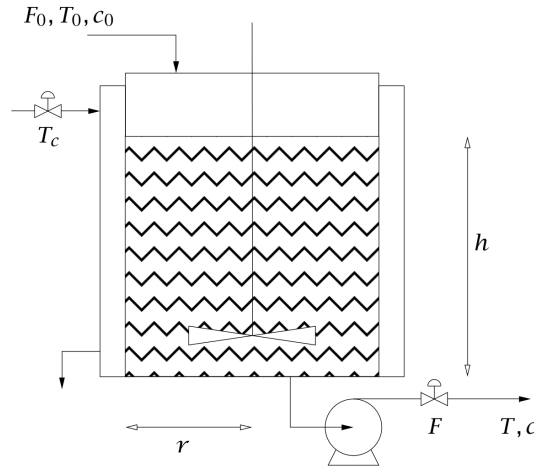


Exercise 2: Numerical Optimal Control and NMPC with MPC-code and acados

Jonathan Frey, Marco Vaccari

This exercise PDF and all accompanying Python template code can be downloaded or cloned from the course repository <https://github.com/CPCLAB-UNIFI/FrontSeatSummerSchool> (please follow the instructions in README.md).

In this and some of the following exercises, we consider a nonlinear continuous stirred-tank reactor (CSTR).¹



An irreversible, first-order reaction $A \rightarrow B$ occurs in the liquid phase and the reactor temperature is regulated with external cooling. Mass and energy balances lead to the following nonlinear model:

$$\begin{aligned}\dot{c} &= \frac{F_0(c_0 - c)}{\pi r^2 h} - k_0 \exp\left(-\frac{E}{RT}\right) c \\ \dot{T} &= \frac{F_0(T_0 - T)}{\pi r^2 h} - \frac{\Delta H}{\rho C_p} k_0 \exp\left(-\frac{E}{RT}\right) c + \frac{2U}{r \rho C_p} (T_c - T) \\ \dot{h} &= \frac{F_0 - F}{\pi r^2}\end{aligned}$$

with states $x = (c, T, h)$ where c is the concentration of substance A , T is the reactor temperature and h is the height. The controls $u = (T_c, F)$ are the coolant liquid temperature T_c and the outlet flowrate F .

Exercise 2.1: MPC-code

Description of closed-loop simulation environment

- `MPC_code.py` is the main file that has to be run containing the simulation of the closed-loop system and plot the trajectories
- `Target_calc.py` defines the steady-state target optimization module
- `Control_calc.py` defines the dynamic optimization module, i.e. the OCP
- `Estimator.py` contains all the possible state estimators

¹The example as well as the figure have been adopted from Example 1.11 in Rawlings, J.B., Mayne, D.Q. and Diehl, M., 2017. *Model predictive control: theory, computation, and design (Vol. 2)*. Madison, WI: Nob Hill Publishing.

- `Utilities.py` contains support functions used in all the other modules
- `Default_Values.py` contains the default values of many options the user can specify in the example file.
- `SS_JAC.ID.py` contains a tool for an automatic system linearization
- `Ex_NMPC.py` defines the example of non-linear MPC containing the model equations stated above with values for all the parameter values and the tuning values for building the NMPC.

Tasks

1. simulate the system with a constant reference control input and nominal NMPC
2. introduce an error in the initial flow rate F_0 of 10% and check how the behavior of the system has changed
3. introduce a non linear disturbance in the model to compensate the offset; use `offree="nl"` and set the dimension of the disturbance "d" to 2.

Exercise 2.2: NMPC with acados

Description of closed-loop simulation environment

- The `main.py` file is prepared to simulate the closed-loop system with `acados` and plot the trajectories.
- The file `cstr_model.py` defines the model equations stated above with values for all the parameter values
- The file `setup_acados_integrator.py` uses this model to generate an integrator of the type `AcadosSimSolver` which we use as our plant model.

Tasks

Uncontrolled simulation:

1. simulate the system with a constant reference control input.

Exact NMPC with acados and CasADi:

2. Set `with_nmpc_controller = True` in `main.py` to also create an `AcadosOcpSolver` and run it in a closed loop simulation.
3. Set `with_casadi_nmpc_controller = True` in `main.py` to also create an `CasadiOcpSolver`.
4. Compare the trajectories with the one in the previous exercise and the computation times of the solvers. Note: the differences are documented in the documentation of `CasadiOcpSolver`.

Fast and approximate controllers:

5. From now on, to save CPU time, we omit the `CasadiOcpSolver` implementation again: set `with_casadi_nmpc_controller = False`
6. Set `with_rti_controller = True` in `main.py` to create an `AcadosOcpSolver` that uses the real-time iteration (RTI) algorithm. This algorithm performs one SQP iteration at each sampling time.
7. Set `with_linear_mpc_controller = True` in `main.py` to create an `AcadosOcpSolver` with a model linearized at the steady state.
8. Compare the resulting closed loop trajectories and the runtime of their controllers

NOTE: If you don't see much difference between LMPC and the constant reference input, run again in a clean `ipython` session or `jupyter` notebook.

Additional exercise (if desired): NMPC with model plant mismatch

- Note that the model parameter F_0 is implemented as a parameter in the `AcadosModel`.
- Task: Introduce a mismatch between the OCP model and the plant, by increasing F_0 in the OCP model by 5%. How well are the references tracked?