TEMPO Summer School on Numerical Optimal Control and Embedded Optimization
University of Freiburg, July 27 - August 7, 2015

# Exercise 10: ACADO code generation for Nonlinear MPC

Rien Quirynen Dimitris Kouzoupis Moritz Diehl

http://syscop.de/teaching/numerical-optimal-control/

In case you did not do this yet, go through the ACADO installation instructions on our course webpage.

### Swing-up of an inverted pendulum

10.1 The task is to make the pendulum perform a full swing-up starting from the stable position described by $p = 0\,\text{m}$ and $\theta = \pi\,\text{rad}$ while the reference point is unstable and corresponds to 0 m, 0 rad. Let us assume that the control input as well as the position of the cart are both bounded, respectively by $-20 \leq F \leq 20$ and $-2 \leq p \leq 2$. The OCP formulation is the following

$$\begin{aligned}
\underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad & \int_0^T \left\| \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} - y(t) \right\|_W^2 \mathrm{d}t \;+\; \|x(T) - y(T)\|_{W_N}^2 && \text{(1a)} \\
\text{subject to} \quad & x(0) \;=\; \bar{x}_0, && \text{(1b)} \\
& \dot{x}(t) \;=\; f(x(t), u(t)), \qquad && \forall t \in [0, T], \quad \text{(1c)} \\
& -2 \;\leq\; p(t) \;\leq\; 2, && \forall t \in [0, T], \quad \text{(1d)} \\
& -20 \;\leq\; F(t) \;\leq\; 20, && \forall t \in [0, T], \quad \text{(1e)}
\end{aligned}$$

where $y(t)$ denotes the reference trajectory, the states are defined as $x = [p, \theta, \dot{p}, \dot{\theta}]^\top$ and the horizon length $T = 2\text{s}$. The used NMPC sampling time is equal to $T_s = 0.05\text{s}$, to be able to deal with the fast nonlinear dynamics in the system. A multiple shooting discretization will be used with $N = 40$ intervals over the control horizon. Go through the template code from our course website and make sure you understand each part of it. You will quickly realize that the following sections in the code are missing:

(a) Define the explicit system of ODE equations, describing our system's dynamics. For this, we need to define an `ode` object like this:

```
ode = [  dot(x) == f(x,u)  ];
```

where `dot` is an ACADO defined operator which returns the time derivative and $f$ denotes the expressions from the right-hand side of our model, as defined in Exercise 1.

(b) Define the stage cost and terminal cost as follows

```
ocp.minimizeLSQ( W, h );
ocp.minimizeLSQEndTerm( WN, hN );
```

where the residual functions $h$ and $h_N$ and the weighting matrices $W$ and $W_N$ are needed to define the discrete objective $\sum_{k=0}^{N-1} \|h_k - y_k\|_{W_k} + \|h_N - y_N\|_{W_N}$. Note that the reference trajectory $y$ will be defined after code generation.

(c) Define the constraints in the OCP, using for example the syntax:

```
ocp.subjectTo( xmin <= x <= xmax );
```

which defines a simple box constraint for a state called $x$ on each stage.

(d) Define the remaining parameters to perform a closed-loop NMPC simulation:

- the initial downward position of the pendulum `X0` and the unstable upward position `Xref`, which forms the reference point to be tracked.
- choose a suitable initial state (`input.x`, $[(N+1) \times n_x]$) and control (`input.u`, $[N \times n_u]$) trajectory to be passed to the RTI solver. A good choice here can be important for the convergence of our scheme as we are solving a nonlinear OCP problem.

- we also need to define the reference trajectory, which remains constant in our case. You can use the matlab function `repmat` to define:
  `input.y = [repmat(Xref,N,1) repmat(Uref,N,1)];`
  NOTE: the dimension of `input.yN` must correspond to the residual function $h_N$ in the terminal cost, which is typically different from the stage cost since the last node defines no corresponding control variables.
  `input.yN = Xref.';`

Do you manage to get a closed-loop NMPC simulation running? If yes, does the pendulum reach the reference point and how would you describe the performance?

NOTE: the flag `EXPORT` at the beginning of the script should be set to zero if one does not want to rerun the code generation and compilation process.

10.2 Tuning: the performance of the NMPC scheme will strongly depend on the various parameters in the OCP formulation such as the number of shooting intervals and the length of the horizon, the number of integration steps and the weighting matrices defining the stage cost. Try to look for values, especially of the weighting matrices, resulting in a satisfactory behavior of the NMPC controller. To avoid the need to go through the code generation and compilation process each single time we change our formulation, one can define e.g. the weighting matrices as

```
W = acado.BMatrix(eye(length(h))); WN = acado.BMatrix(eye(length(hN)));
ocp.minimizeLSQ( W, h );           ocp.minimizeLSQEndTerm( WN, hN );
```

after which they can be defined and passed to the generated solver as

```
input.W = diag([ ... ]);           input.WN = diag([ ... ]);
```

The `BMatrix` keyword stands for *Boolean Matrix* and defines the sparsity of the weighting matrix.

10.3 RTI: remember that ACADO generates a custom RTI scheme to solve each OCP instance, without iterating the SQP method each time until convergence. As a result, the sampling time of the NMPC scheme needs to be small enough to let the solver converge over time and result in a stabilizing controller. Also the choice for the initialization and shifting procedure is typically very important. Let ACADO shift the optimization trajectories after each call to the solver by setting

```
input.shifting.strategy = 1;
```

which means that the last state will be reused at the end of the horizon. You can also use different ways of initializing the state trajectory

`input.x = repmat(X0,N+1,1);` (initialize in the initial state)

`input.x = repmat(Xref,N+1,1);` (initialize in the reference)

Try to get an idea of how these choices affect the convergence of the scheme and therefore the closed-loop control performance. TIP: in case you make the ACADO solver crash such that it spits out `NaNs`, you can reinitialize all its memory as follows:

```
input.control = 1; output = acado_MPCstep(input); input.control = 0;
```

10.4 Stability: one often defines a terminal cost and/or a terminal constraint for stability reasons of the NMPC scheme. A simple example would be to impose the last state in the horizon to be equal to our reference point. You can specifically define constraints on the last stage as follows:

```
ocp.subjectTo('AT_END', ...);
```

10.5 **Extra**: Let us go back to the OCP formulation without any terminal constraint and compare the control feedback map by varying the initial states in the direction of the cart position $p$ and this for both RTI and a fully converged scheme (e.g. by doing 30 iterations). For this extra task, you can look at the template file `RTI_compare.m`. Try to understand this code and interpret the resulting plots when running the script. It is known that the solution manifold is smooth when the active set does not change, but non-differentiable points occur whenever the active set changes. Do you observe this also here? Note that each RTI step for all position values uses the same linearization point, while the resulting manifold follows the exact solution quite well even when active set changes do occur. The latter is known as an (approximate) *generalized tangential predictor*.