Numerical Optimal Control, August 2014

Exercise 9: Nonlinear model predictive control

Joel Andersson Joris Gillis Greg Horn Rien Quirynen Moritz Diehl

University of Freiburg – IMTEK, August 7th, 2014

NMPC with direct multiple shooting and Gauss-Newton SQP

In nonlinear model predictive control (NMPC), we repeatedly solve an optimal control problem (OCP) with changing data in order to derive an optimal feedback strategy for a controller. Both direct collocation methods and direct multiple shooting methods are good candidates for transcribing the OCP since in both cases we are able to efficiently exploit the similarity between consecutive problems by initializing the NLP solver. To solve the NLP, we can use either sequential quadratic programming (SQP) or interior-point methods (IP).

Since solving an NLP is an expensive operation, there is often a tradeoff between finding a better solution to the NLP or returning feedback to the system more frequently. In the most extreme case, we just do one iteration of the NLP solver for every feedback time. In the case of an SQP solver, this means solving a single QP.

We continue to work with the simple OCP from the previous exercises:

$$
\begin{array}{ll}
\underset{x,u}{\text{minimize}} & \displaystyle\int_0^T x_1(t)^2 + x_2(t)^2 + u(t)^2 \, dt \\
\text{subject to} & \dot{x}_1 = (1 - x_2^2)\, x_1 - x_2 + u, \qquad x_1(0) = 0, \qquad x_1(T) = 0, \\
& \dot{x}_2 = x_1, \qquad x_2(0) = 1, \qquad x_2(T) = 0, \\
& -1 \le u(t) \le 1,
\end{array} \tag{1}
$$

where $T = 10$ as earlier.

Tasks:

9.1 On the course webpage will find a file `nmpc_ipopt.py` with an implementation of an NMPC controller using direct multiple shooting and IPOPT as an NLP solver. The file depends on the file `plotter.py`, available at the same location, to make nice plots. You should recognize the code from previous exercises, in particular Exercise 7. The code has been modified to have a sum-of-squares objective, which is made possible by extending our RK4 integrator to also output the contribution to the objective at each step of the integrator. Go through the script and make sure that you understand the code. Run the script.

9.2 Modify the script and replace the invocation of IPOPT with your own full-step Gauss-Newton SQP method. Your Gauss-Newton SQP only needs to make a single iteration. You will find a template for doing so in the file `nmpc_gn_template.py`. You can use the Gauss-Newton SQP implementation of Exercise 7 for inspiration. Note that different from that exercise, we allocate a QP solver instance just once and then call it multiple times. Compare the result with Task 9.1 above.

9.3 When just solving a single QP per NMPC iteration, it often make sense to divide the solution code into a *preparation phase* and a *feedback phase*. The preparation phase contains the part of the algorithm that can be calculated before we obtain measurements for the state of the system (i.e. the initial conditions of the ODE). This allows the controller to return feedback to the system faster. What part of the algorithm can be made part of preparation phase?

9.4 **Extra:** Modify the solution to take more than on SQP iteration per NMPC iteration. Does it improve the controller?