ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
IDSC
Institut für Dynamische Systeme und Regelungstechnik

Dr. T. Albin

151-0310-00 **Model Predictive Engine Control** **Group Work 1**

Topic: **Single NMPC step in Matlab and Simulink**

Review: 08. 05. 20

May 6, 2020

Overview

The main goals of the group work are to implement a nonlinear model predictive controller (NMPC) in Simulink, compile and test it on the engine test-bench, and of course to win the final competition ☺. In this first group work, you will implement a single step of the NMPC algorithm in both MATLAB and Simulink.

Problem Description

We formulate the optimal air-path control problem for NMPC as follows:

$$\begin{aligned} \min_{\boldsymbol{x}(\cdot|k),\,\boldsymbol{u}(\cdot|k)} \quad & J(\boldsymbol{x}(\cdot|k), \boldsymbol{u}(\cdot|k)) \\ \text{s.t.} \quad & \boldsymbol{x}(k+i+1|k) = f_{\text{ROM}}(\boldsymbol{x}(k+i|k), \boldsymbol{u}(k+i|k)) \\ & 0 \le u_{\text{vtg}}(\cdot|k) \le 1 \\ & 0 \le u_{\text{egr}}(\cdot|k) \le 1 \\ & \boldsymbol{x}_0 = \boldsymbol{x}(k|k) \end{aligned} \tag{1}$$

Whereas k denotes the general discrete time index, the local index i is used for the prediction/control horizon of a single NMPC step. The function f_{ROM} denotes the discretized system dynamics. We define the cost function J in discrete-time as:

$$\begin{aligned} J(\boldsymbol{x}(\cdot|k), \Delta\boldsymbol{u}(\cdot|k)) = Q_1 \cdot \sum_{i=1}^{N} \Big(p_{\text{im}}(k+i\,|k) - p_{\text{im,ref}}(k)\Big)^2 + Q_2 \cdot \sum_{i=1}^{N} \Big(x_{\text{bg}}(k+i\,|k) - x_{\text{bg,ref}}(k)\Big)^2 + \\ R_1 \cdot \sum_{i=0}^{N-1} \Big(\Delta u_{\text{vtg}}(k+i\,|k)\Big)^2 + R_2 \cdot \sum_{i=0}^{N-1} \Big(\Delta u_{\text{egr}}(k+i\,|k)\Big)^2 \end{aligned} \tag{2}$$

where Q_1 and Q_2 are weights on the reference tracking errors, and R_1 and R_2 are used to penalize the change in the control action, which we define as $\Delta\boldsymbol{u}(k+i\,|k) = \boldsymbol{u}(k+i\,|k) - \boldsymbol{u}(k+i-1\,|k)$. For $i = 0$, we have $\Delta\boldsymbol{u}(k|k) = \boldsymbol{u}(k|k) - \boldsymbol{u}(k-1)$, where $\boldsymbol{u}(k-1) \stackrel{\text{def}}{=} \boldsymbol{u}_{\text{prev}}$ is the applied control action at the previous MPC step. The definitions for the MPC timeline are summarized in Figure 1.

In addition to the weightings, also other parameters of the NMPC algorithm can/must be tuned to obtain good tracking performance with limited computation time. Among others, these are the prediction/control horizon N, the sampling time T_s, the search direction step length α, and the maximum number of SQP iterations n_{SQP}. The complete set of parameters, together with some reasonable values to start with, can be found in the MATLAB template script that is provided with this group work.

For the references $p_{\text{im,ref}}(\cdot)$ and $x_{\text{bg,ref}}(\cdot)$, you can use the trajectories defined in the MATLAB template script, use the ones introduced in the problem sets, or define your own. Notice that for each MPC step, the references are kept constant over the whole prediction/control horizon. In other words, we do not use predictive information about the references.

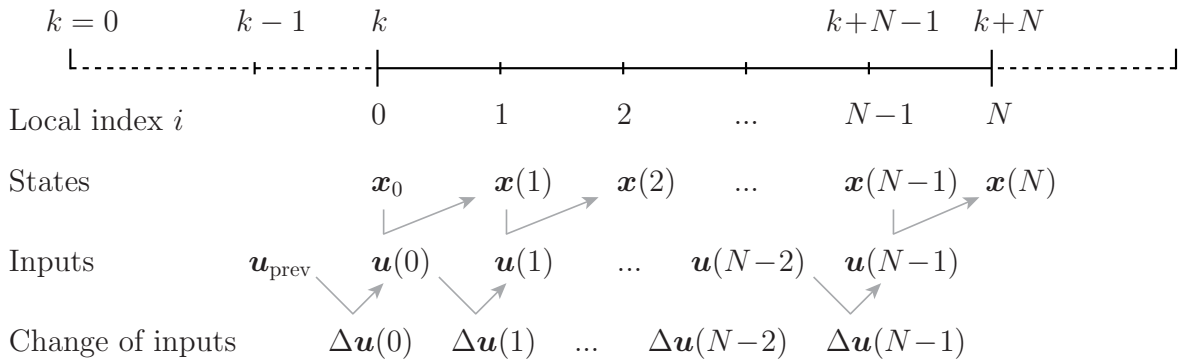

Figure 1: MPC timeline definitions. For graphical clarity, we use the shorthand notation $\boldsymbol{x}(i) = \boldsymbol{x}(k+i|k)$ *for the states, and likewise for the inputs and change of inputs.*

Steps

Use the following list of steps as a guideline to complete this group work. For your convenience, a template MATLAB script `main_GW01.m` is provided that neatly defines the options, parameters, and reference trajectories. It also automatically generates the C code for the S-functions of the Simulink implementation by calling the function `createSfunctions(qp_W,qp_gradJ,qp_gradhT,qp_h)`. The script then triggers both the MATLAB and Simulink simulation and plots the results.

a) Use the solution code from problem set 8, exercise 1, question b (multiple shooting), as a starting point to construct a nonlinear program for the optimal control problem and to create the CasADi functions required for SQP. Wrap your code in a function named `createCasadiFunctions(parNMPC)`, which is then called in the main file.

b) Use the solution code from problem set 8, exercise 2, as a starting point to write a script to solve the nonlinear program using SQP and the solver qpOASES. Wrap your code in a function named `NMPC_Matlab_singlestep(...)`, which is then called in the main file.

c) Run your script to solve the optimization problem. Choose reference values other than the initial outputs of the ROM in order to simulate a step in one (or both) of the references. Evaluate the performance of your MATLAB implementation:

- How does the solution improve/converge as the number of SQP iterations increases?
- What is the effect of N, T_s and n_{RK4} on the computation time?

d) Implement the NMPC algorithm and simulation of the ROM in Simulink. Start with the template model `NMPC.slx`, which contains a Simulink implementation of qpOASES for the SQP, as well as S-functions for the QP matrices. Use the main script to generate the C code for the S-functions, run the Simulink simulation, and plot the results.

e) Validate your implementations of the single NMPC step in MATLAB and Simulink by comparing their respective solutions. More specifically, check if the optimal control trajectories $\boldsymbol{u}^*(\cdot|k)$ are the same. If significant differences are present, investigate and correct the bug(s). We do not recommend to compare the state and/or output trajectories, because they will not be the same as long as the integration/simulation method and corresponding sampling time are not exactly equal.