Exercises for Lecture Course on Numerical Optimal Control (NOC)
Albert-Ludwigs-Universität Freiburg – Winter Term 2024 / 25

Exercise 6: Optimal Control Formulations

Prof. Dr. Moritz Diehl, Andrea Zanelli, Dimitris Kouzoupis, Florian Messerer, Yizhen Wang

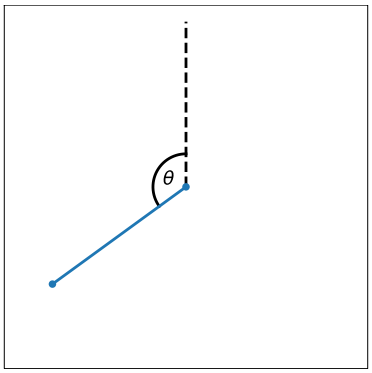

Figure 1: Sketch of the pendulum with its position defined by the angle θ.

In this exercise we will have a look at the special structures of the optimal control formulations discussed in the class. To this end, we will consider the control of a pendulum, as illustrated in Figure 1. The pendulum position is described by the angle θ with corresponding angle velocity w and we can control it by applying a torque τ. This system can be described by idealized dynamics

$$\begin{aligned} \dot{\theta} &= \omega, \\ \dot{\omega} &= \sin(\theta) + \tau, \end{aligned} \tag{1}$$

where for simplicity we are ignoring all units. Defining the state $x = (\theta, \omega)$, with control $u = \tau$, our control aim is to swing up the pendulum to the upright position and balancing it there, corresponding to a state of $x_{\text{ref}} = (0, 0)$. We start with the pendulum hanging down in a rest position, i.e., $\bar{x} = (\pi, 0)$. We can express this as the discrete time optimal control problem

$$\begin{aligned} \min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}} \quad & \sum_{i=0}^{N-1} \left(x_i^\top x_i + 2u_i^2 \right) + 10 x_N^\top x_N \\ \text{s.t.} \quad & x_0 = \bar{x}_0, \\ & x_{i+1} = F(x_i, u_i), \quad i = 0, ..., N-1, \end{aligned} \tag{2}$$

where $x_0, \dots, x_N \in \mathbb{R}^2$, $u_0, \dots, u_{N-1} \in \mathbb{R}$, and $F(x, u)$ are the discretized dynamics obtained by applying the explicit RK4 integration scheme to (1). The objective expresses our desire to bring the state x close to the origin, while using not too much control effort, as quantified by deviations of the control variable from 0. We will consider different reformulations of problem (2) and analyze their structure.

1. **Sequential approach:** The state variables in (2) can be eliminated by means of a forward simulation such that a more compact optimization problem is obtained,

$$\min_{U} \quad \Phi(U), \tag{3}$$

with $U = (u_0, \dots, u_{N-1})$. Using CasADi, formulate $\Phi(U)$ and set up problem (3), where $F(x_i, u_i)$ is an RK4 integrator with 10 intermediate RK4 steps over a total integration time

of $h = 0.1$. Use $N = 50$ for the discrete time horizon (such that the continuous time horizon is $T = 5$) and $\bar{x}_0 = (\pi,\, 0)$ as initial state. Solve the problem using IPOPT and plot the state and control trajectories in two separate plots. Use $u_i = 0.1,\ i = 0, ..., N-1$, as initial guess. Before solving the problem, compute the Hessian of $\Phi(u)$ for some non-zero u and plot its sparsity pattern using the MATLAB command `spy` / Python function `matplotlib.pyplot.spy` (often aliased as `plt.spy`).

Hint: You can use `rk4step.m` / `rk4step.py` *provided with this exercise to do one step of the RK4 scheme.*

2. **Simultaneous approach**: If both states and controls are kept as decision variables, a larger problem is obtained with a special sparsity structure that can be exploited by tailored algorithms.

   (a) Using CasADi and the provided template `simultaneous.m` / `simultaneous.py`, formulate problem (2) without eliminating the state variables, solve it with IPOPT and plot the state and control trajectories. Make sure that you obtain the same result as with the sequential approach. Eliminate the initial state, by fixing it to $x_0 = \bar{x}_0$ and use $u_i = 0.1,\ i = 0, ..., N-1$ and $x_i = (0.1, 0.1),\ i = 1, ..., N$ as initial guess. Before solving the problem compute the Hessian of the Lagrangian and plot its sparsity pattern using the MATLAB command `spy` / Python function `matplotlib.pyplot.spy` (often aliased as `plt.spy`). In order to highlight the block-banded structure of the Hessian, reorder the variables as $[u_0,\, \lambda_1,\, x_1,\, u_1, \lambda_2,\, ...\,]$. For this last part, you can set the equality multipliers to any arbitrary non-zero value since we are only interested in the sparsity pattern here.

   (b) Implement now your own full-step Newton method and use it to solve problem (2). Initialize both primal and dual variables to 0.1. Terminate the algorithm when the 2-norm of the right-hand-side is less than 10^{-6}. Plot the resulting trajectories and make sure you obtain the same results as with the previous implementations.

   Hint: you already have the Lagrangian of the problem from a) which can be used to compute the right-hand-side of the linear system associated with the Newton steps. Analogously, the matrix coefficient of the linear system can be obtained by computing the Hessian of the Lagrangian.

   (c) Make a copy of your script from a) and modify the formulation in order to include box constraints on the inputs:

$$-1 \leq u_i \leq 1,\ \forall i = 0, ...,\ N-1. \tag{4}$$

   Solve the modified problem with IPOPT using the same initial guess as before and plot the obtained trajectories.

   (d) [**Bonus**] Make a copy of your script from b) and modify the formulation by including logarithmic barrier terms in the cost that approximate the constraints introduced in c):

$$\sum_{i=0}^{N-1} \left(\|x_i\|_2^2 + 2u_i^2 - \rho \left(\log(-u_i + 1) + \log(u_i + 1)\right)\right) + 10\, \|x_N\|_2^2\,, \tag{5}$$

   where $\rho = 0.01$ is a fixed parameter. Solve the formulation with your implementation of the Newton method. In order to make sure that the arguments of the logarithms in the cost are always positive, implement a line search: starting from a full step, check whether any of the constraints are violated and half the step size until the arguments of the logarithms are positive. Initialize both primal and dual variables to 0.1.