Control Systems Module II: Review of self-driving car controls

Review of self-driving car controls

Explains

- Overview of state of the art
- Pure pursuit
- Lyapunov control
- Feedback linearization
- Model Predictive Control

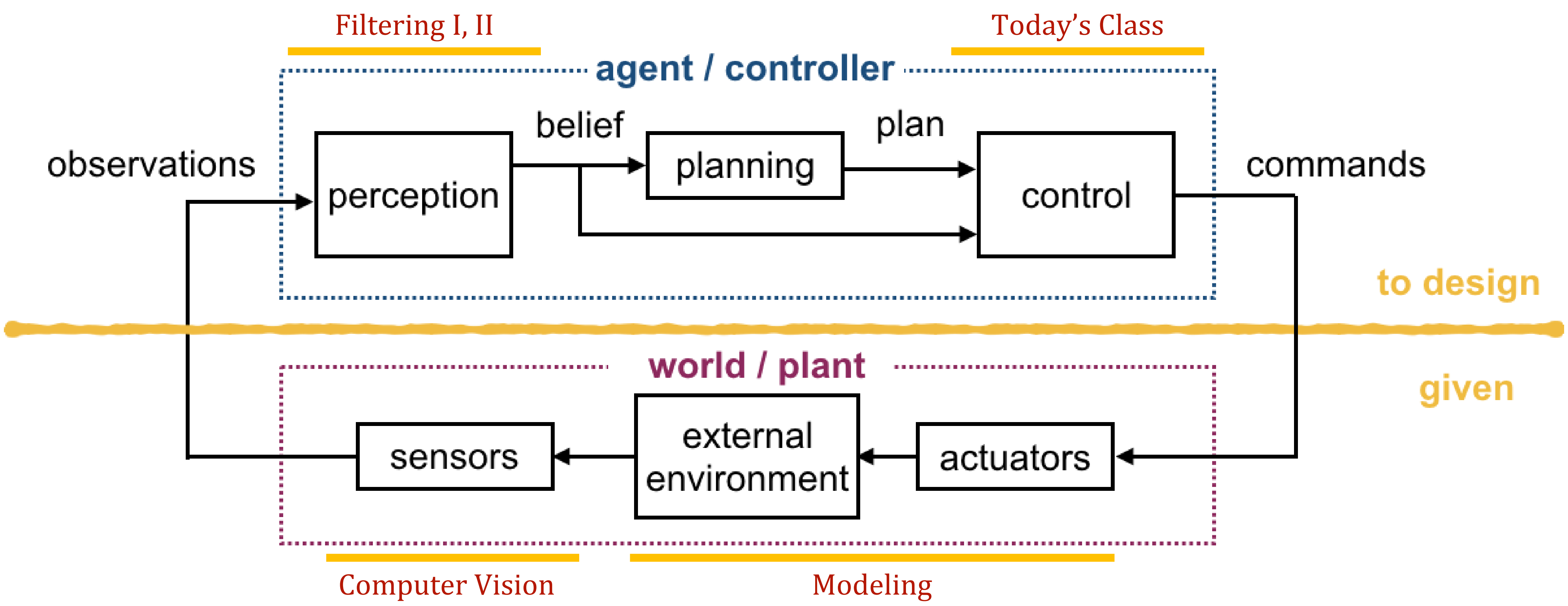
Prerequisites

- Linear Algebra
- Fundamentals of modeling and control
- Notions of Lyapunov control

Credits

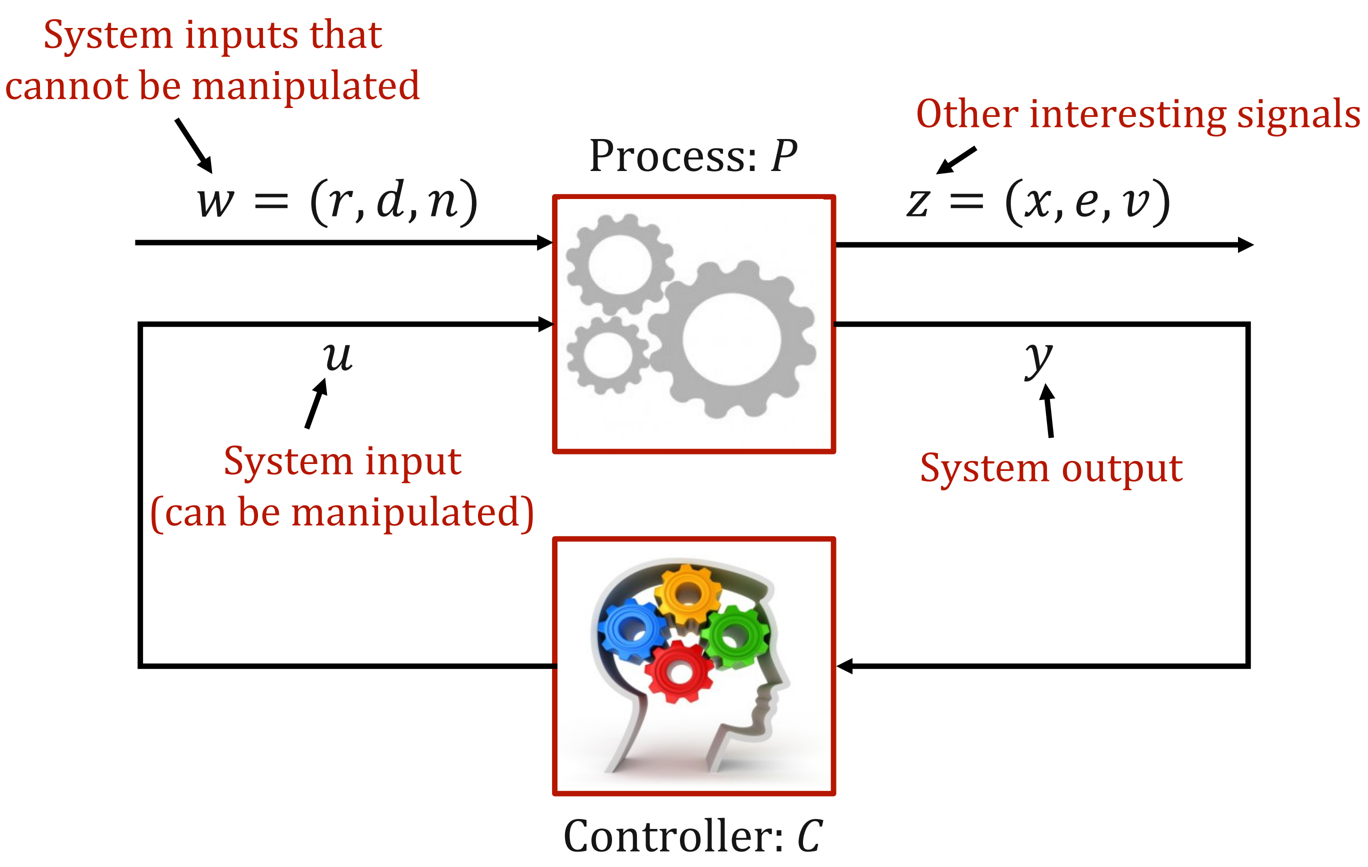
- Jacopo Tani – ETHZ – 1st of November 2017

Big picture



In this module we review existing control approaches for self-driving vehicles
Details will be tackled in dedicated modules

Problem formulation



Given a model of the process P and a reference $r(t)$, find a control law $u(x, t)$ such that:

- the closed loop system is (asymptotically) stable
- progress along the reference path tends to a nominal rate

Review of 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

Review of 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

Pure Pursuit (first paper in 1985)

- Idea: Base control law on fitting a circle that is (a) tangent to car's heading, (b) passes through current position and (c) through the point of $r(t)$ at lock-ahead distance L from the car.

- Design parameter: Look-ahead distance L

- Assume constant driving speed: v_r

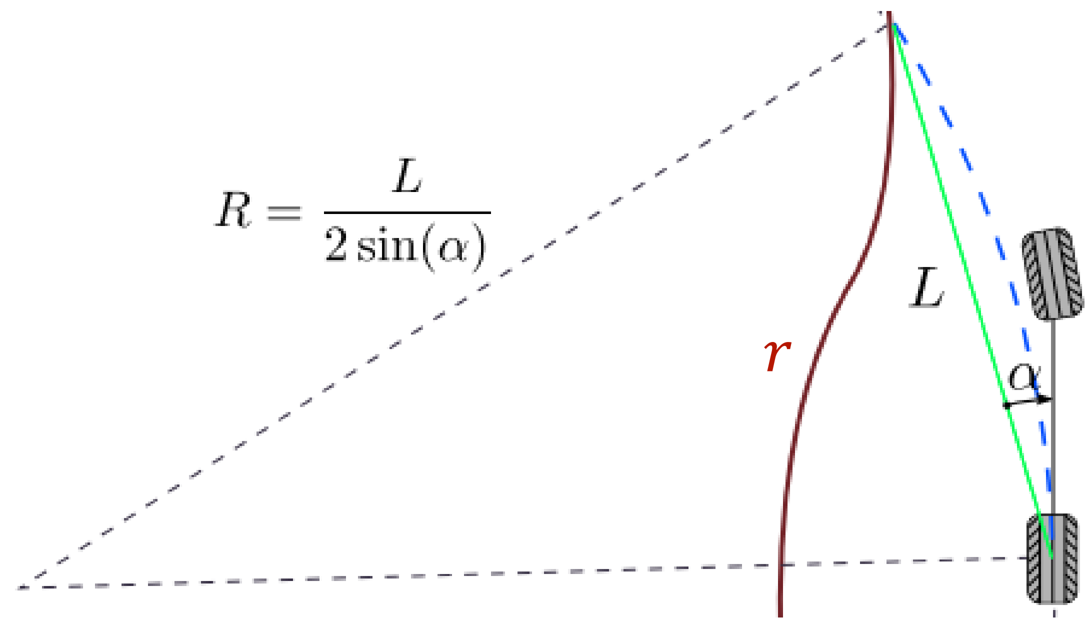
- Given current pose $(x, y, \theta)^T$, always take point more ahead in the path if multiple satisfy:

$$||(x_r, y_r) - (x, y)|| = L$$

- Commanded heading rate:

$$\omega = \frac{2v_r \sin \alpha}{L}$$

$$\alpha = \arctan(\frac{y_r - y}{x_r - x}) - \theta$$



Pure Pursuit: considerations

+ Simple implementation
+ Uses (simple) kinematic model
= For fixed non zero curvature, pure pursuit has small tracking error
= Instead of doing state estimation, α could be measured directly
- Changes in path curvature lead to error accumulation. Ok if driving, not ok if parking
- Heading rate command sensitive to α as speed increases. Fix: scale L with speed

Control Lyapunov design

- Idea: Cast the configuration error on the robot frame and construct a proper Lyapunov function w.r.t. the configuration error.

- Configuration error:

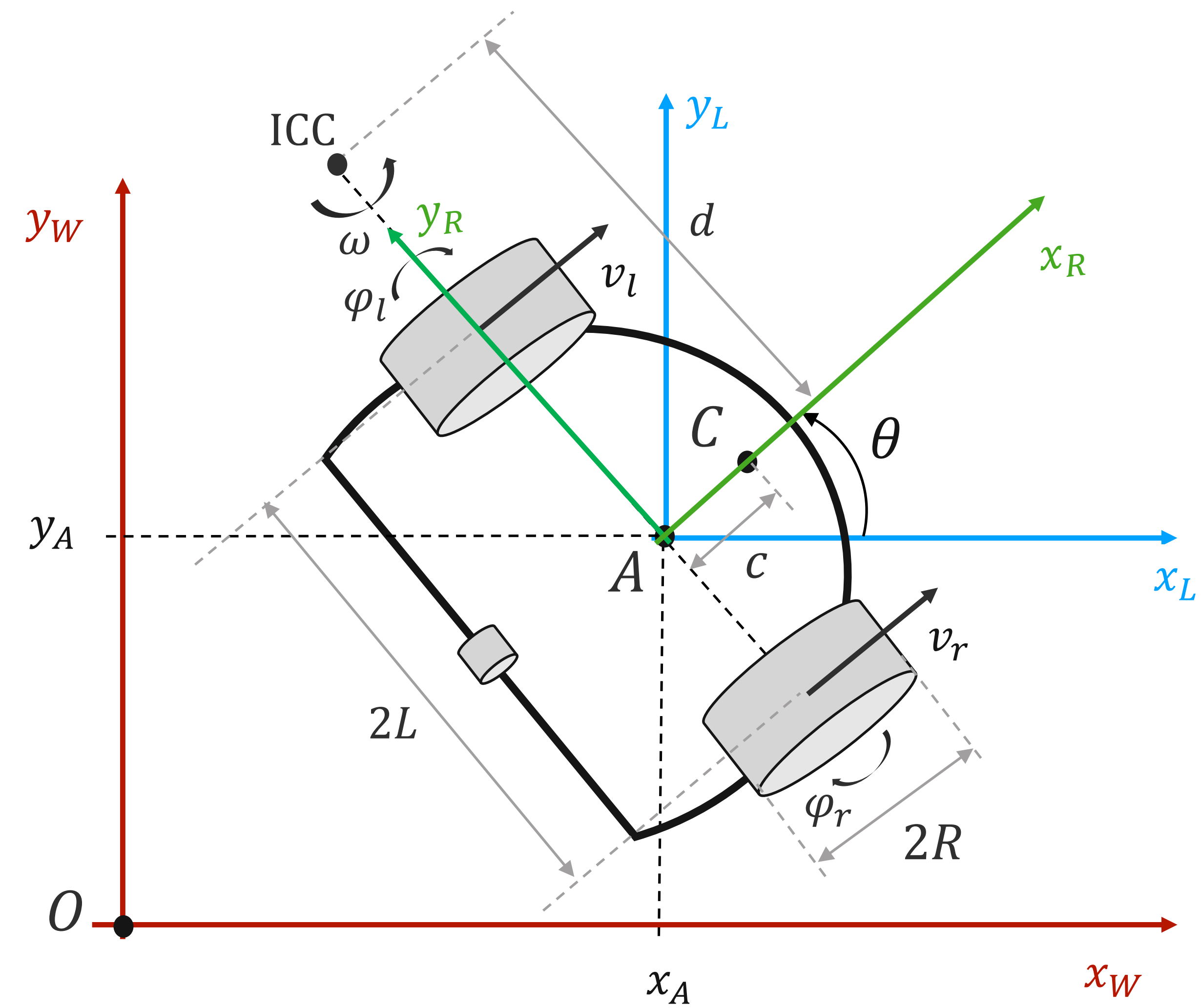
$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ref} - x \\ y_{ref} - y \\ \theta_{ref} - x \end{bmatrix}$$

- Tracking error dynamics:

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = h(x_e, y_e, \theta_e, v_{ref}, \omega_{ref})$$

- Propose candidate control laws

- Write Lyapunov function



Control Lyapunov design: considerations

\+ Local exponential stability can be achieved

\+ Simple kinematic model

\+ Alterations to the basic Lyapunov approach can yield uniform local exponential stability for time varying v_{ref} and ω_{ref}

= Writing Lyapunov function and candidate control laws is somewhat arbitrary

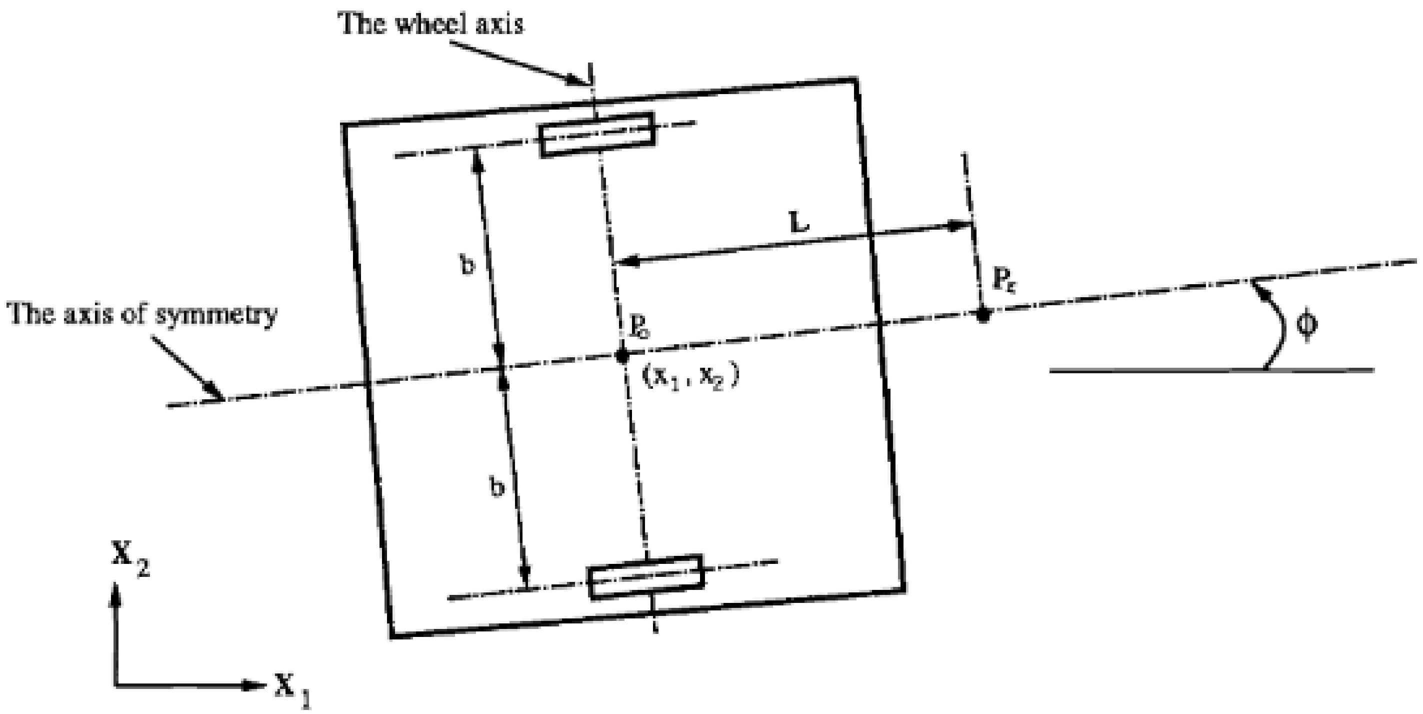
\- For the closed loop controller to be time invariant, v_{ref} and ω_{ref} have to be constant

Output Feedback Linearization

- Idea: Close a first loop around the system with a static nonlinear feedback to input-output linearize it. *When possible*, this enables application of linear systems theory to a nonlinear plant, without approximation (not like with Taylor!)

- Twist: This is possible (with a static linearizing feedback) only if the considered output is a point ahead of the vehicle. This approach is sometimes referred to "look-ahead" control

  - The system is asymptotically stable iff the output point is controlled to move forward
  - The system is unstable if moving backwards (not suited for, e.g., parking maneuvres)



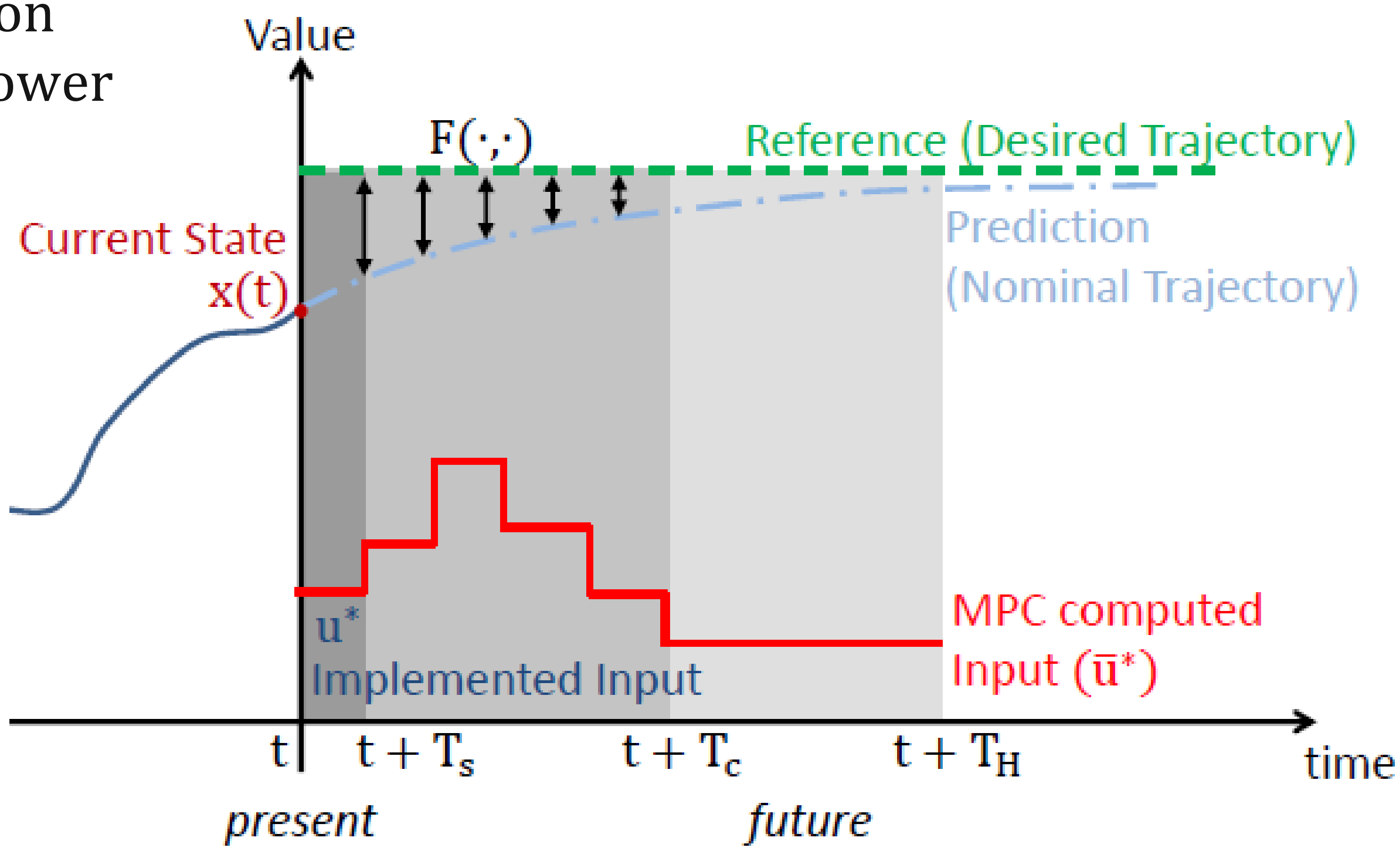
Output Feedback Linearization: considerations

- \+ Obtain an exact input-output linear behavior through inner nonlinear loop
- \+ Can use (e.g.) PID to stabilize
- \+ Particularly suited for when kinematic model is no longer representative (e.g., high speeds)
- = Somewhat intense math to properly understand what is going on (on dedicated module)
- = Based on dynamic model
- \- Works only when moving forward (no parking)

Model Predictive Control (MPC)

- Idea: The essence of MPC is to "optimize forecasts of process behavior". Plant model and current state estimate are used to find a control function (over a time T_C) that optimizes a user defined cost function over a time horizon T_H ($\geq T_C$), subject to constraints. However, the computed input $\bar{u}^*$ is only implemented for a single time step. At the next time step, the problem is solved again. Typically $T_C = T_H$.

- Essence: leverage model prediction capabilities and computational power to deal with complex scenarios

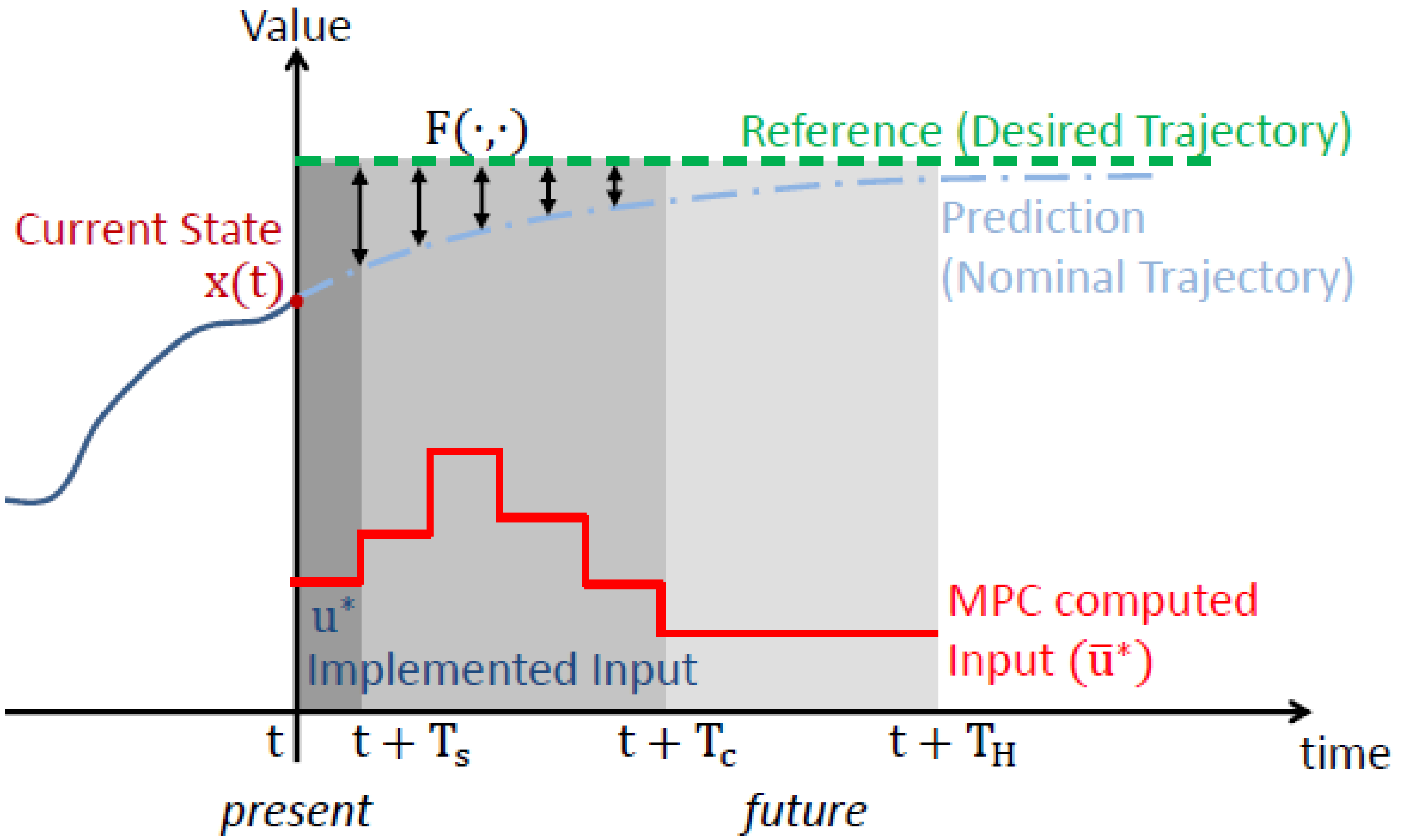


Model Predictive Control (MPC)

- A typical MPC control law looks comes from:

$$u_k(x_{meas}) = \arg \min_{\substack{\text{feasible} \\ \text{space of x,u}}} \left\{ \text{terminal cost} + \sum_{n=k}^{k+N-1} g(\text{distance from reference}, \text{control cost}) \right\}$$

subject to: state estimates
process model
allowable states
allowable inputs



- Hard constraints can be recast as soft (as penalty terms in cost function)

MPC: considerations

\+ Field of research on its own. Many variants and improvements

\+ Extremely adaptable, suited for complex tasks

\- High computational cost leads to look for quadratic formulation (easier to solve)
  Quadratic formulation translates in model linearization about different references: current operating point, reference path or trajectory

\- Numerics play a big role

\- Needs reliable model

Final Remarks

- Different controllers trade off: (a) computational cost, (b) robustness, (c) approximations
- "Best" controller is a function of reference trajectory as well as process model
- All require a model

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

Next Steps

- In Duckietown we don't have a good model as parameters need to be identified

- What kind of control can we do without a model? PID (next module)

- If we had a model, what controller would you use?

- Pure pursuit: because it is really simple
- Feedback linearization: because we love linear systems
- MPC: because it's the end game solution (might be too much for the Raspberry Pi though)