O(1)	Stable for constant path curvature and velocity
Linear MPC	(V-C)	$C^1(\mathbb{R}^n \times \mathbb{R}^m)$ model$^\sharp$	LES* to ref. or path	$O\left(\sqrt{N}\ln\left(\frac{N}{\varepsilon}\right)\right)^\dagger$	Stability depends on horizon length
Nonlinear MPC	(V-C)	$C^1(\mathbb{R}^n \times \mathbb{R}^m)$ model$^\sharp$	Not guaranteed	$O(\frac{1}{\varepsilon})^\ddagger$	Works well in practice

From Paden et al., IEEE Trans. Intell. Vehicles, 1:1, 2016.

Vehicle model

- We'll use the kinematic single-track vehicle model in the lecture
- More advanced vehicle dynamics models
  - Tires and friction
  - Suspension
  - Load transfer
  - ...
- TSFS02 - Vehicle Dynamics and Control
- Techniques still useful!

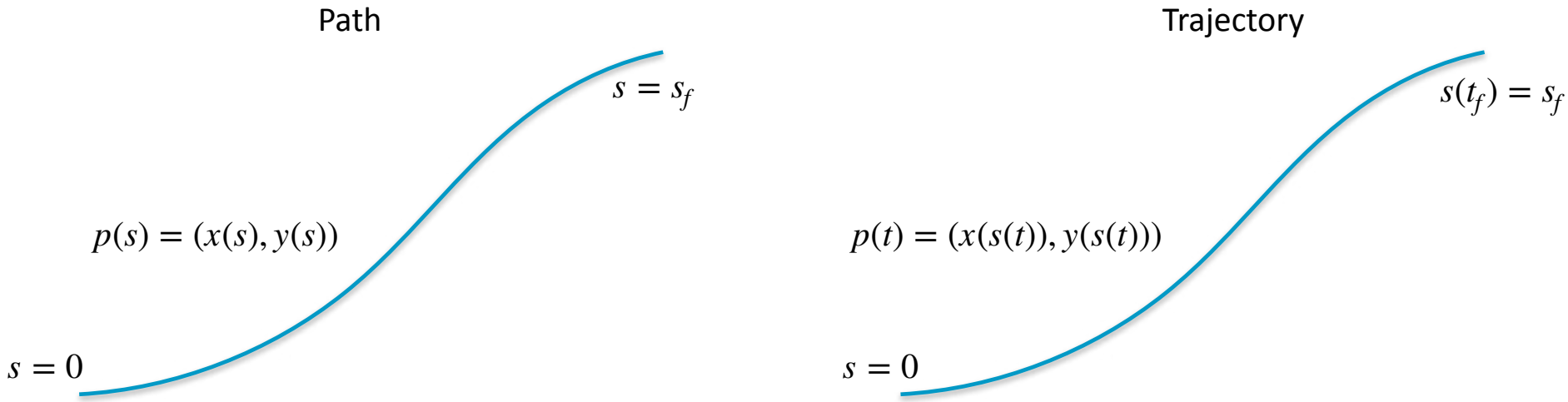


$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \frac{v}{L}\tan(\delta)$$

$$m(\dot{v}_x - v_y\dot{\psi}) = F_{x,f}\cos(\delta) + F_{x,r} - F_{y,f}\sin(\delta) = F_X$$
$$m(\dot{v}_y + v_x\dot{\psi}) = F_{y,f}\cos(\delta) + F_{y,r} + F_{x,f}\sin(\delta) = F_Y$$
$$I_Z\ddot{\psi} = l_f F_{y,f}\cos(\delta) - l_r F_{y,r} + l_f F_{x,f}\sin(\delta) = M_Z$$

Paths and trajectories

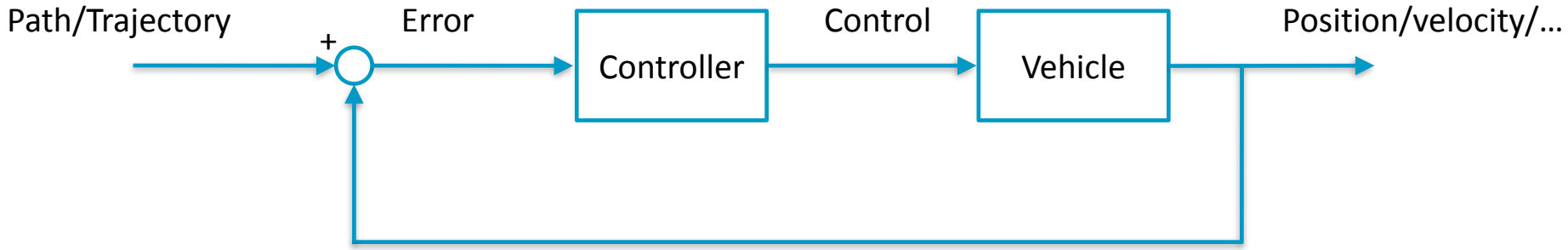
Path vs trajectory



- Path - $p(s), s \in [0,\ s_f]$, points parameterized by position s
- Trajectory - $p(s(t)),\ t \in [0,\ t_f]$. points parameterized by time t
- Path to trajectory through a velocity profile

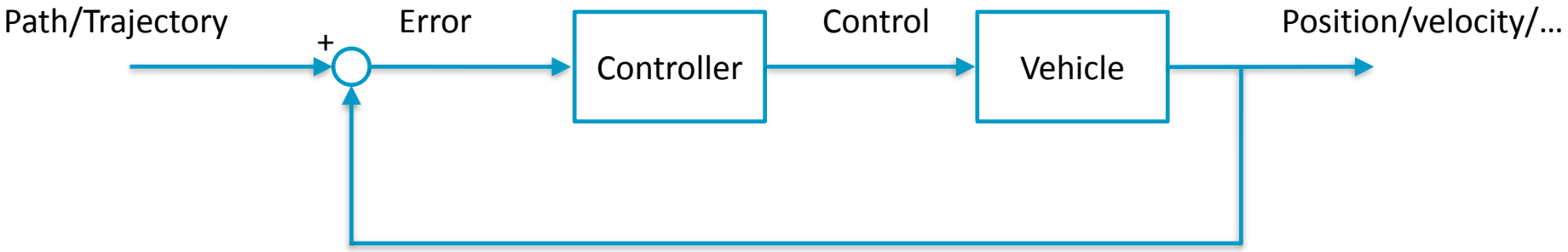
$$s(t) = \int_0^t v(\tau)\, d\tau$$

Trajectory control or path + velocity control

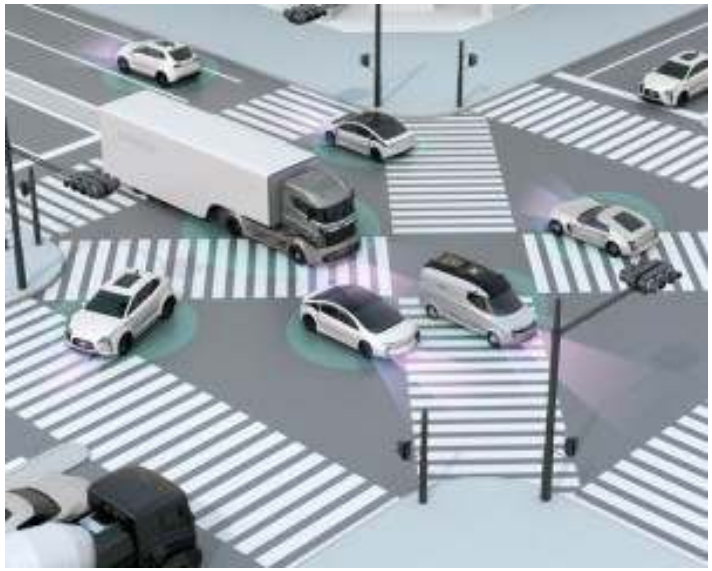


- Trajectory control/stabilization $\approx p_{veh}(t) \rightarrow p_{ref}(t)$
- Path control/stabilization $\approx p_{veh}(s(t)) \rightarrow p_{ref}(t)$ for some function $s(t)$
  - $s(t)$ position at point t ~ velocity function $v(t) = \dot{s}(t)$
- Sometimes; trajectory control is split into two sub-controllers
  - Path control using steering, and velocity control using acceleration/ velocity
  - Simpler to separate, but combined approaches have more freedom and could result in better performing solutions

Trajectory control or path + velocity control



- For collision avoidance, it clearly matters not only where to drive but also when you are at specific locations.
- Path control is mathematically relaxed compared to trajectory control
- An essential part of vehicle control and this lecture will focus on path control



Curvature - rate of change of heading

- Curvature $\kappa(s)$ of a planar curve is defined as how fast the tangent T approaches the normal N

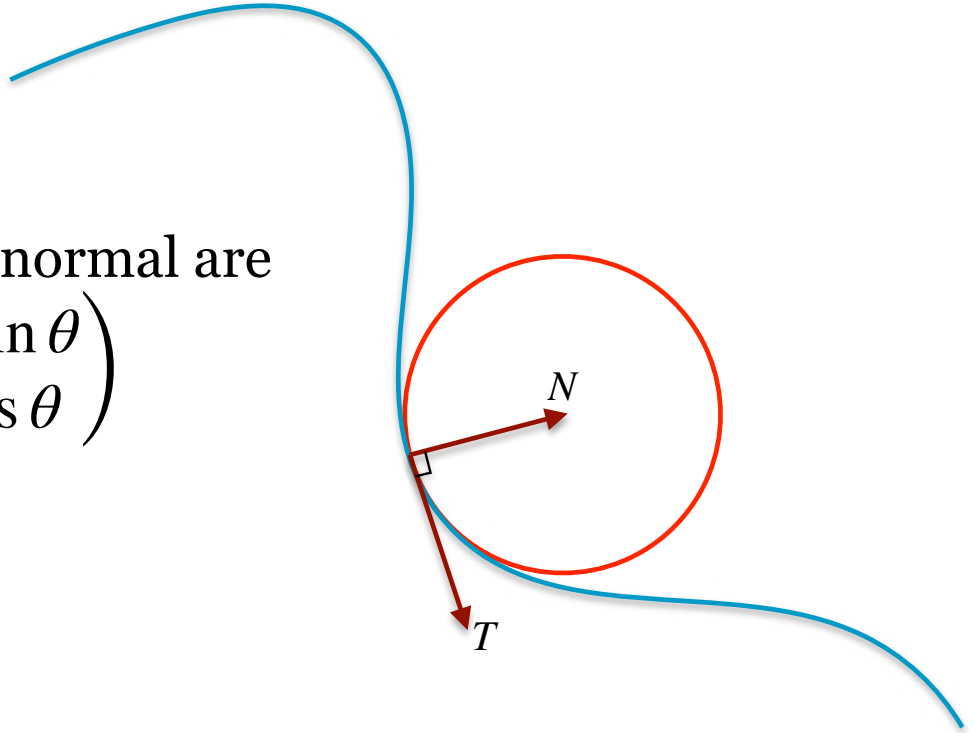
$$\frac{dT(s)}{ds} = \kappa(s)N(s)$$

- With heading θ, the tangent and normal are

$$T = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \; N = \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}$$

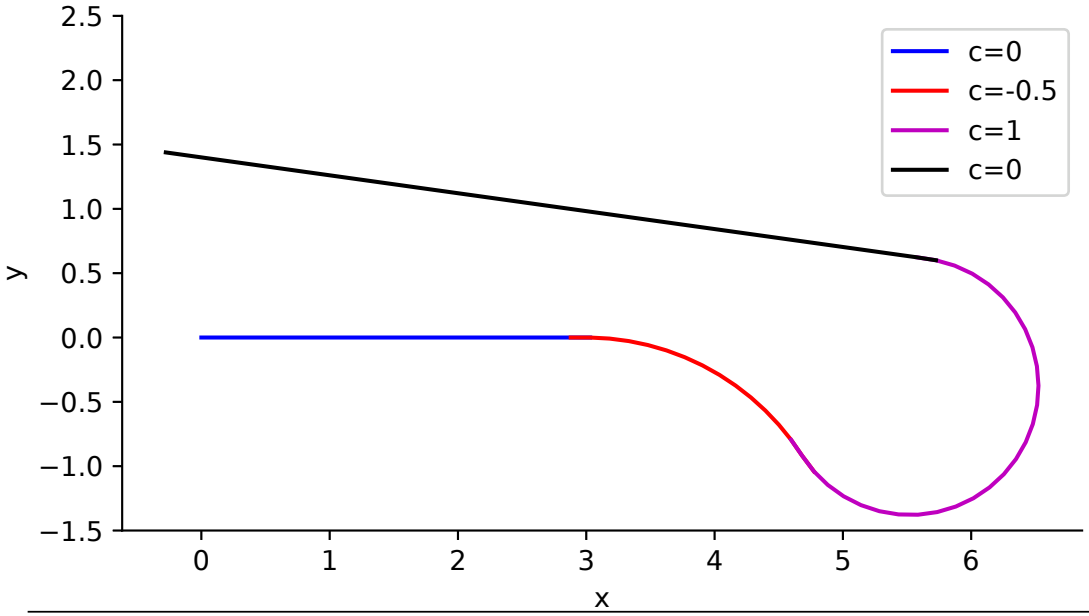
and then the curvature is

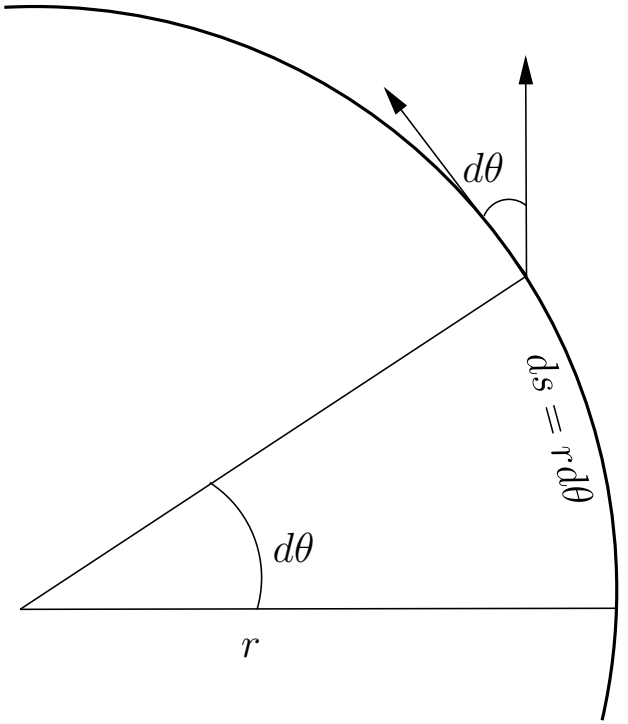
$$\kappa = \frac{d\theta}{ds}$$



Curvature - rate of change of heading

- A straight line has $\kappa(s) = 0$
- A circle has $\kappa(s) = \dfrac{d\theta}{ds} = 1/r$
- $\kappa(s) > 0$ counter clockwise





Curvature - rate of change of heading

- In cartesian coordinates one can show that locally, for a curve parameterized by a position parameter s, $p(s) = (x(s), y(s))$ the curvature is

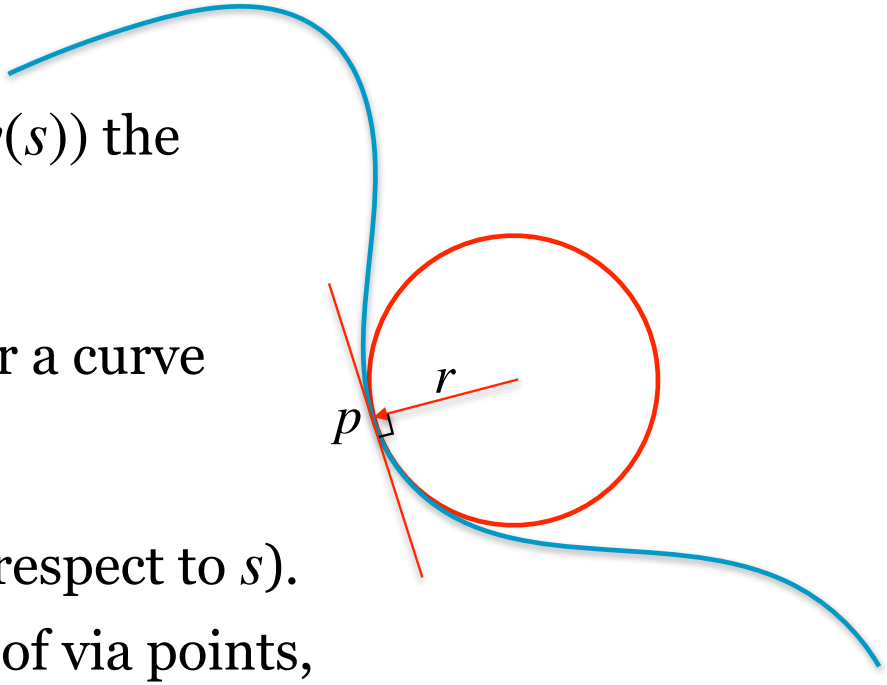
$$\kappa = x'y'' - y'x''$$

- Home exercise: Derive this expression for a curve $(x(s), y(s))$ by differentiating equations

$$x' = \sin(\theta),\ y' = \cos(\theta)$$

and solve for θ' (all differentiations with respect to s).

- A path is often specified by a discrete set of via points, then, e.g., spline interpolation techniques can be used to compute $x'(s), y'(s), x''(s), y''(s)$ [This is the approach in hand-in 3]



Curvature - single-track kinematic model

- For a single-track kinematic model

$$\dot{x} = v \cos(\theta)$$

$$\dot{y} = v \sin(\theta)$$

$$\dot{\theta} = \frac{v}{L} \tan(\delta)$$

- The curvature of the path then becomes

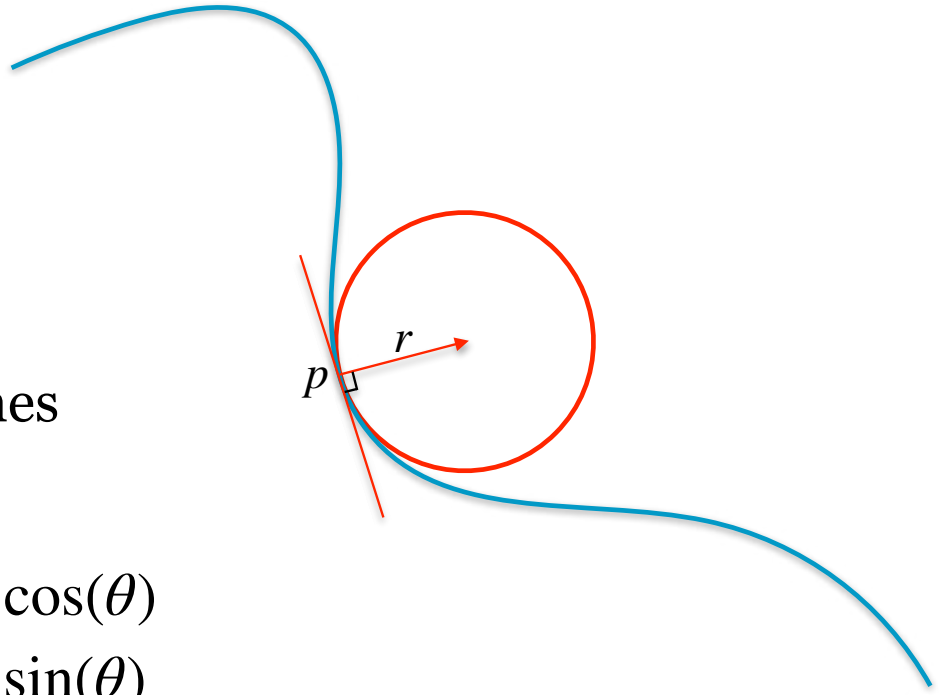
$$\kappa = \frac{d\theta}{ds} = \frac{1}{L} \tan(\delta)$$

- And the model can be written

$$\dot{x} = v \cos(\theta)$$

$$\dot{y} = v \sin(\theta)$$

$$\dot{\theta} = v\,u, \quad u = \frac{1}{L} \tan\delta \sim \text{curvature of path}$$

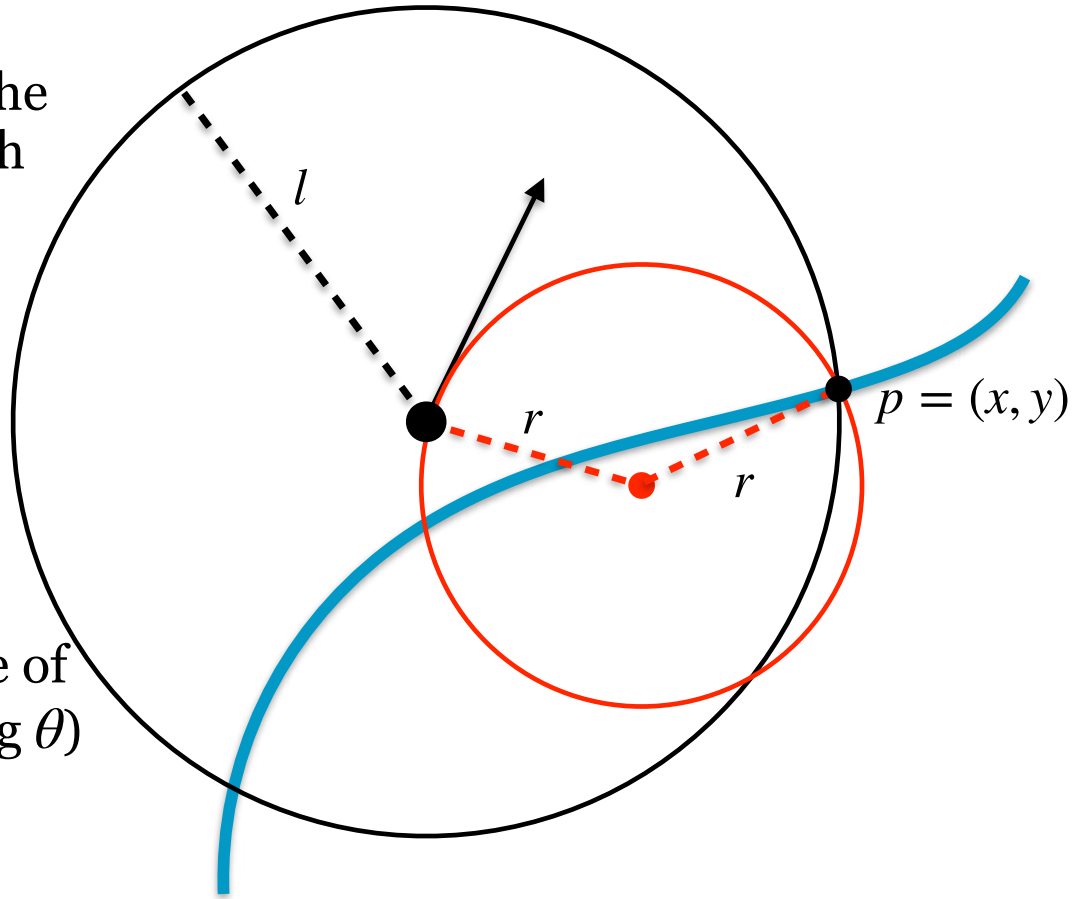


Pure-pursuit control

Pure-pursuit control, basic idea

- A simple control technique to compute the arc needed for a robot to get back on path
- Derived early 80's
- With a look-ahead horizon, l, find point $p = (x, y)$ on the path to aim for
- Compute the turning radius r to get there
- For a single-track ground vehicle, this corresponds to a steering angle with care of sign of r, positive for left-turn (increasing θ)

$$\frac{1}{L}\tan\delta = \frac{1}{r}$$



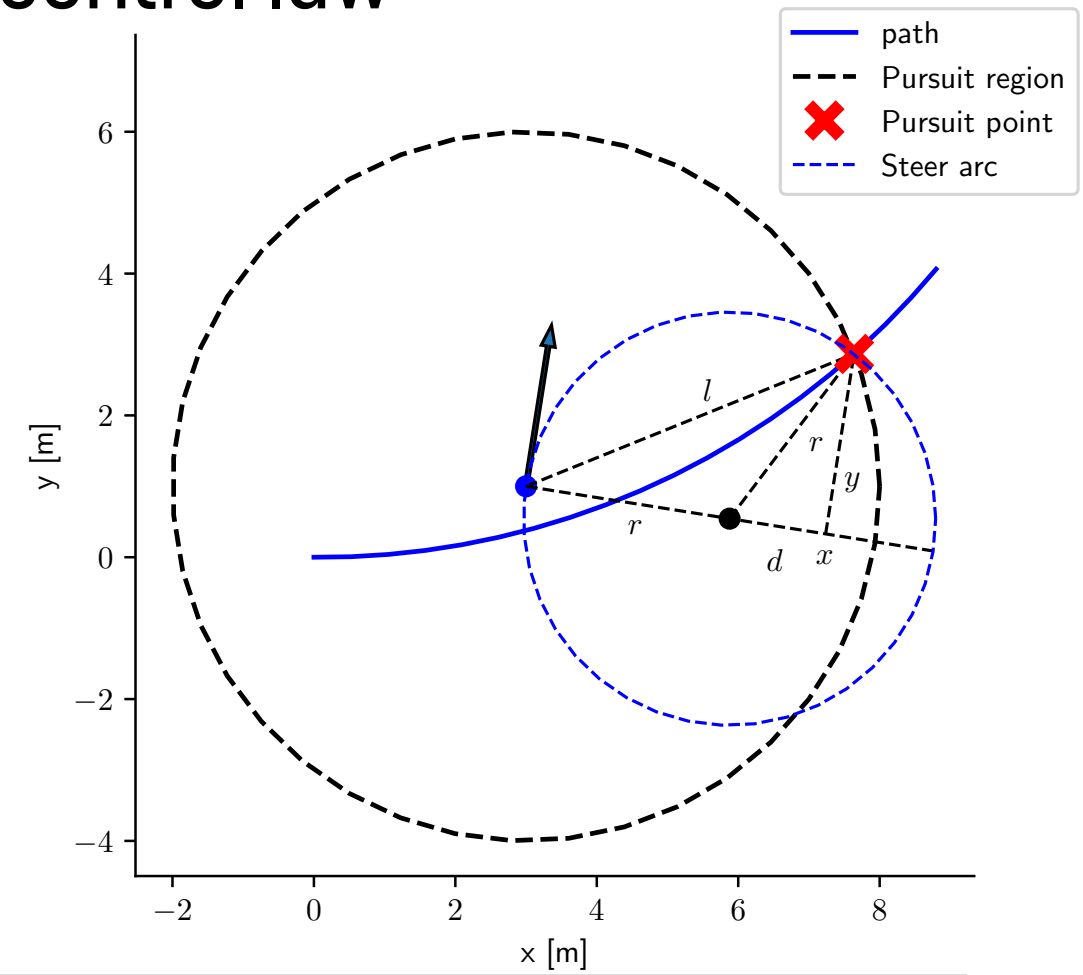
Derivation of pure pursuit control law

- Pure-pursuit horizon $l \sim$ look-ahead horizon
- Pursuit point $p = (x, y)$ intersection between path and look-ahead circle in vehicle local coordinates
- From pursuit point p determine steer radius r and then steer action δ
  - $p = (x, y) \rightarrow r \rightarrow \delta$
- Basic relations

$$l^2 = x^2 + y^2$$

$$r^2 = d^2 + y^2$$

$$x = d + r$$



Control law

- Simple algebra

$$r = \frac{l^2}{2x}$$

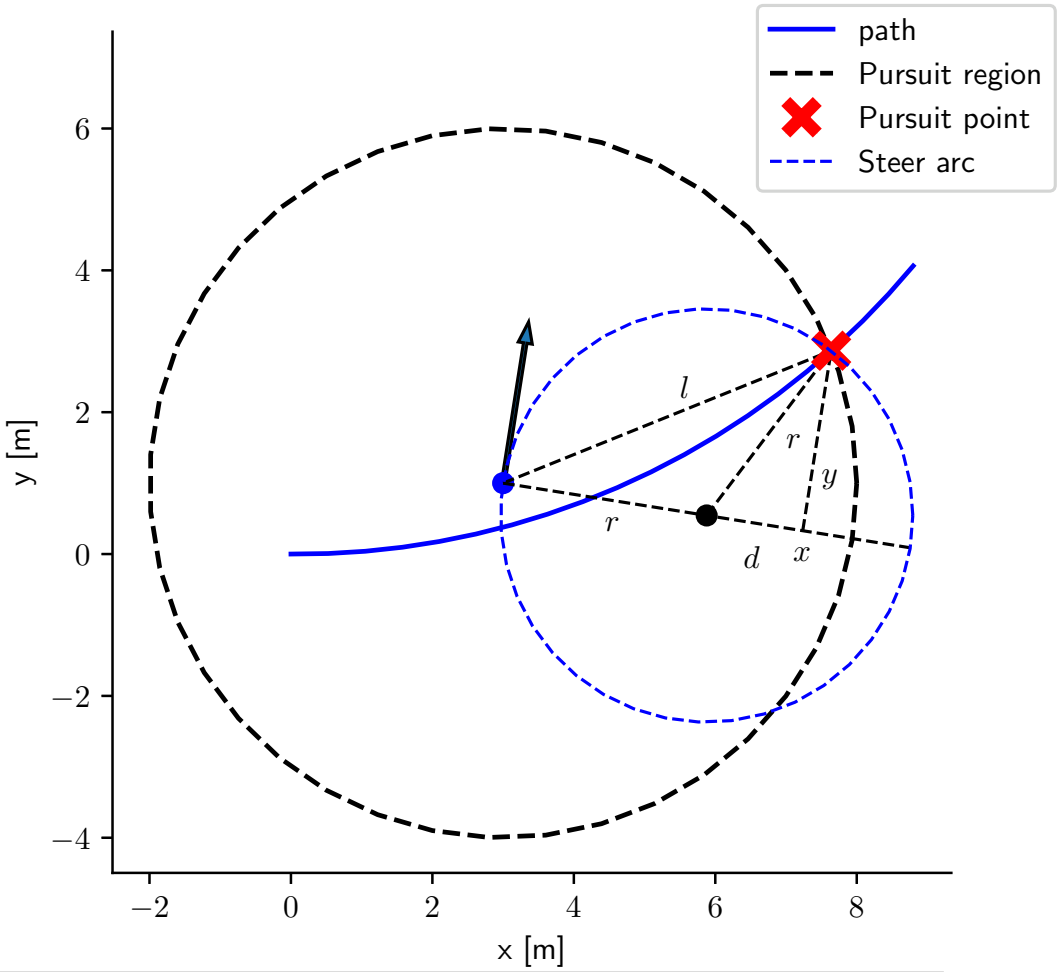
- Curvature of a single track model

$$c = \frac{d\theta}{ds} = \frac{1}{L}\tan\delta$$

- Pure-pursuit control action is then

$$\tan\delta = -L\frac{2x}{l^2}$$

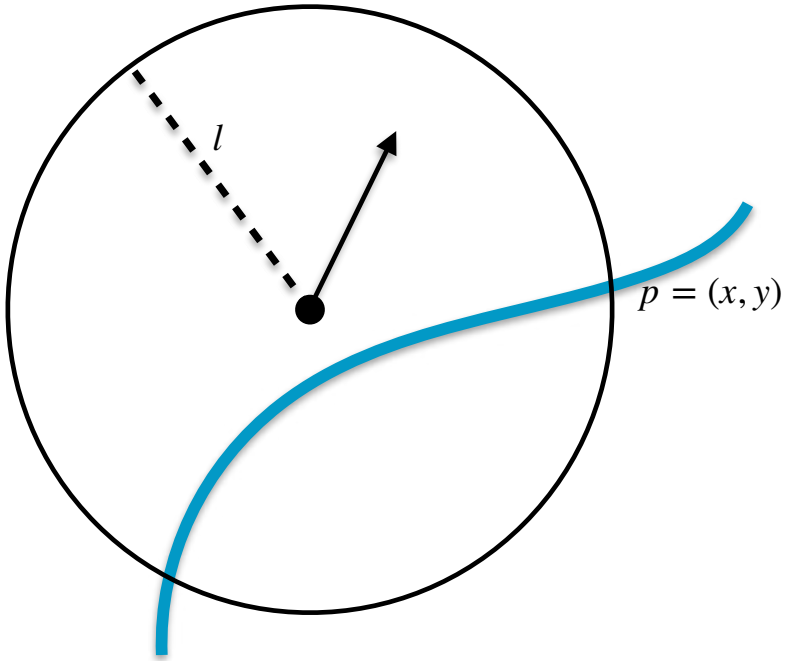
- Negative sign since $\delta > 0$ corresponds to left-turn (increasing θ)



Pure-pursuit, summary

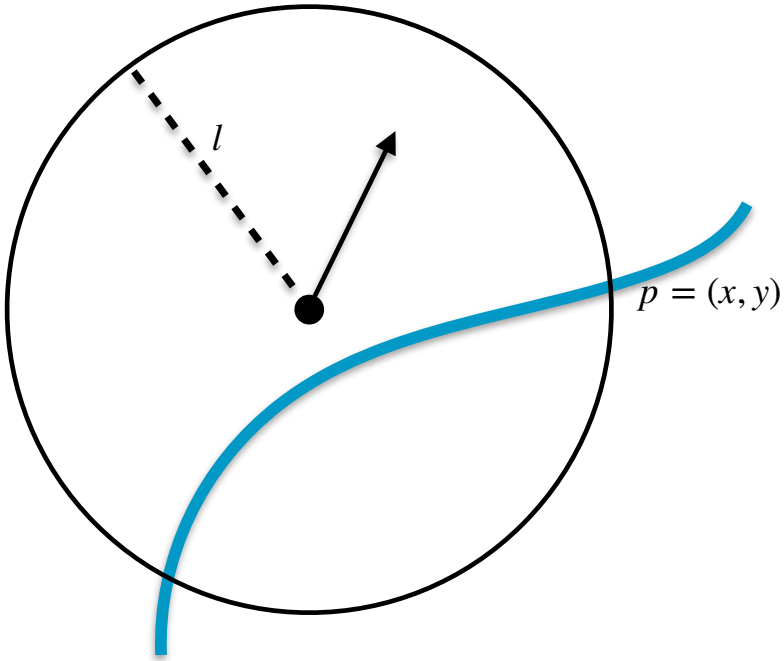
1. Determine intersection point p between path and look-ahead circle (with radius l)
2. Determine coordinates of p in vehicle-local coordinate system $p = (x, y)$
3. Steering angle δ for single-track model with wheel-base L

$$\tan \delta = -L \frac{2x}{l^2}$$



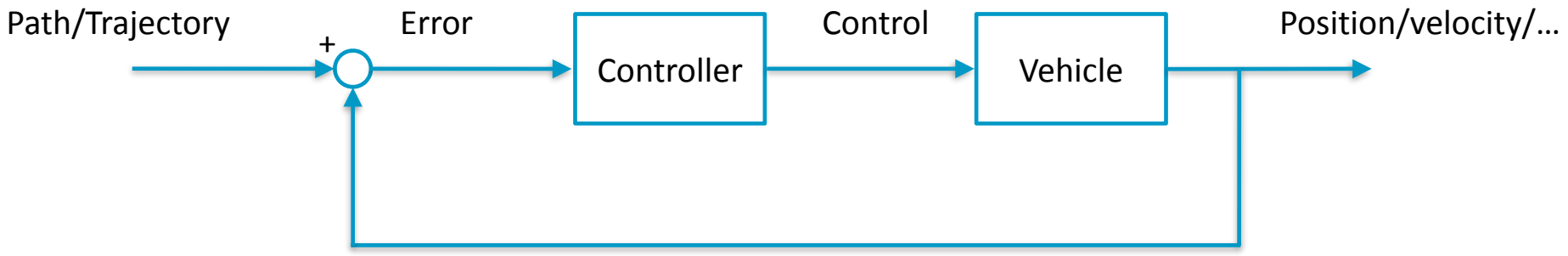
Some implementation considerations

- How point p is determined is important for smooth controls
- Look-ahead horizon
  - Short horizon — agile but less smooth
  - Long horizon — smooth and less agile
- Possible extensions
  - Ellipses instead of circles
  - Offset circle
  - Speed dependencies
  - …



Formulating path tracking as a state-stabilization problem

State-feedback control



$$\dot{x} = g(x,u) \qquad u = K(y, \text{ref}) =$$

$$y = h(x) \qquad = K_0(\text{ref}) - K_1(y - \text{ref})$$

Feed-forward

Feed-back

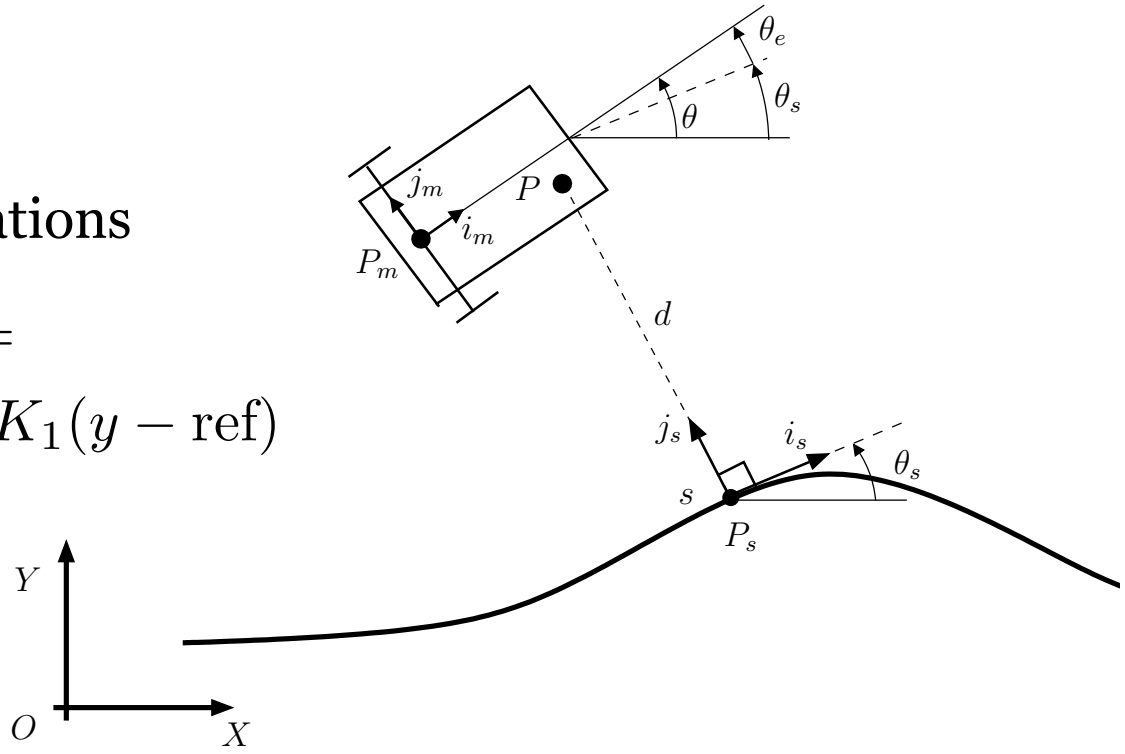
- Classical stabilization problem, choose controller such that $y \to \text{ref}$
- With $y = x$, it is a state-feedback problem; otherwise output feedback

Formulating path tracking as a stabilization problem

- Distance from path is described by two variables
  d - distance from path
  θ_e - heading error
- To reformulate, write the equations

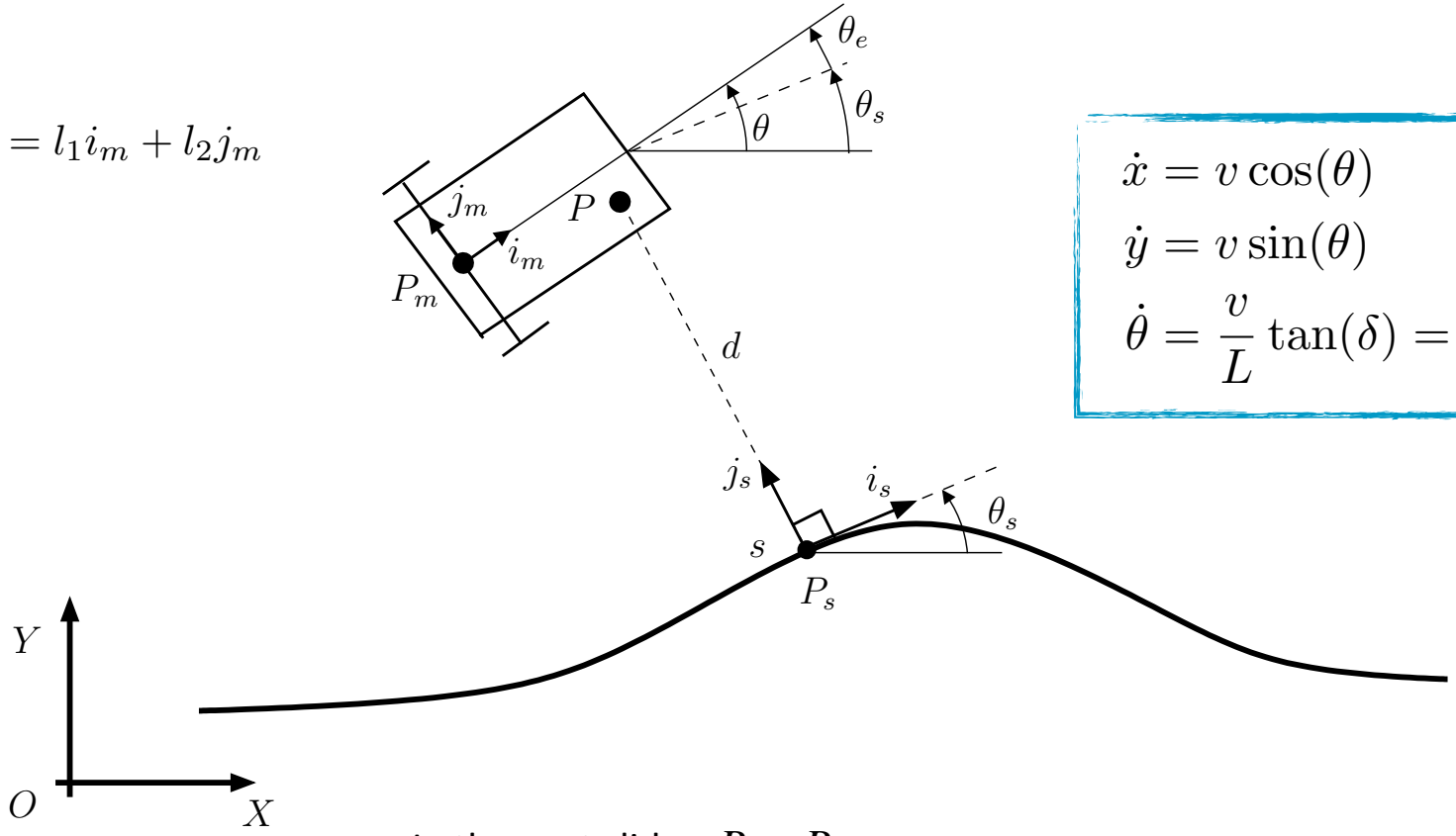
$$\dot{x} = g(x,u) \quad u = K(y,\text{ref}) =$$
$$y = h(x) \qquad\quad = K_0(\text{ref}) - K_1(y - \text{ref})$$

such that $x = (d, \theta_e)$



Frenet frame/local error equations

$$\overrightarrow{P_m P} = l_1 i_m + l_2 j_m$$



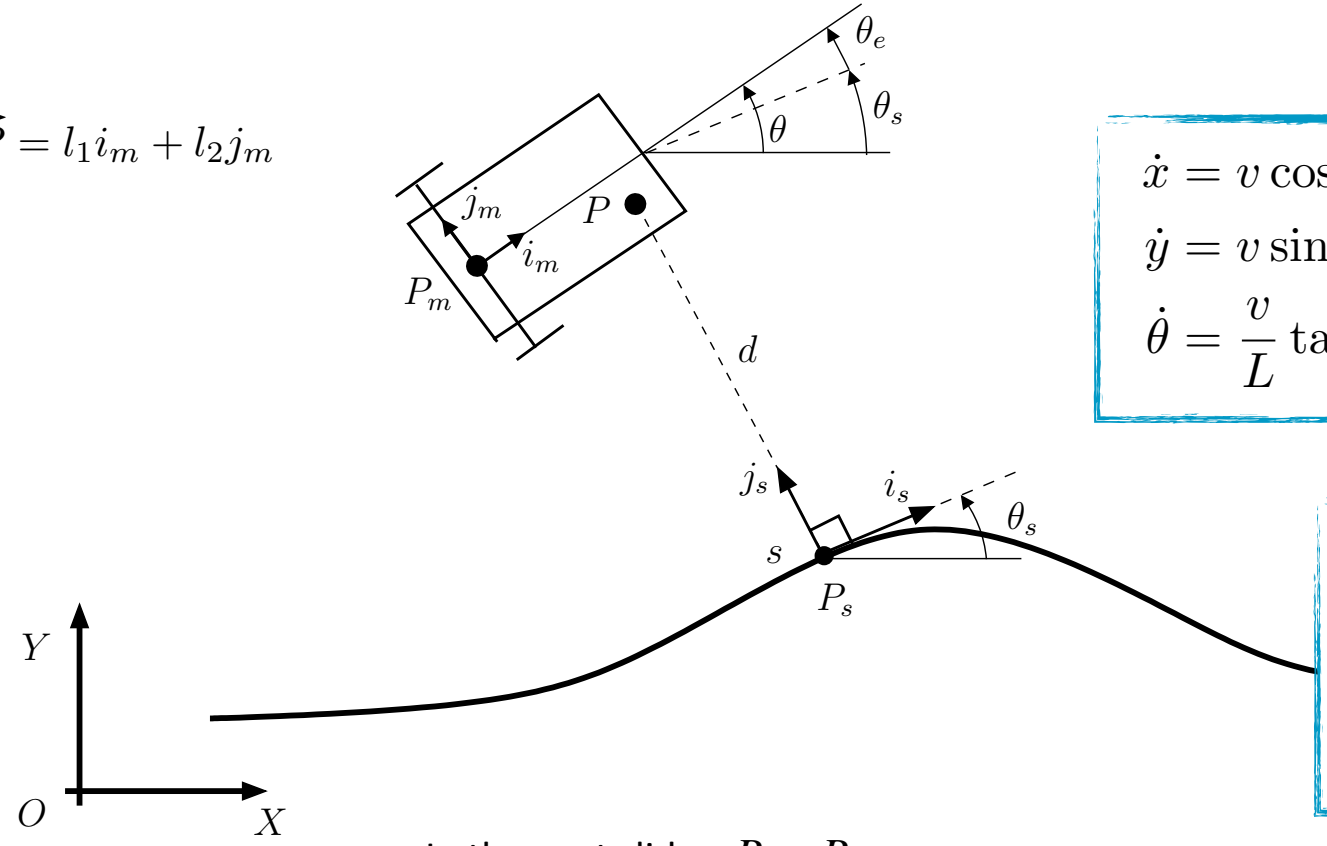
$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \frac{v}{L}\tan(\delta) = v\,u, \quad u \sim \text{curvature}$$

$$\dot{d} = ?$$
$$\dot{\theta}_e = ?$$

In the next slides, $P = P_m$

Frenet frame/local error equations

$$\overrightarrow{P_m P} = l_1 i_m + l_2 j_m$$



$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \frac{v}{L}\tan(\delta) = v\,u, \quad u \sim \text{curvature}$$

$$\dot{s} = \frac{v}{1 - dc(s)}\cos(\theta_e)$$
$$\dot{d} = v\sin(\theta_e)$$
$$\dot{\theta}_e = v\,u - \dot{s}c(s)$$

In the next slides, $P = P_m$

Path following

$$(P = P_m)$$

$$\dot{s} = \frac{v}{1 - dc(s)} \cos(\theta_e)$$

$$\dot{d} = v \sin(\theta_e)$$

$$\dot{\theta}_e = v\,u - \dot{s}c(s)$$

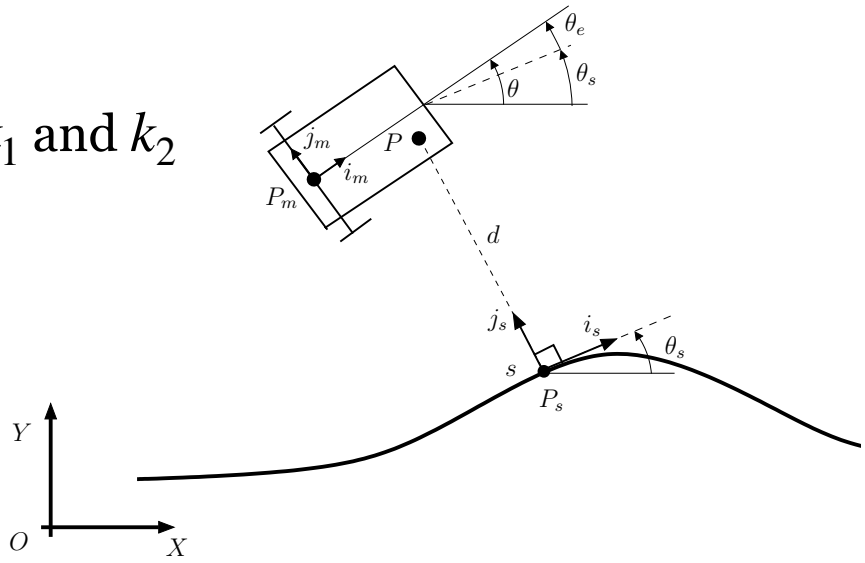
- A linear controller with feed-forward term u_0

$$u = u_0 - k_1 d - k_2 \theta_e$$

- Steer signal δ is then determined in

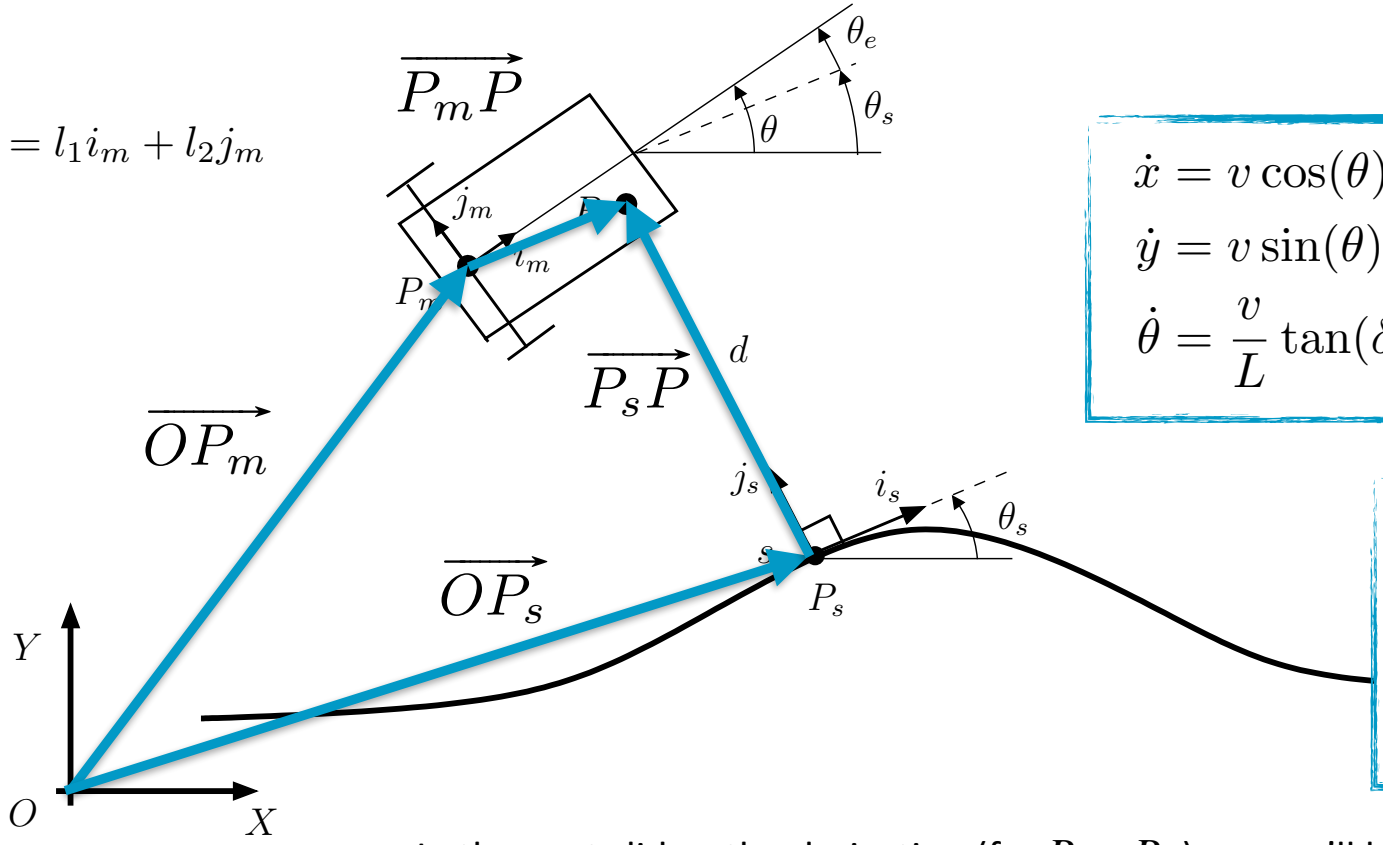
$$\frac{1}{L} \tan \delta = u$$

- Design variables are then: u_0 and constants k_1 and k_2
- Stability: $(d, \theta_e) \to 0$
- LQR-design is a systematic way (more later)
- Non-linear control design opens up new possibilities



Derivation of Frenet frame/local error equations

$$\overrightarrow{P_m P} = l_1 i_m + l_2 j_m$$



$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \frac{v}{L}\tan(\delta) = v\,u, \quad u \sim \text{curvature}$$

$$\dot{s} = \frac{v}{1 - dc(s)}\cos(\theta_e)$$
$$\dot{d} = v\sin(\theta_e)$$
$$\dot{\theta}_e = v\,u - \dot{s}c(s)$$

In the next slides, the derivation (for $P = P_m$), so you'll know how to derive for other vehicles

Derivation of Frenet equations for single-track model

- For curvature it holds

$$c(s) = \frac{d\theta_s}{ds} = \frac{d\theta_s}{dt}\left(\frac{ds}{dt}\right)^{-1} \Rightarrow \dot{\theta}_s = c(s)\dot{s}$$

- Then the heading error dynamics are

$$\dot{\theta}_e = \dot{\theta} - \dot{\theta}_s = v\,u - \dot{s}c(s)$$

- For movement in a rotating coordinate system (see your old calculus course)

$$\frac{dp}{dt} = \frac{d^*p}{dt} + \omega \times r$$

First expression ($\overrightarrow{OP} = \overrightarrow{OP_s} + \overrightarrow{P_sP}$)

$$\frac{d\overrightarrow{OP}}{dt} = \frac{d\overrightarrow{OP_s}}{dt} + \frac{d\overrightarrow{P_sP}}{dt} \qquad \frac{d\overrightarrow{OP_s}}{dt} = \dot{s}i_s$$

$$\frac{d\overrightarrow{P_sP}}{dt} = \dot{d}j_s + \begin{pmatrix}0\\0\\\dot{\theta}_s\end{pmatrix} \times \begin{pmatrix}0\\d\\0\end{pmatrix}$$

$$= \dot{d}j_s - d\dot{\theta}_s i_s = \dot{d}j_s - dc(s)\dot{s}i_s$$

Derivation of Frenet equations for single-track model

- Second expression ($\overrightarrow{OP} = \overrightarrow{OP_m} + \overrightarrow{P_mP}$)

$$\frac{d\overrightarrow{OP}}{dt} = \frac{d\overrightarrow{OP_m}}{dt} + \frac{d\overrightarrow{P_mP}}{dt} \qquad \frac{d\overrightarrow{OP_m}}{dt} = v\, i_m \qquad \frac{d\overrightarrow{P_mP}}{dt} = 0 + \begin{pmatrix} 0 \\ 0 \\ \dot{\theta} \end{pmatrix} \times \begin{pmatrix} l_1 \\ l_2 \\ 0 \end{pmatrix}$$

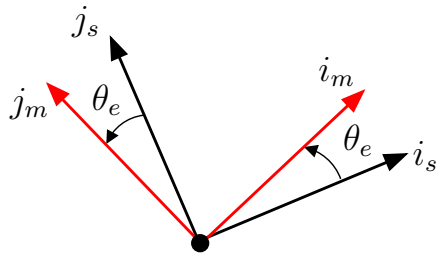
- Setting expressions equal, expressing all in (i_s, j_s)-frame, and some algebra (with $dc(s) < 1$)

$$\dot{s} = \frac{1}{1 - dc(s)}\left[(v - l_2 u)\cos(\theta_e) - l_1 u \sin(\theta_e)\right]$$

$$\dot{d} = (v - l_2 u)\sin(\theta_e) + l_1 u \cos(\theta_e)$$

$$\dot{\theta}_e = \dot{\theta} - \dot{\theta}_s = v\, u - \dot{s} c(s)$$

$$i_m = \cos(\theta_e) i_s + \sin(\theta_e) j_s$$
$$j_m = -\sin(\theta_e) i_s + \cos(\theta_e) j_s$$

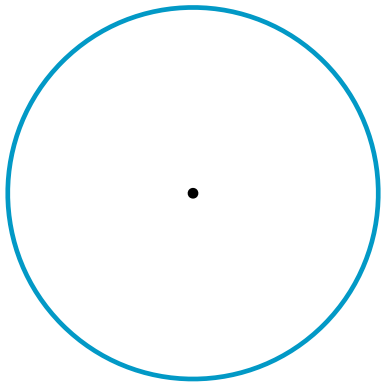


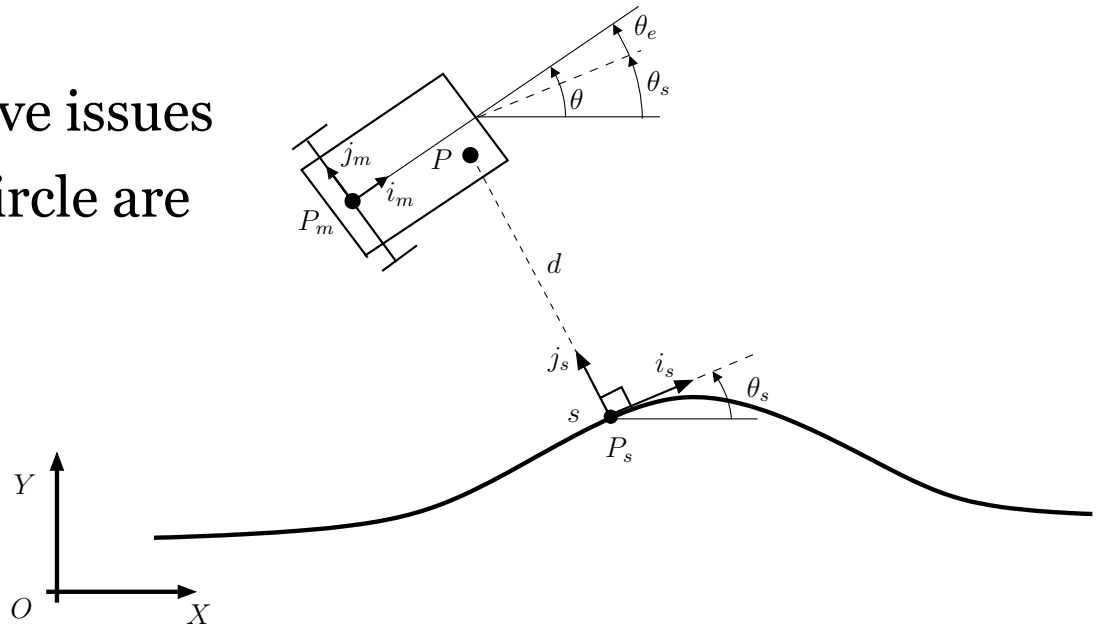
- Stabilizing $(d, \theta_e) \to 0$ means following the path
- Note that s is not a state, only (d, θ_e)

What does $d\,c(s) < 1$ mean?

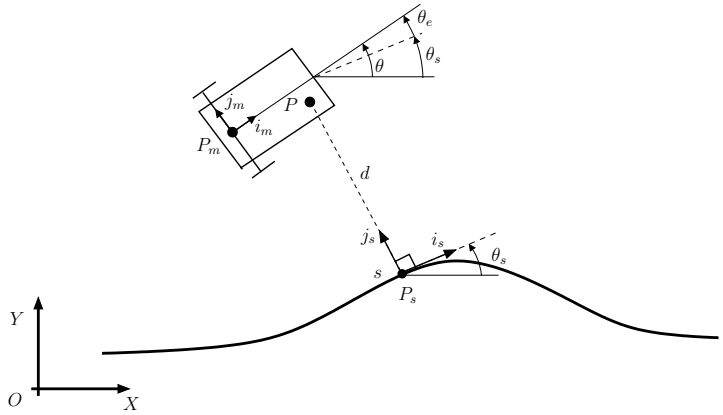
$$\dot{s} = \frac{v}{1 - dc(s)} \cos(\theta_e)$$
$$\dot{d} = v \sin(\theta_e)$$
$$\dot{\theta}_e = v\,u - \dot{s}c(s)$$

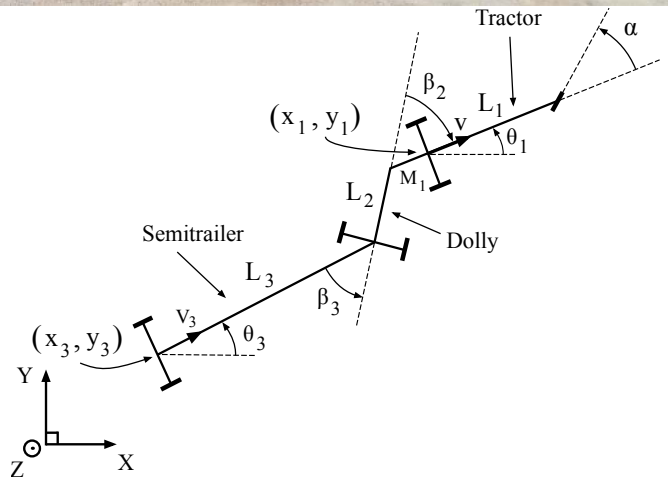
- d is the orthogonal distance from the vehicle to the path
- What happens when $dc(s) = 1$?
  - $c(s) = 1/r$, when $d = r$ we have issues
- Extreme case; all points on the circle are orthogonal and same distance





Why did I show this algebra?





- Not expected to memorize calculations
- Other kinds of vehicles
- Tracking-point *P* might be very important, e.g., when reversing the truck to the right

Figures from "*A path planning and path-following control framework for a general 2-trailer with a car-like tractor*", O. Ljungqvist, et.al.

Some implementation comments

- Tuning of controller means selecting
  - feedback gain
  - Feed-forward — anticipate curves, we know what is coming
  - Vehicle does not perfectly follow the kinematic motion model, feedback introduces robustness!
- Main step is to determine d and θ_e
- Note
  - distance error d has a sign
  - Angles wrap around 2π, tempting to litter code with modulo 2π. **Don't do that!** Instead look at next slide!

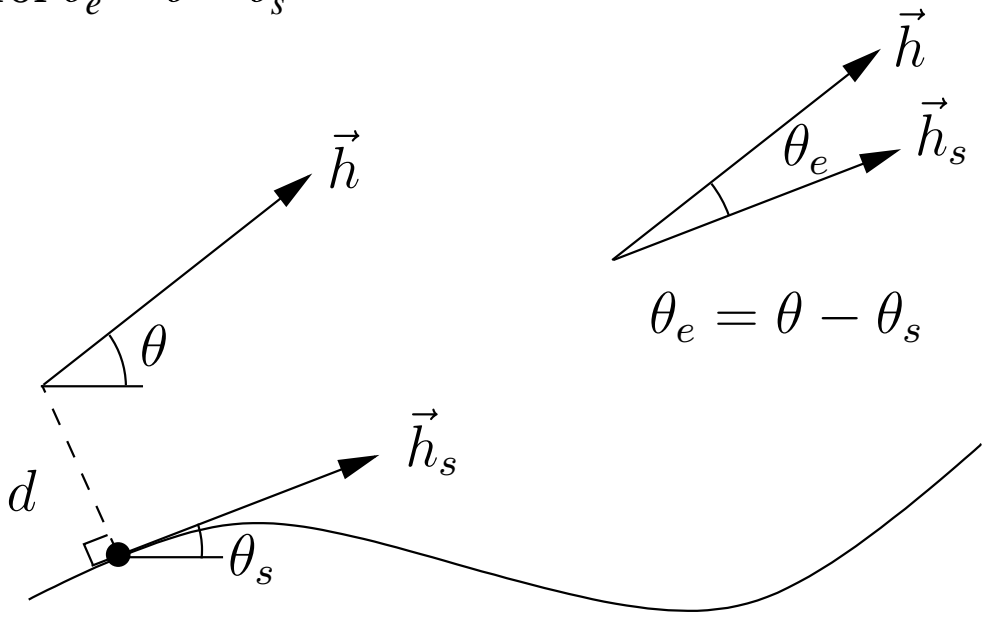
Heading error using cross and dot products

- Heading vectors h and h_s, heading error $\theta_e = \theta - \theta_s$
- Properties of dot and cross product
  - $\vec{\mathbf{x}} \cdot \vec{\mathbf{y}} = |\vec{\mathbf{x}}||\vec{\mathbf{y}}|\cos\alpha$
  - $\vec{\mathbf{x}} \times \vec{\mathbf{y}} = |\vec{\mathbf{x}}||\vec{\mathbf{y}}|\sin\alpha\vec{\mathbf{n}}$
- Use these for heading vectors

$$\vec{h}_s \cdot \vec{h} = |\vec{h}_s||\vec{h}|\cos\theta_e$$

$$\begin{pmatrix} h_s \\ 0 \end{pmatrix} \times \begin{pmatrix} h \\ 0 \end{pmatrix} = |\vec{h}_s||\vec{h}|\sin\theta_e \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

and $\theta_e = \arctan\dfrac{\sin\theta_e}{\cos\theta_e}$.



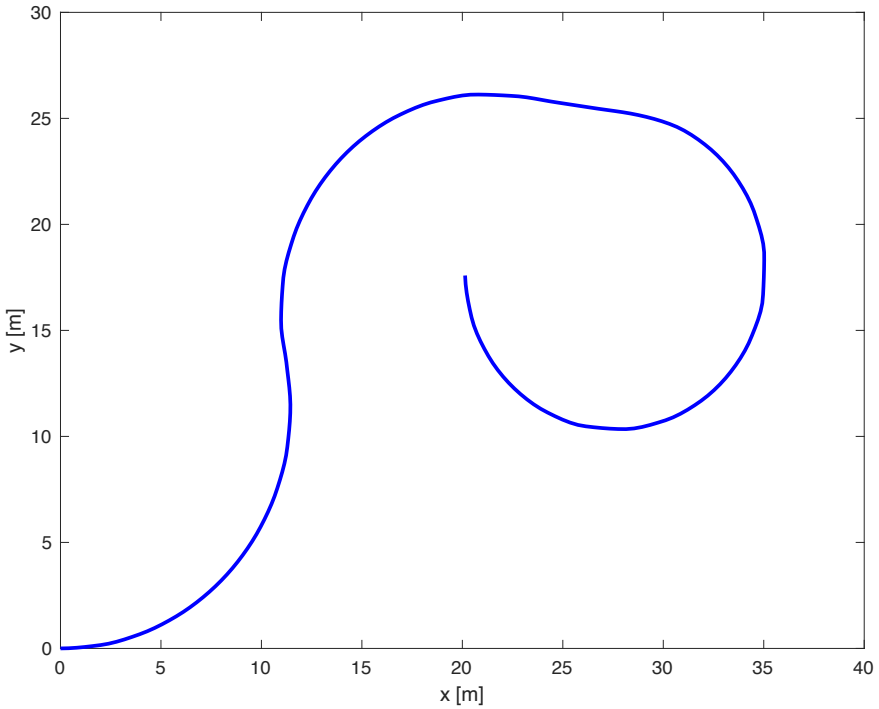
$$\theta_e = \theta - \theta_s$$

Important

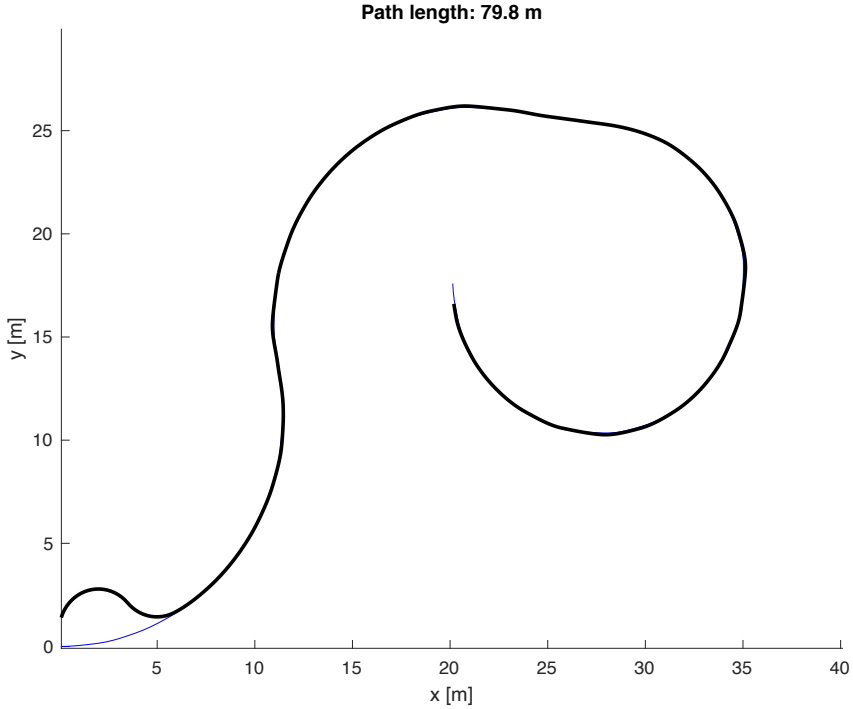
Use function `arctan2()` to compute θ_e, otherwise you get problems for $\theta_e = \pi/2$.

Linear controller

A linear feedback controller can be expected to work well close to the path



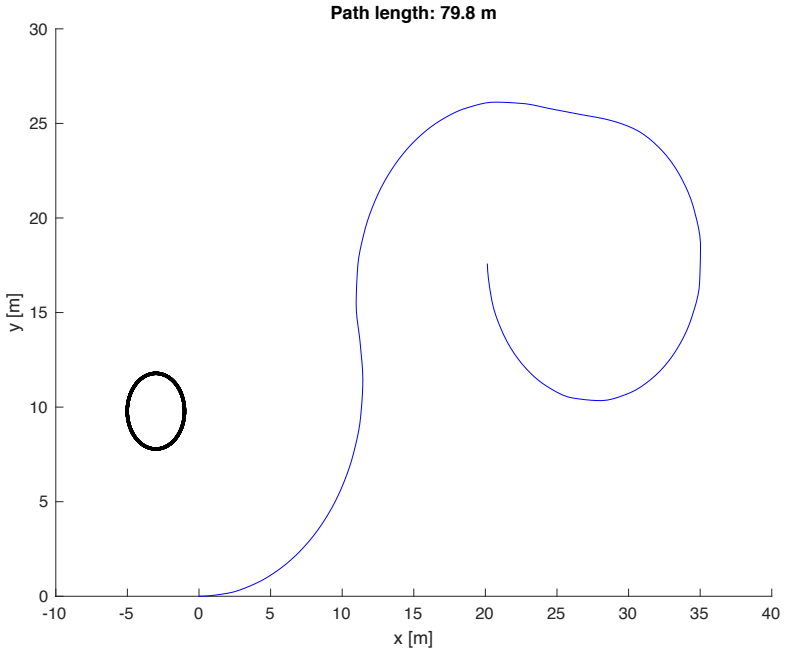
Path



Linear controller

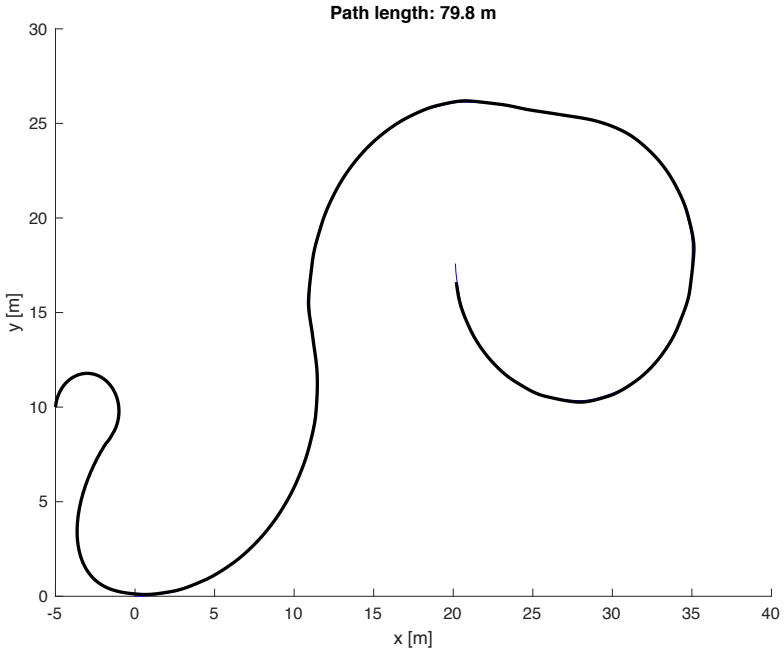
Limitations of linear controllers

A linear feedback controller can be expected to work well close to the path



Linear controller

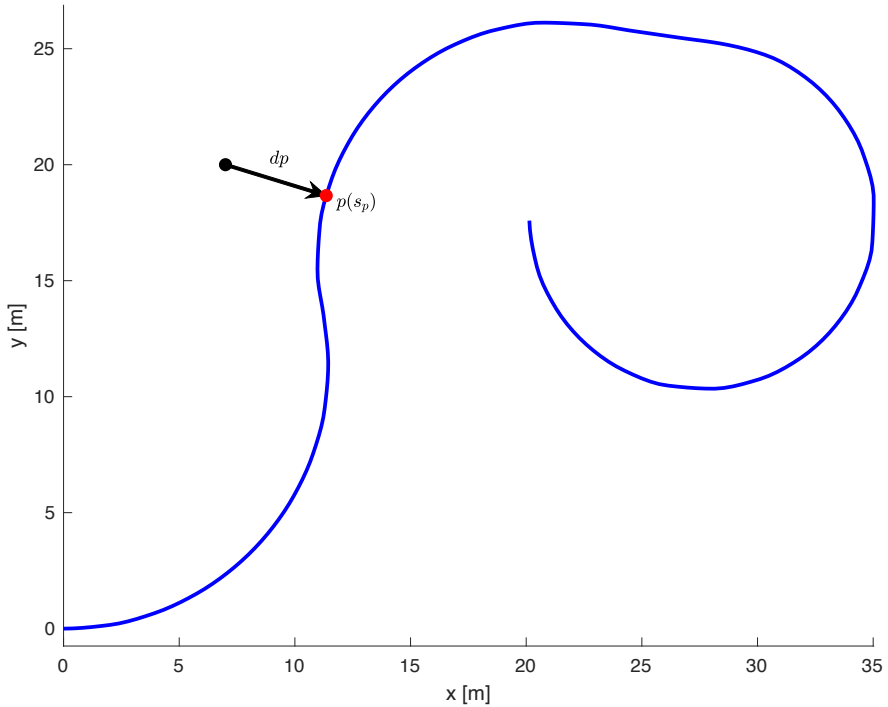
$$u = u_0 - k_1 d - k_2 \theta_e$$

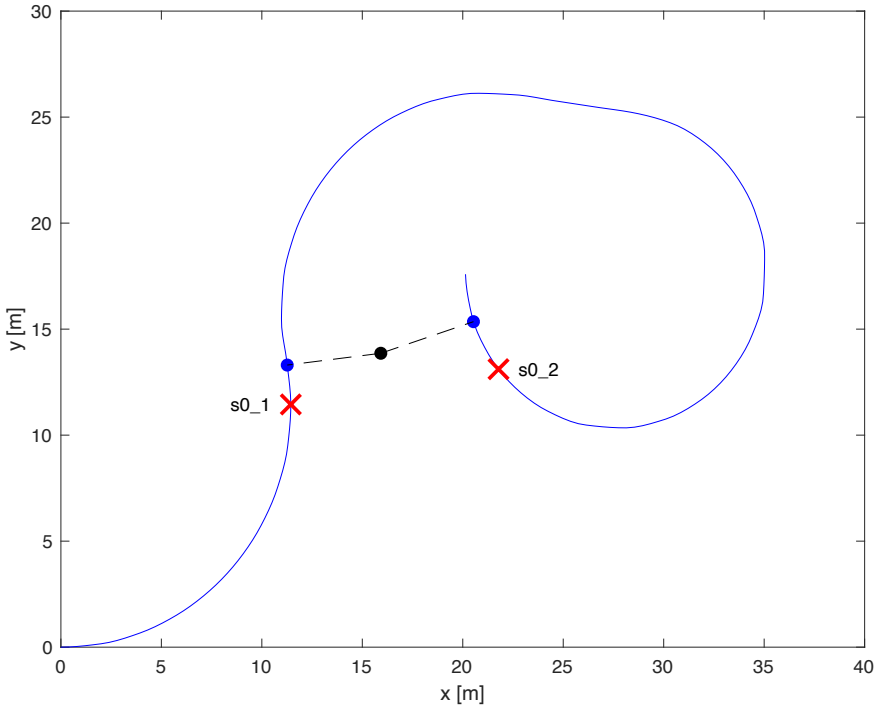


Non-linear controller

$$u = u_0 - k_1 \frac{\sin \theta_e}{\theta_e} d - k_2 \theta_e$$

A main difficulty – efficient determination of d and θ_e





- Efficient and smooth determination of distance and heading error is a main difficulty

A FIRST EXPERIMENT

PATH FOLLOWING IN CARLA

ERIK FRISK, ISY

Feedback control design - LQR and Nonlinear Design

Linear Quadratic Regulators (discrete time)

- A systematic way to compute the feedback gain for linear systems is LQR - Linear Quadratic regulators

$$x_{t+1} = Ax_t + Bu_t, \quad u_t = -Kx_t$$

- The feedback gain K minimizing the cost-function

$$J = \sum_{t=0}^{\infty} x_t^T Q x_t + u_t^T R u_t$$

is given by $K = (R + B^T P B)^{-1} B^T P A$ where P is the solution to the Discrete-time Algebraic Riccati Equation (DARE)

$$P = A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A + Q$$

- Looks hairy, but the solvers are available: `idare` (Matlab) or `scipy.linalg.solve_discrete_are` (Python)

Linear Quadratic Regulators

- Cost function from last slide

$$J = \sum_{t=0}^{\infty} x_t^T Q x_t + u_t^T R u_t,$$

- Matrices Q and R are the tuning variables for the controller
- For the path follower $x = (d, \theta_e)$ and $u =$ curvature. With

$$Q = \begin{pmatrix} q_1 & 0 \\ 0 & q_2 \end{pmatrix},$$

the parameters get intuitive meaning and the cost-function becomes

$$\sum_{t=0}^{\infty} q_1 d_t^2 + q_2 \theta_{e,t}^2 + R\kappa_t^2$$

Applying the LQ controller

- The LQR controller is based on a *linear* model, the error system of the Frenet frame is non-linear
  - This means you have to linearize
    - around which operating condition?
    - Which speed?
    - Which curvature?
- Compute a feedback gain and keep it constant, I don't recommend a time-varying feedback gain and recomputing K in every step.

Linearized error dynamics

- Linearized error dynamics can be useful in simple controller design
- Non-linear error dynamics,

$$\dot{s} = \frac{v}{1 - d\,c(s)} \cos(\theta_e)$$

$$\dot{d} = v\,\sin(\theta_e)$$

$$\dot{\theta}_e = v\,u - \dot{s}\,c(s), \quad u \sim \text{ curvature}$$

- Linearized error dynamics given using the Jacobians

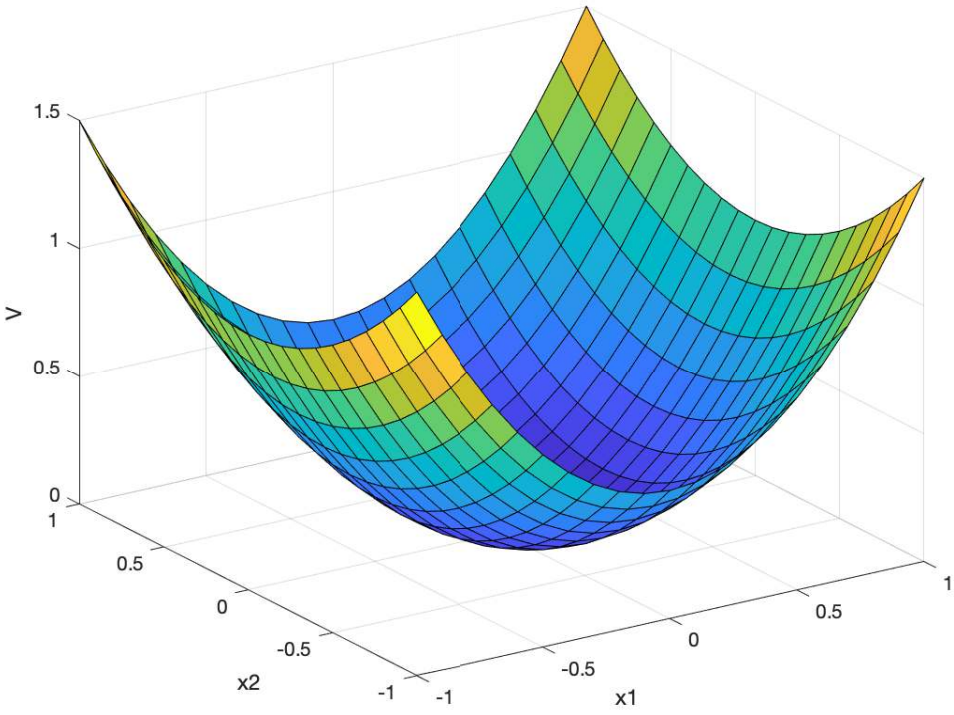
$$\frac{\partial f}{\partial x} = v \begin{pmatrix} 0 & \cos(\theta_e) \\ -\frac{c(s)^2 \cos\theta_e}{(1 - d\,c(s))^2} & \frac{\sin\theta_e}{1 - d\,c(s)} \end{pmatrix}, \quad \frac{\partial f}{\partial u} = v \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Non-linear path followers - Lyapunov stability

Parts of the material in the following section are slightly more advanced and can be considered as bonus material in the course. The methods presented here will not be part of the basic hand-in exercise.

Stability of non-linear systems – a large topic

- A linear system
  $\dot{x} = Ax$
  is asymptotically stable if
  $\text{eig}(A) < 0$
- Concept of eigenvalues no longer exists for non-linear systems as in linear systems
- One classical approach is to use a *measure* of state size, $V(x) \geq 0$ with equality only for $x = 0$
- If $\dot{V}(x) < 0, x \neq 0$, then the state-size always decreases and then $x \to 0$



Classical example – stability of a damped pendulum

- A damped pendulum is clearly stable, it will settle in the downward position
- State-size ~ energy in the system

$$V = \text{kinetic energy} + \text{potential energy}$$

$$= \frac{mL^2}{2}\omega^2 + mgL(1-\cos\theta) \geq 0$$

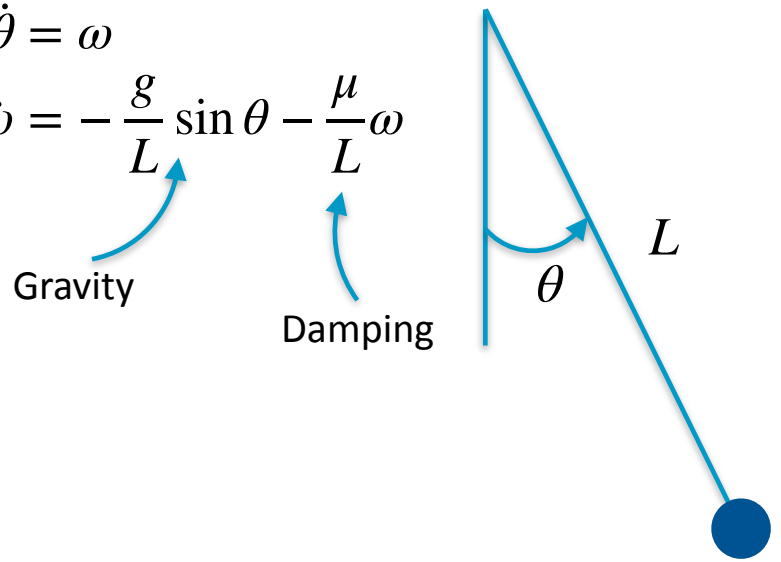
- Simple algebra gives that

$$\dot{V} = -\mu mL\omega^2 \leq 0 \text{ (only semi-definite, here sufficient)}$$

$$\Rightarrow \omega \to 0 \Rightarrow \theta \to 0$$

$$\dot{\theta} = \omega$$

$$\dot{\omega} = -\frac{g}{L}\sin\theta - \frac{\mu}{L}\omega$$



Linear systems and the Lyapunov equation

- For a linear system
  $$\dot{x} = Ax$$
  and a quadratic Lyapunov candidate
  $$V = x^T P x, \quad P > 0,\ P^T = P$$
- Then
  $$\dot{V} = x^T (A^T P + PA) x$$
- Corresponds to the Lyapunov equation which you might have seen
  $$A^T P + PA + Q = 0, \quad Q > 0$$
  which has a symmetric positive-definite solution P iff matrix A is asymptotically stable.
- Common Lyapunov functions: system energy and quadratic forms

Lyapunov stability of path follower

- The nonlinear controller

$$u = u_0 - k_1 \frac{\sin\theta_e}{\theta_e} d - k_2 \theta_e$$

was derived using Lyapunov arguments

- Formulate a size measure

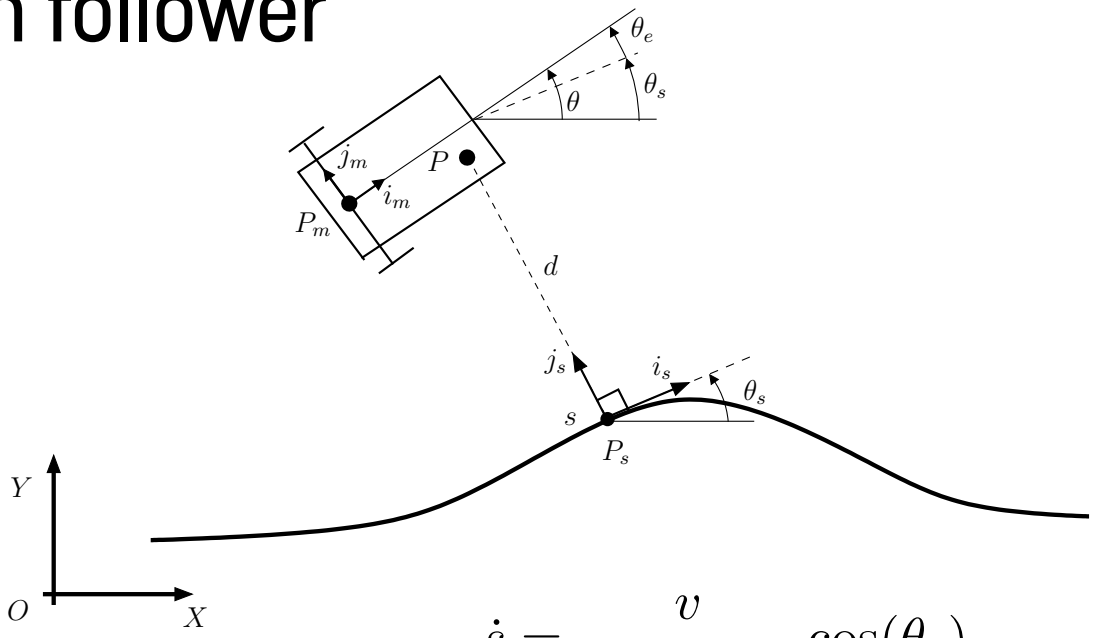
$$V(d, \theta_e) \geq 0$$

and choose control-law u such that

$$\dot{V} \leq 0$$

- A quadratic form

$$V = c_1 d^2 + c_2 \theta_e^2$$



$$\dot{s} = \frac{v}{1 - dc(s)} \cos(\theta_e)$$

$$\dot{d} = v \sin(\theta_e)$$

$$\dot{\theta}_e = v\, u - \dot{s} c(s)$$

Werling, M., Gröll, L., and Bretthauer, G.. "*Invariant trajectory tracking with a full-size autonomous road vehicle*". IEEE Transactions on Robotics 26.4 (2010): 758-765.

Summary and take-home message

Summary and take-home message

- Introduce a control structure and architecture for path and trajectory following
- Know basic ways to characterize paths, trajectories, curvature
- Pure-pursuit control for path following
- How to transform path following into a classical state-stabilization problem
- Reflect upon non-linear effects on the path follower and how to approach that

Path following algorithms covered

Pure-Pursuit

- Determine point $p = (x, y)$ on path on look-ahead distance l
- Determine curvature κ, or radius r, to get to that point
- Determine steering angle δ such that the vehicle follows a circle segment with radius r
  - For kinematic single-track model

$$\tan\delta = -L\frac{2x}{l^2}$$

State-feedback

- Determine distance-error d and heading-error θ_e
- Select a feedback-law, e.g., a linear controller

$$u = u_0 - K_1 d - K_2 \theta_e$$

that stabilizes the error-dynamics
- Linear controller might not work far from the path

www.liu.se

LINKÖPINGS UNIVERSITET