25th International Conference on Process Control, June 3-6, 2025, Štrbské Pleso, Slovakia
Workshop on *Model Predictive Control with `MPC-code`: Theory and Implementation*

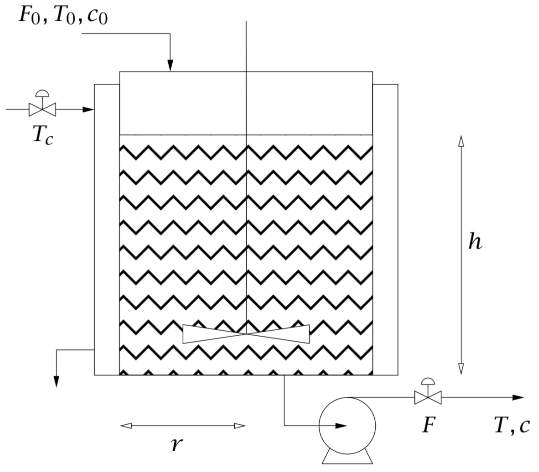
LMPC, NMPC, and more: a Continuous Stirred Tank Reactor Example

Marco Vaccari

This exercise PDF and all accompanying Python template code can be downloaded or cloned from the workshop repository https://github.com/CPCLAB-UNIPI/MPC-Code-WS (please follow the instructions in README.md).

Description of system

Let us consider a nonlinear continuous stirred-tank reactor (CSTR).[1]



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

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

If the <u>reactor volume is constant</u> and the <u>volumetric flowrates of the inflow (F_0) and outflow (F) are the same</u>, the mass and energy balances collapse to:

$$\begin{aligned}
\dot{c} &= \frac{1}{\tau}(c_0 - c) - k_m \exp\left(-\frac{E}{R(T - T_m)}\right) c \\
\dot{T} &= \frac{1}{\tau}(T_0 - T) - \frac{\Delta H}{\rho C_p} k_m \exp\left(-\frac{E}{R(T - T_m)}\right) c + \frac{2U}{r \rho C_p}(T_c - T)
\end{aligned}$$

with states $x = (c, T)$ where c is the concentration of substance A and T is the reactor temperature,control u is the coolant liquid temperature T_c, and finally $\tau = \frac{\pi r^2 h}{F_0}$ is the mean residence time of the CSTR.

[1]The 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.

Table 1: Parameter for simulating the CSTR system

`tau`	3	Mean residence time (min)
`CA0`	2	Feed concentration of A (kmol/m3)
`T0`	298	Feed temperature (K)
`Tm`	298	Reference temperature (K)
`km`	4e-3	Pre-exponential factor calculated at reference temperature (1/min)
`EoR`	8e3	Activation energy over Gas constant (K)
`deltaH`	-3e5	Heat of reaction (kJ/kmol)
`rho`	1e3	Density (kg/m3)
`Cp`	4	Heat capacity (J/g.K)
`UA_V`	340	Heat transfer coefficient * area/ volume (kJ/(m3*min*K)

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
- `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_PC25WS.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. This is the only file to be coded

Tasks

1. write system equations using parameters in Table 1.
2. simulate 200 steps of the system with a constant reference control input (i.e. 0.2 kmol/m^3 for c and 400K for T) and nominal NMPC by launching `MPC_code.py` setting an horizon length of 10 and a discretization time step of 0.25 min.
3. introduce an error in the mean residence time (representing a deviation of the initial flow rate F_0) of 17% in the process equations (`User_fxp_Cont(x,t,u,pxp,pxmp)`) and check how the behavior of the system has changed
4. 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, then insert `"d"` into the model equations (`User_fxm_Cont(x,u,d,t,px)`).
5. test now the nominal LMPC case by linearizing the model at the initial point. How does the behavior change? Do we need to adjust the constraints?
6. finally, test the typical plant model mismatch: linear model inside the MPC versus a non linear plant. How well can you maintain performances?
7. To check how the OCP is created in `Control_calc.py`$\rightarrow$ `opt_dyn`.