Model Predictive Control and Reinforcement Learning

Exercise Sheet 2: Introduction to `acados` and `leap-c` in `Python`

Joschka Boedecker, Moritz Diehl, Leonard Fichtner, Jonathan Frey, Andrea Ghezzi, Jasper Hoffmann

Exercise 1: Getting started with `acados`

In this exercise, you will implement a simple MPC controller for the Cart Pole environment (aka inverted pendulum on a cart), see Figure 1. The goal is to implement a controller that balances the pole in the closed-loop simulation provided in `example_closed_loop.py`.

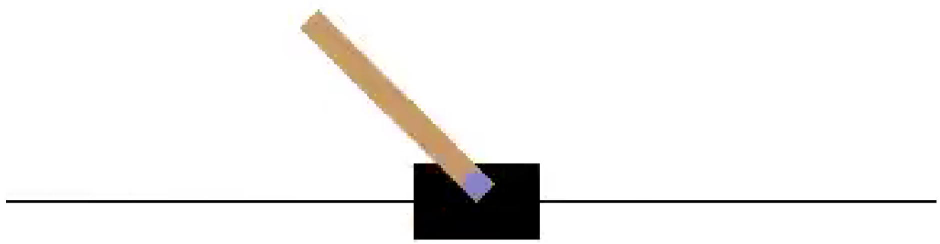

Figure 1: An illustration of Cart Pole while swinging up.

1.1 Familiarize yourself with the `CartPoleEnv` environment, e.g., by reading the documentation of the class.

1.2 Complete the controller formulation in the function `setup_ocp` provided in `setup_ocp.py`. For a given initial state s, this function formulates a simple MPC controller described by the problem formulation

$$\begin{aligned} \min_{x_0,u_0,x_1,\dots,u_{N-1},x_N} \quad & \frac{1}{N}\sum_{k=0}^{N-1} l(x_k,u_k) + \overline{V}(x_N) \\ \text{s.t.} \quad & x_0 = s \\ & x_{k+1} = f^{\mathrm{dyn}}(x_k,u_k), \; k = 0,\dots,N-1 \\ & \underline{u} \le u_k \le \overline{u}, \; k = 0,\dots,N-1, \end{aligned}$$

where $f^{\mathrm{dyn}}(x_k,u_k)$ are the dynamics of the Cart Pole system, given by one step of an RK4 integrator, $\underline{u}$ and $\overline{u}$ are the actuator limits, and the stage costs and terminal cost are given by

$$l(x_k,u_k) = \frac{1}{2}|| \begin{pmatrix} x_k \\ u_k \end{pmatrix} - \begin{pmatrix} x^{\mathrm{ref}} \\ u^{\mathrm{ref}} \end{pmatrix} ||_W^2$$

and

$$\overline{V}(x_N) = \frac{1}{2}||x_N - x_{\mathrm{e}}^{\mathrm{ref}}||_{W_{\mathrm{e}}}^2 .$$

Set the values of $x^{\mathrm{ref}}, u^{\mathrm{ref}}, x_{\mathrm{e}}^{\mathrm{ref}}, W, W_{\mathrm{e}}$ in a way that the controller successfully balances the pole. This will be indicated by messages printed by `example_closed_loop.py`. This will also generate plots and a rendered video that might be helpful.

Model Predictive Control and Reinforcement Learning

Exercise Sheet 2: Introduction to acados and leap-c in Python

Joschka Boedecker, Moritz Diehl, Leonard Fichtner, Jonathan Frey, Andrea Ghezzi, Jasper Hoffmann

Exercise 2: Parametric sensitivities

In this exercise, we show how to extend the `acados` formulation from the previous exercise step-by-step, to arrive at a `leap-c` controller that can be used to train parameters with one of the `leap-c` algorithms and `PyTorch` in general. As an example, we will use this formulation to solve MPC problems for varying values for the reference of the angle θ and compute their sensitivities in `example_diffMPC.py`.

2.1 Make yourself familiar with the `AcadosParameterManager` class.

2.2 The goal now is to make a modified version of `setup_ocp`, in which the parameter manager class is used to introduce new parameters into the acados ocp formulation.
At the bottom of `setup_ocp_with_manager.py` you will find a method `create_cartpole_params` which returns a list of `AcadosParameters` to be passed to the `AcadosParameterManager` instance at creation. As an example, it already contains instances meant for the weight matrix. Extend this list to also contain 5 more `AcadosParameter`s, one for each field in the reference part of the cost. Make only the reference of the angle θ learnable.

2.3 Now complete the function `setup_ocp_with_manager` in `setup_ocp_with_manager.py`.

2.4 Familiarize yourself with `AcadosDiffMpcTorch` and `AcadosController`. Create an `AcadosController` in `example_diffMPC.py`. Make sure to set `n_batch_max` of the `AcadosDiffMpcTorch` high enough such that the batch of parameters `p_values` can be passed.

2.5 Run `example_diffMPC.py` to solve a batch of optimization problems with varying reference of the angle and compute their sensitivities. Think about what these plots say in the context of learning approaches aiming to leverage the sensitivities.

2.6 (Bonus) Feel free to play around with values of the parameters, ocp formulations, calculating other sensitivities, or making parameters differ over the horizon (field `end_stages` for the `AcadosParameter`). Note that while playing around, it will likely happen, that the solver will not converge for some of the parameters you will throw at him. Therefore, you might want to adjust your ocp formulation (e.g., adjust "solver_options", adjust fixed values, slack constraints), adjust the sampled parameters (e.g., their range), or just ignore it and filter the non-converged samples for whatever you want to do (e.g., plotting).

Useful Resources

- `acados` problem formulation PDF
- `CartPoleEnv`
- `AcadosParameters`
- `AcadosParameterManager`
- `AcadosDiffMpcTorch`
- `AcadosDiffMpcCtx`
- `AcadosController`