MULTIVARIABLE CONTROL AND COORDINATION SYSTEMS (EE-477)

Case study:

Path tracking for an automated vehicle.

`http://react.epfl.ch`

Denis Gillet, Ezequiel Debada
Lausanne, EPFL, 2018

© 2018-2019
All Rights Reserved.

Exercise 0

Introduction and system model.

0.1 Introduction

With the case study of the course *Multivariable Control and Coordination Systems*, we aim at putting into practice the concepts and methodologies presented in the lectures, as well as at providing a quick glance at how Matlab-Simulink can be used to simulate dynamical systems and implement and assess control strategies.

Four exercises are proposed, all of them somewhat related to some of the control challenges connected to self-driving cars technology. Specifically, we will address the problems of path tracking and trajectory and path planning problems. Path tracking consists in controlling the vehicle so that it accurately tracks a predefined path of interest. Trajectory planning deals with the task of planning, in a certain temporary horizon, a sequence of control actions that will make the vehicle avoid collision and track the desired path. Similarly, path planning deals with the calculation of the path to be followed without considering any temporary information. The mentioned problems are complex in reality and we will tackle here simplified versions of them.

We will begin by implementing and assessing the linear-model-based control and observation techniques described in the course. Namely we will discuss regarding the usefulness and limitations of linear-model-based control techniques, LQR control design and implementation, and observers. Then, we will make use of the dynamic programming concept and the Bellman's principle of optimality to address the trajectory planning problem through a graph-search algorithm. Finally, path planning will be tackled with a navigation function-based technique.

Files In every exercise session you will be provided with the description of the exercise, a Matlab script `exercise#_*.m` (which does not need to be modified), a Matlab class `utilities.m` (common to all exercises) gathering some auxiliary functions the exercise scripts will use, a Matlab-class named `ex#.m` which will include the functions that you have to complete, as well as simulink files which also have to be completed.

The fact that we are using a class to gather the functions you have to complete is simply for clarity proposes. The methods of such a class will then be called from the provided exercise script.

Evaluation At the end of the course we expect you to submit a zip file (named `Surname-Name.zip`) with a total of 8 items:

- A pdf document reporting the results and discussions of all the proposed exercises.
- Four `ex#.m` files, one per session, containing your solutions to the proposed exercises.
- Three Simulink files with the block diagrams used to run the experiments.

Even though we encourage you to work in small groups (preferably up to 3 people) and to discuss ideas to overcome the difficulties you might find, the report **must** be written individually. Please keep in mind that assessing simulation results is crucial and a bad/good discussion can invalidate/validate good/bad results. Important: although the matlab and simulink files are requested, **the evaluation of the exercises will be solely based on the pdf report**.

Meaning that references in the pdf document such as "as can be seen in the attached simulink file" or others of the sort must be avoided. Do not expect us exploring the Matlab files to find the solutions. Everything you want us to take into account when grading the exercises must be included in the report. The only propose of requesting the Matlab files is guaranteeing the uniqueness of your solution.

Please notice that this is the first iteration of this case study as it is presented this year. Therefore, it exists the probability of finding bugs, typos, or inconsistencies in the code. We kindly ask you to point out any flaw you find so that we can improve and update the material of the case study over the year, converging to a more robust version of it.

0.2 The model

To tackle the control/planning problems here-above mentioned, we are going to make use of the so-called kinematic bicycle model of a car, described by the next set of differential equations

$$\dot{p}_x = v\cos(\theta) \tag{1a}$$
$$\dot{p}_y = v\sin(\theta) \tag{1b}$$
$$\dot{\theta} = \frac{v}{L}\tan(\phi) \tag{1c}$$
$$\dot{v} = \sigma_a(\alpha - v) \tag{1d}$$
$$\dot{\phi} = \sigma_s(\beta - \phi) \tag{1e}$$

where p_x, p_x, θ represent the absolute position and orientation of the vehicle w.r.t a fixed frame of reference, v denotes the *longitudinal* speed, L denotes the car length, ϕ shows the steering wheel angle, α and β show the speed and the steering wheel angle reference (control inputs), and σ_a and σ_s represent the dynamic of the actuators in charge of tracking longitudinal speed and steering wheel angle references.

A common practice to perform path tracking consists in rewriting the system in such a way that the position is expressed w.r.t. the path that should be followed (as it is illustrated in Fig 1). Technique which reveals itself very useful when constant-curvature paths (or paths resulting from the concatenation of constant-curvature segments) are followed. The technique consists in expressing the position of the vehicle through the curvilinear coordinate s, the *lateral deviation w.r.t. the path of reference* d, as well as the *heading error* θ_e (see Fig. 1 for a better understanding of the introduced states).

The car model can then be reformulated as follows:

$$\dot{s} = \frac{v\cos(\theta_e)}{1 - d\kappa(s)} \tag{2a}$$
$$\dot{d} = v\sin(\theta_e) \tag{2b}$$
$$\dot{\theta_e} = \frac{v}{L}\tan(\phi) - \kappa(s)\dot{s} \tag{2c}$$
$$\dot{v} = \sigma_a(\alpha - v) \tag{2d}$$
$$\dot{\phi} = \sigma_s(\beta - \phi) \tag{2e}$$

where $\kappa(s)$ shows the path curvature w.r.t. the path coordinate s. Looking at equations 2a-2e it becomes evident why constant-curvature paths are desired. Mainly because of the fact that the

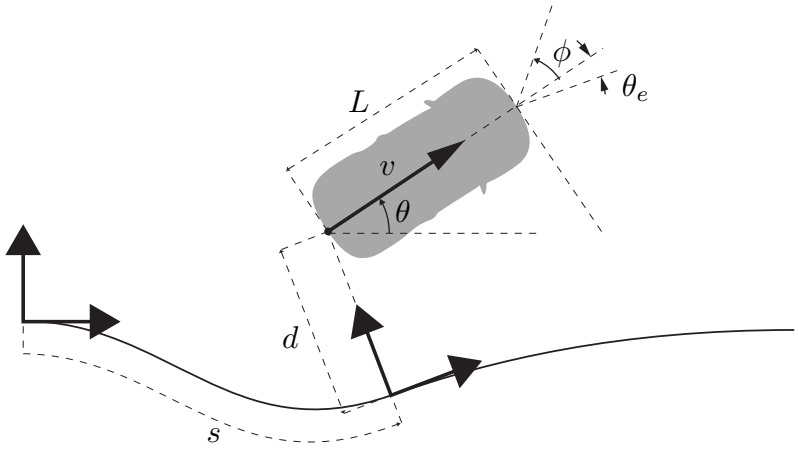

Figure 1: Coordinate and variable representation of the path tracking problem

tracking problem becomes much more easy to handle. For the sake of simplicity, we are going to assume that the path has a constant curvature. Moreover, we will consider the state-space vector, and control inputs to be

$$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5] = [s, d, \theta_e, v, \phi] \tag{3}$$

$$\mathbf{u} = [u_1, u_2] = [\alpha, \beta]\,. \tag{4}$$

which would then allow us to rewrite the model in the form $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$ as

$$\dot{x}_1 = \frac{x_4 \cos(x_3)}{1 - x_2\kappa} \tag{5}$$

$$\dot{x}_2 = x_4 \sin(x_3) \tag{6}$$

$$\dot{x}_3 = \frac{x_4}{L}\tan(x_5) - \frac{\kappa x_4 \cos(x_3)}{1 - x_2\kappa} \tag{7}$$

$$\dot{x}_4 = \sigma_a(u_1 - x_4) \tag{8}$$

$$\dot{x}_5 = \sigma_s(u_2 - x_5) \tag{9}$$

which is the continuous non-linear model of the system we will use to implement linear-model-based control and observation techniques in the following exercises.