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

Exercise 2: Dynamic programming

Rien Quirynen Dimitris Kouzoupis Moritz Diehl

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

Dynamic programming for a 1-state system Here we shall consider a simple OCP with one state x and one control u:

$$\begin{aligned} \underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad & \int_0^T \left(x(t)^2 + u(t)^2\right) \, \mathrm{d}t \;+\; P\, x(T)^2 \\ \text{subject to} \quad & \dot{x} = (1+x)x + 2\,u, \qquad x(0) = -0.95, \\ & -1 \le x(t) \le 1, \quad -1 \le u(t) \le 1, \end{aligned} \tag{1}$$

with horizon length $T = 2$. To be able to solve the problem using dynamic programming, we parameterize the control trajectory into $N = 20$ piecewise constant intervals of size $T_s := T/N = 0.1$. On each interval, we then take 1 step of the RK4 integrator in order to get a discrete-time OCP of the form:

$$\begin{aligned} \underset{x,u}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \left(x_k^2 + u_k^2\right) \;+\; P\, x_N^2 \\ \text{subject to} \quad & x_{k+1} = F(x_k, u_k), \quad \forall k = 0, \ldots, N-1, \quad x_0 = -0.95, \\ & -1 \le x_k \le 1, \quad -1 \le u_k \le 1 \qquad \forall k. \end{aligned} \tag{2}$$

2.1 Similar to previous exercise: Design an LQR controller for this 1-state system, to be used in the terminal cost $x_N^\top P\, x_N$. For this, you will need to linearize the discretized nonlinear system $x_{k+1} = F(x_k, u_k)$ around the steady state point $(\bar{x}, \bar{u}) = (0, 0)$. Write a MATLAB function `ode(t,x,u)` to evaluate the differential equation and use the provided `RK4_integrator` to simulate the system. Important is that you will need both the optimal gain matrix K and the cost-to-go matrix P:

```
[K,P] = dlqr(A,B,Q,R);
```

2.2 Based on the template code `dynamic_programming.m` on our course webpage, try to complete the implementation of dynamic programming for the OCP formulation in (2). Implement the backward pass (recursion) to calculate the cost-to-go function $J_k(x)$ going from $k = N$ to $k = 1$. For $k = N$, the cost-to-go is initialized to $x_N^\top P\, x_N$ as defined by the LQR cost. Fill in the missing lines in the template file for this task. As we will eventually perform a closed-loop receding horizon simulation, we are mainly interested in the cost-to-go $J_0(x)$ and the first control input to be applied $u_0(x)$ as a function of the state x.
NOTE: to be able to implement dynamic programming, one needs to discretize the control $u_k \in U := [-1, 1]$ and state space $x_k \in X := [-1, 1]$. For this, we will use the same equidistant grid for both state and control values:

```
x_values = linspace(-1,1,101);
u_values = linspace(-1,1,101);
```

Using the provided function `project.m`, one can then project a certain value onto the grid of admissible values in the following way:

```
index_x = project(x_value, x_values);
index_u = project(u_value, u_values);
```

2.3 Plot and interpret the cost-to-go function $J_0(x)$ and the feedback map $u(x)$ as a function of x. In the same figure, plot also the function $x^\top P\,x$ and the feedback map $u = -K\,x$ for the LQR controller and compare with the result from dynamic programming.

2.4 Perform a closed-loop simulation (similar to the previous exercise) by completing the template file `closed_loop.m`. Compare and interpret the performance of dynamic programming with the LQR controller. Are the results as you would expect?

2.5 **Extra**: What happens if we instead choose a grid of 100 equidistant points for the state and control values? You can quickly try this out using the same MATLAB code.