Optimal Control & MPC Homework

Vehicle Dynamics, Planning and Control of Robotic Cars

Mattia Piccinini &
Gastone Pietro Papini Rosati

EIT Digital
University of Trento

November 19th, 2021

Assignment

This homework is about motion tracking with optimal control and MPC (model predictive control).
The following exercises are **not mandatory**, but extra points on the final mark will be given if you complete (at least part of) them.

1 Problem Description

The main objective consists of solving optimal control problems for motion tracking, with certain vehicle models.

1.1 Optimal control theory

1.1.1 Continuous-time formulation

A generic continuous-time optimal control problem (OCP) can be formulated as:

$$\min_{\boldsymbol{u} \in \mathcal{U}} \quad \mathcal{J} = \boldsymbol{B_i}\big(\boldsymbol{x}(t_0) - \boldsymbol{x}_i\big)^2 + \boldsymbol{B_f}\big(\boldsymbol{x}(t_f) - \boldsymbol{x}_f\big)^2 + \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{x}(t), \boldsymbol{u}(t))\mathrm{d}t \tag{1a}$$

$$\text{s.t.} \quad \begin{cases} \dfrac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \forall t \in [t_0, t_f], & \text{(1b)} \\ \boldsymbol{b}\big(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f)\big) = 0, & \text{(1c)} \\ \boldsymbol{c}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \geq 0, \quad \forall t \in [t_0, t_f] & \text{(1d)} \end{cases}$$

The cost function $\mathcal{J}$ in (1a) consists of two main terms, which are often referred to as **Mayer** and **Lagrange** terms.
The Mayer term is the summation in (1a), and it is often used to impose *soft* initial (for time $t = t_0$) and final (for $t = t_f$) conditions for the model states $\boldsymbol{x} \in \mathbb{R}_{n_x,1}$, via the tunable weights $\boldsymbol{B_i}$ and $\boldsymbol{B_f} \in \mathbb{R}_{1,n_x}$. Setting some of the initial and/or final conditions with the Mayer term does not guarantee that those conditions will be exactly satisfied by the solver, since only the quadratic deviations from the target initial and final values $\{\boldsymbol{x}_i, \boldsymbol{x}_f\}$ are penalized in (1a). As an alternative, strict initial and/or final conditions can be set with the **equality constraints**

(1c), so that they will be exactly satisfied by the solver. In this exercise, the initial conditions are all set with strict equality constraints.
The integral cost in (1a) is the **Lagrange** term, also called running cost or stage cost. The Lagrange term can be employed to impose a cost function covering the entire horizon of the problem (i.e. from time t_0 to t_f). For example, the cases of minimum time/jerk/energy motion planning can be dealt with the Lagrange cost. In this exercise, a motion tracking task is formulated by means of the Lagrange term.
Note that, in general, an OCP might not have both a Lagrange and a Mayer cost. Moreover, it can be proved that, with suitable mathematical manipulations, a Mayer term can be reformulated as a Lagrange term, and viceversa.
The optimization problem (1a) is subject to a **dynamical constraint** (1b), with the states and the controls being $\boldsymbol{x}$ and $\boldsymbol{u}$. In this exercise, (1b) is a dynamical vehicle model.
The **inequality constraints** (1d) can be used to impose path and actuation constraints involving the states and/or the controls.
Note that in this exercise the OCP is formulated in the **time domain**. However, other efficient formulations exist, for example in the space domain (using the curvilinear abscissa of a road as the independent variable for motion planning with optimal control).

1.1.2 Converting to discrete-time

One method to solve generic nonlinear optimal control problems is named ***direct approach***, and it consists in the discretization of the continuous-time OCP. The **discrete-time** version of the OCP of section 1.1.1 is:

$$\min_{\boldsymbol{u} \in \mathcal{U}} \quad \mathcal{J} = \boldsymbol{B_i}\big(\boldsymbol{x}_0 - \boldsymbol{x}_i\big)^2 + \boldsymbol{B_f}\big(\boldsymbol{x}_N - \boldsymbol{x}_f\big)^2 + \sum_{k=0}^{N-1} l(\boldsymbol{x}_k, \boldsymbol{u}_k)\mathrm{d}t \tag{2a}$$

$$\text{s.t.} \quad \begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \quad \forall k \in \{0, 1, \dots, N-1\}, & (2b) \\ \boldsymbol{b}\big(\boldsymbol{x}_0, \boldsymbol{x}_N\big) = 0, & (2c) \\ \boldsymbol{c}\big(\boldsymbol{x}_k, \boldsymbol{u}_k\big) \geq 0, \quad \forall k \in \{0, 1, \dots, N-1\} & (2d) \end{cases}$$

A discretized **mesh** must be defined for the independent variable (in this case time). A **numerical integration** scheme is needed to approximate the dynamics (2b). The inequality constraints (2d) are enforced only on the N mesh points.
Various techniques exist to build the discrete-time version of an OCP. The most used methods are *single shooting*, *multiple shooting* and *direct collocation*.
In single shooting, the optimization variables are only the controls $\boldsymbol{u}_k$, $k \in \{0, 1, \dots, N-1\}$. The system dynamics is integrated forward with the controls $\boldsymbol{u}_k$, for

the entire horizon N, and a final condition is set to match the desired final state $\boldsymbol{x}_f$ (if any). However, the forward integration of the system dynamics may lead to convergence issues if the plant is open loop unstable.
In multiple shooting, the optimization variables are instead both the states $\boldsymbol{x}_k$ and the controls $\boldsymbol{u}_k$. The system dynamics is integrated among each pair of mesh nodes, and interface conditions are set to impose the continuity of the model states at the mesh nodes. Since the integration is carried out in smaller intervals, multiple shooting is usually more suited to deal with unstable and heavily nonlinear plants.
In direct collocation, the optimization variables are again both the states and the controls. However, the system dynamics between two consecutive mesh points is approximated with custom *collocation polynomials*, whose parameters are computed by introducing additional *collocation points* between each pair of mesh nodes. Implicit Runge Kutta schemes (typically Gauss-Legendre or Gauss-Radau schemes) are adopted to numerically integrate the plant dynamics and build the interpolating polynomials.
In this exercise, **direct collocation** is employed to approximate the dynamics $\boldsymbol{f}$ and build the discretized OCP.
The previously formulated discrete-time optimization problem is, in general, a nonlinear programming problem (**NLP**), which can be solved with off-the-shelf tailored solvers. In this exercise, we use **IPOPT** to solve the NLP.
The software tool **CasADi** is used to generate the Jacobians and Hessians (also called sensitivities of the integration scheme) for the solver, with the automatic differentiation (**AD**) technique.

1.2 CasADi

You are going to use CasADi, which is an open source software tool for nonlinear optimization and algorithmic differentiation.
Please go to `https://web.casadi.org` and download the binary files for your architecture.
Quickly read the documentation and check that the tool works correctly in MATLAB.
You are provided with a MATLAB class (named `Dimei`) which enables the definition of an optimal control problem in the CasADi framework.

1.3 Contents of the provided folder

The main file which you need to launch is named `main_OCP.m`.
The `utilities` folder contains auxiliary functions and scripts, which you are required to use e.g. to change the vehicle model, the cost function of the problem, and the post-processing.

2 Exercises

2.1 Exercise 1 – Optimal Control

You are required to answer the following questions:

1. Analyze the optimal control problem (OCP) formulation.
   Open the `VehicleDynamics.m` file, which contains the differential equations of the vehicle model. What type of vehicle model is this? Comment each of the 4 equations, the definition and assumptions for the side slip angles and tire forces.

2. Changing the OCP settings:

   (a) In the `CostFunction.m` file, use `vx_target` = min(t,15). Set a sine law for `Omega_target`, with an amplitude of 5 deg/s and a period of 10 s. Use the weights `w_vx` = 1, `w_Omega` = 1, `w_Fx` = 1e-6, `w_delta` = 1e-4. What is the role of each weight?
   Use a time horizon of 30 s (parameter `T`), and an initial condition for v_x of 0.1 m/s (parameter `X0.vx`).
   What is the expected behavior of the vehicle, with this cost function?
   Using subplots, plot the model states, controls, tire forces, side slip angles. The `vx_target` and `Omega_target` profiles have to be plotted (with a dashed line) together with the corresponding v_x and Ω solutions. Comment the results of the optimal control problem.
   How is the evolution of the longitudinal and lateral tire forces?

   (b) Use the same cost function as in 2(a), but now set the initial condition for v_x to 3 m/s. Compare the results with 2(a), focusing especially on the longitudinal and lateral tire forces.

   (c) What happens if you set the initial condition for v_x to 0 m/s? What could be a possible workaround?

   (d) Keep the same settings of 2(b), but now use the weight `w_Fx` = 1e-4. Comment the results.

   (e) Keep the same settings of 2(b), but now change the bounds for the control `delta_u` to $\pm$ 1° (`ControlBounds` parameter). Comment the results.

   (f) Keep the same settings of 2(b), but now use a time horizon `T` = 10 s. It seems that the tracking of the speed profile gets worse. Why? How can you improve the tracking of the speed profile?

2.2 Exercise 2 – Model Predictive Control

It is now requested to implement a receding horizon model predictive control problem (MPC), starting from the provided optimal control formulation.
Use the following settings for each MPC problem: time horizon `T` = 5 s, number of mesh points `N` = 80, initial condition for v_x = 3 m/s, `vx_target` = 3 + min(t,15), `Omega_target` = same as in 2(a) but with a sine amplitude of 2 deg/s. The other settings (control bounds etc) are the same as in 2(a).

1. Suppose that the vehicle will perfectly track the computed optimal control solutions. Solve again new optimal control problems every 100 ms, updating the initial conditions for the 4 vehicle states (you can assume that the vehicle is moving along the computed optimal trajectory). In this way, solve 30 optimal control problems (MPC steps) – the total simulation time will be $30 \cdot 0.1 = 3$ s –. In the report, provide a screenshot of the MATLAB code that you implemented or write a pseudo code.
   Hint: use a `for` loop to execute the 30 optimal control problems. Before solving a new OCP, change the initial conditions by interpolating forward in time the previous MPC solution (e.g. use a simple linear interpolation, with `interp1`, to get the vehicle states after 100 ms). Also change `t0`, re-execute `ocp.setup(T,N,t0)`, and save the solutions. Do not care about real-time execution (the simulation on your compute will last more than 3 s).

2. Plot the results that you obtain, for all the model states, controls and tire forces (in the time interval [0,3] s). Briefly comment the results.

3. In a real motion planning application, what is, in your opinion, the advantage of solving on-line optimal control problems (with a suitably fast rate), as the vehicle moves? Does the vehicle model used for MPC need to perfectly match the dynamics of the real car? Do you think that it is possible to tolerate small model mismatches with MPC?