Assignment 02: Compute the viability kernel

Elisa Alboni[*] - elisa.alboni@unitn.it

November 2021

1 Description

The goals of this assignment are:

- implementing and solving a sequence of optimal control problems to determine a control-invariant set for a single pendulum, taking into account state and input bounds;
- discussing about the shape of the computed set;
- discussing the application of the obtained set for solving Model Predictive Control (MPC) problems.

2 Submission procedure

You are encouraged to work on the assignments in groups of 2 people. **If you have a good reason to work alone, then you can do it, but this has to be previously validated by one of the instructors**. Groups of more than 2 people are not allowed. The mark of each assignment contributes to 10% of you final mark for the class (i.e. 3 points out of 30).

When you are done with the assignment, please submit a single compressed file (e.g., zip). **The file name should contain the surnames of the group members**, and it must contain:

- A pdf file with the answers to the questions, the **names and ID number** of the group members; you are encouraged to include plots and/or numerical values obtained through simulations to support your answers. **This pdf does not need to be long. One or two pages of text should be enough to answer the questions. You can then add other pages for plots and tables.**
- The complete orc folder containing all the python code that you have developed.

If you are working in a group (i.e., 2 people) only one of you has to submit.

Submitting the pdf file without the code is not allowed and would result in zero points. Your code should be consistent with your answers (i.e. it should be possible to produce the results that motivated your answers using the code that you submitted). If your code does not even run, then your mark will be zero, so make sure to submit a correct code.

[*]Optimization and Learning for Robot Control, Industrial Engineering Department, University of Trento.

3 Computing control invariant sets — Single pendulum with state and input bounds

The viability kernel is the largest set of feasible states, starting from which the system can always remain inside the set, therefore without violating the constraints. The viability kernel is the largest control invariant set, and its computation has been the subject of many research papers. While for linear systems there exist tools for computing this set, for the case of nonlinear dynamics and/or nonlinear constraints, computing the viability kernel remains an open problem.

The system under analysis is a single pendulum, a single swinging link connected to a fixed base through a planar revolute joint. The pendulum dynamics is modelled by the following equations:

$$\begin{cases} \dot{q} &= v \\ \dot{v} &= g\sin(q) + u \end{cases} \tag{1}$$

where $x = [q, v]$ is the state vector and u is the control. We assume unitary mass and length. The state and input constraints are set to constant values representing the limit positions, q_{min} and q_{max}, velocities, v_{min} and v_{max}, and torques, τ_{min} and τ_{max}. Despite its low dimensionality, the pendulum is a nonlinear system, which therefore doesn't enjoy any simple way to compute its viability kernel.

For nonlinear systems, one way in which we can approximate the viability kernel is by solving a sequence of optimal control problems (OCP). In particular, we can determine if a state x_{sample} belongs to the viability kernel if there exists a solution a specific OCP. By randomly sampling the set of feasible states, we can therefore determine which of these states belong to the viability kernel, and which do not. The OCP that needs to be solved is:

$$\begin{aligned} \text{minimize} \quad & J = 1 \\ \text{subject to} \quad & x_{k+1} = f(x_k, u_k) \quad \text{for} \quad k = 0, \dots, N-1 \\ & u_k \in U \quad \text{for} \quad k = 0, \dots, N-1 \\ & x_k \in X \quad \text{for} \quad k = 0, \dots, N \\ & x_0 = x_{sample} \\ & x_N = x_{N-1} \end{aligned}$$

where $U = [\tau_{min}, \tau_{max}]$ and $X = [q_{min}, q_{max}] \times [v_{min}, v_{max}]$. The OCP presents a constant cost as we exploit the OCP to check whether a trajectory starting from $x_0 = x_{sample}$ that satisfies the constraints exists. The final constraint, $x_N = x_{N-1}$ guarantees that the final state of the trajectory is an equilibrium. Therefore, it ensures that there exists a control enabling the system to remain in this state, preventing any violation of constraints in the future. In practice, there are two strategies to implement this constraint:

1. Constrain the final velocity to 0, $\dot{q}_N = 0$, (in practice, it is suggested to replace the equality constraint with a couple of inequality constraints ensuring that $\|\dot{q}_N\| < \epsilon_{thr}$ where ϵ_{thr} is a small value (for example $\epsilon_{thr} = 10^{-7}$);
2. Add a quadratic penalty on the final velocity, $J(x_N) = \frac{1}{2} w_v (\dot{q}_N)^2$, where w_v is the cost weight.

4 Tests

All the files required for this assignment are located in the folder:

`orc/A2/`

The joint position, velocity and control bounds values are reported in the configuration file `A2_conf` (`lowerPositionLimit`, `upperPositionLimit`, `lowerVelocityLimit`, `upperVelocityLimit`, `lowerControlBound`, and `upperControlBound`). Please do not change any parameter in the configuration file `A2_conf.py` if not explicitly asked, except for `activate_equilibrium_ineq` and `weight_eq_dq`. You need:

- to implement the Jacobian of the dynamics (in `ode.py`);
- to enforce the bound on the control `bnds` in the minimization (in `single_shooting_problem.py`); look at the documentation of the `minimize` function of scipy to see how you can add simple bounds on the decision variables of the problem;
- to use the inequality constraints to represent the joint position and velocity limits (in `inequality_constraints.py`);
- to use the inequality constraint on the final velocity (in `inequality_constraints.py`) or the penalty on the final velocity (in `cost_functions.py`)
- (suggested) to use a cost to regularize the control (in `cost_functions.py`)

Try to answer the following questions:

1. Implement a sampling strategy to select x_{sample} (for instance, uniform random sampling or grid-based sampling) and generate a plot of the viability kernel based on the results of the OCPs. What can you say about the states visited with the optimal control trajectory? Can they be used to augment the dataset in the plot?
2. Why do you think the viability kernel presents this particular shape? How may it be affected if the absolute value of the control bounds were increased (for example $5g$)? Do you expect the boundaries of the viability kernel to present the same trend regardless of the values/type of the limits on the joint position and velocity? Motivate the answer with an example.
3. Can you exploit the obtained data to address the problem of recursive feasibility in an MPC problem? Present and discuss your ideas.