Exercise 3

Trajectory Optimization

Due date: March 23, 2018
Submit by Brightspace before midnight.

Course: SC42090 Robot Motion Planning and Control, TU Delft
Contact: Linda van der Spaa, email: L.F.vanderspaa@tudelft.nl
Office hours: Wiener hall C-3-190, Thursday 15/3 10:00 – 11:00, Thursday 22/3 13:00 – 14:00 (!)

Acknowledgements

This exercise is derived from a similar example by Embotech Gmbh.

1 Introduction

The goal of this exercise is to optimize the path of a robotic vehicle in an environment with obstacles. In order to achieve this task, nonlinear programming (NLP) will be applied. The algorithm is described in the lecture slides.

The FORCES Pro solver [Embotech, 2017] is used for solving the nonlinear optimisation problem. In order to be able to use this service, you need a licence. These have been requested. You should get an email this week. If you do not get an email from Embotech, you may have to request an academic license online. For the field of application specify [COURSE SC42090 TU DELFT].

The exercise consists of tasks that you have to do, but do not require a written answer, and questions that do require a written answer. Note: This time, no validation method is provided for the tasks. The questions to be answered are indicated by the number of points a correct answer will yield between parentheses. A maximum number of 20 points can be earned by completing this exercise, 10 points extra for the bonus part. To complete this exercise, hand in your answers to the questions using at most 2 pages (A4, 11pt). For the bonus exercise you are allowed two extra pages. You may use additional pages that contain only supporting figures.

2 Matlab Implementation

In order to complete the implementation, you will have to fill out blanks (indicated by the TODO symbol) in the provided Matlab code.

In this exercise we illustrate the high-level user interface with an optimal trajectory generation problem. In particular, we use a vehicle model described by a set of ordinary differential equations

(ODEs):

$$\dot{x} = v\cos(\theta) \tag{1a}$$
$$\dot{y} = v\sin(\theta) \tag{1b}$$
$$\dot{v} = F/m \tag{1c}$$
$$\dot{\theta} = (vs)/L. \tag{1d}$$

The model consists of four differential states: x and y are the Cartesian coordinates of the car, v is the linear velocity and θ is the heading angle. Next, there are two control inputs to the model: the acceleration force F and the steering torque s. The two parameters are the car mass $m = 0.9$kg and the wheel base which we assume to be $L = 0.12$m.

(2pt) What physical system do the dynamic equations (1) represent? And how does it compare to the standard definition?

The trajectory of the vehicle will be defined as an NLP. First, we define stage variable z by stacking the input and differential state variables

$$z := \begin{bmatrix} F & s & x & y & v & \theta \end{bmatrix}^T.$$

Exercise 3.1: Objective

In this exercise the cost function is the same for all stages. We want to maximize progress in the y direction, with quadratic penalties on the inputs F and s, i.e.

$$f(z) = -ay + b_1F^2 + b_2s^2 \tag{2}$$

Task: Choose the weighting factors $a = 100, b_1 = 0.1, b_2 = 0.01$ and implement the cost function in place of the `TODO` symbol in the Matlab section marked `(Ex. 3.1)`.

Exercise 3.2: Matrix equality constraints

The matrix equality constraints in this exercise represent only the discretized dynamic equations of the vehicle using an explicit Runge-Kutta integrator of order 4. The vehicle dynamics defined in (1) are represented by a function `continuous_dynamics` and the NLP constraint function as the function `model.eq`. Note that the function RK4 is included in the FORCES Pro client software.

Task: Fill out the system dynamic equations (1) in place of the `TODO` symbols in the Matlab section marked `(Ex. 3.2)`.

Exercise 3.3: Inequality constraints

The maneuver is subjected to a set of constraints, involving both the state and input bounds

$$\begin{array}{rl} -5 \leq F \leq 5 & \text{N} \\ -1 \leq s \leq 1 & \text{Nm} \\ -3 \leq x \leq 0 & \text{m} \\ 0 \leq y \leq 3 & \text{m} \\ 0 \leq v \leq 2 & \text{m/s} \\ 0 \leq \theta \leq \pi & \text{rad} \end{array}$$

as well the nonlinear nonconvex constraints for avoiding collision with ellipsoid obstacles

$$\begin{array}{ll} 1\ \text{m}^2 & \leq x^2 + y^2 \leq \quad 9\ \text{m}^2 \\ 1\ \text{m}^2 & \leq (x+2)^2 + (y-2.5)^2. \end{array}$$

Task: Implement the constraints by filling out the blanks marked with the `TODO` symbols in the Matlab section marked `(Ex. 3.3)`.

Exercise 3.4: Initial and final conditions

The goal of the maneuver is to steer the vehicle from a set of initial conditions

$$x_{\text{init}} = -2.5\text{m}, \qquad y_{\text{init}} = 0\text{m}, \qquad v_{\text{init}} = 0\text{m/s}, \qquad \theta_{\text{init}} = \frac{3}{4}\pi\text{rad}$$

to another point in the state-space subjected to the final conditions

$$v_{\text{init}} = 0\text{m/s}, \qquad \theta_{\text{init}} = 0\text{rad}.$$

Additional final conditions on x and y position can be specified.

Task: Implement the initial and final conditions by filling out the blanks marked with the `TODO` symbols in the Matlab section marked `(Ex. 3.4)`.

3 Results and Evaluation

Now all blanks are filled, the code can be run and the results will be shown in three different windows: one showing the path of the vehicle in the workspace, the other two showing respectively the velocity v and heading angle θ of the vehicle and the control inputs F and s.

Exercise 3.5: Optimization settings

(4pt) Try different weighting factors in the cost function you implemented in Exercise 3.1 and discuss the effects on the performance.

(2pt) In line `47` the control horizon N is defined. Try different values and describe how N should be chosen.

Exercise 3.6: Changing task and environment

(6pt) Design your own new environment with at least two obstacles which must be avoided. Add a figure.

(6pt) Define a navigation goal different from maximizing the vertical displacement. Redefine your cost function and initial and final state constraints accordingly. Show and explain your equations.

Discuss your results for this custom scenario you created.

Bonus Exercise 3.7: Differential Drive Robot

(3pt) So far the collision avoidance has assumed point size of the vehicle. Describe (you do not need to implement) how the real size of the vehicle could be taken into account when planning a trajectory avoiding obstacles.

(7pt) Add a trailer to the vehicle, see Fig. 1. Note that the magnitude of the angle ϕ of the trailer with respect to the vehicle may not be larger than $\phi_{\max} = 45°$. Show and explain your equations. Discuss your results for this new dynamical system.

References

[Embotech, 2017] Embotech (2017). Forces pro. `https://www.embotech.com/FORCES-Pro`.

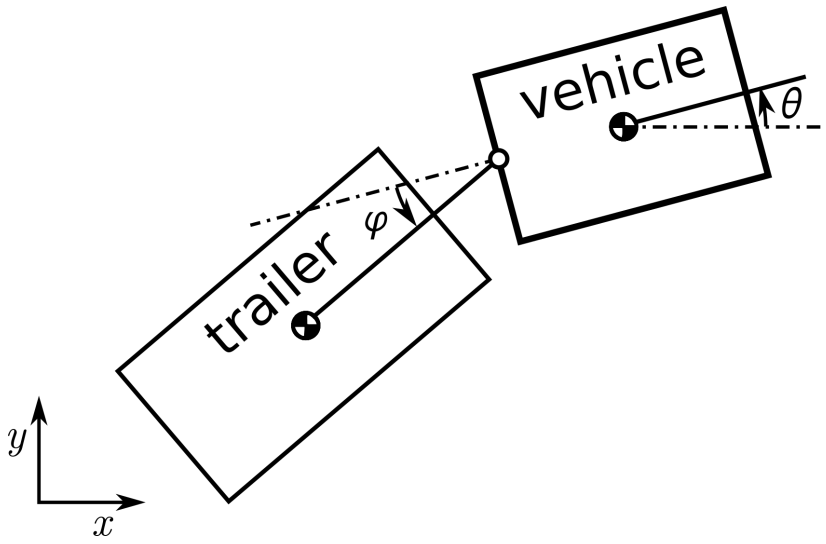

Figure 1: Vehicle with trailer