Nonlinear Control and Servo Systems (FRTN05)

Computer Exercise 5

Last updated: December 1, 2014

The following exercises should be solved with the optimization tool JModelica.org together with the numerical solver IPOPT. JModelica.org is an open-source software initiated at the Department of Automatic Control and the department is still an active partner in developing the software. For an overview of JModelica.org, see the slides from lecture 11. For more information about the software, visit the homepage `www.jmodelica.org` and especially the User guide found on that homepage.

The software JModelica.org will also be used during Laboratory Exercise 3 in order to do optimal control of a pendulum attached on a cart. Therefore, this computer exercise will let you get acquainted with the software and get feedback from the teaching assistants on the preparations for Laboratory Exercise 3.

Initialization and starting the software

Before you start to solve the problems, you must do the following steps:

1. Download the file `ce5.zip` from the course homepage and unzip it.
2. Open a new terminal and open the directory `ce5` that you downloaded in step 1.
3. Start the optimization software by typing `jmodelica`. This will start a Python environment, which is the interface between the user and the optimization software. By giving appropriate commands you can solve optimization problems and also plot the result of the optimization. The required Python commands can conveniently be collected in a file, which then can be executed in the Python environment, in the same way as with an m-file in MATLAB.

You can now start to solve the problems.

Computer exercises to solve

1. **2D Double Integrator** Consider the following model of a two dimensional double integrator:

$$\ddot{x}(t) = u_x(t)$$
$$\ddot{y}(t) = u_y(t)$$

We would like to find trajectories that transfer the state of the system from $(-1.5, 0)$ to $(1.5, 0)$ in 4.5 s with the control constraint $u_x^2 + u_y^2 \leq 1$. This constraint ensures that the resulting force has a magnitude equal to or less than 1. As a second step we would like to see how the solution changes if we impose the condition $y \geq \cos x - 0.2$, *i.e.*, avoiding a certain area in the XY-plane. Finally, the control energy needed to transfer the state should be minimized. This gives us the following optimal control formulation

$$\min_{u} \int_{0}^{4.5} u_x(t)^2 + u_y(t)^2 dt$$

subject to

$$\begin{aligned}
\ddot{x}(t) &= u_x(t) \\
\ddot{y}(t) &= u_y(t) \\
x(0) &= -1.5, \quad x(4.5) = 1.5 \\
y(0) &= 0, \quad y(4.5) = 0 \\
\dot{x}(0) &= 0, \quad \dot{x}(4.5) = 0 \\
\dot{y}(0) &= 0, \quad \dot{y}(4.5) = 0 \\
1 &\geq u_x(t)^2 + u_y(t)^2 \\
\\
y(t) &\geq \cos x(t) - 0.2 \qquad \text{(Try both with and without)}
\end{aligned}$$

The dynamics of the double integrator system is given by the following Modelica model:

```
model DoubleIntegrator2d
   input Real ux;
   input Real uy;
   Real x(start=-1.5, fixed=true);
   Real vx(start=0, fixed=true);
   Real y(start=0, fixed=true);
   Real vy(start=0, fixed=true);
equation
   der(x) = vx;
   der(vx) = ux;
   der(y) = vy;
   der(vy) = uy;
end DoubleIntegrator2d;
```

and the description of the optimization problem by the following optimization class

```
optimization optDI2d(objectiveIntegrand=ux^2 + uy^2,
                     startTime=0,
                     finalTime=4.5)
   extends DoubleIntegrator2d;
constraint
   x(finalTime) = 1.5;
   vx(finalTime) = 0;
   y(finalTime) = 0;
   vy(finalTime) = 0;
   ux^2 + uy^2 <= 1;
   // y >= cos(x) - 0.2; // Try both with and without
end optDI2d;
```

Notice that the initial conditions are expressed in the Modelica model using the `start` attribute, whereas the terminal constraints are given in the `constraint` clause in the description of the optimization problem.

(a) Read and understand the model and the optimization problem description in the file `DoubleIntegrator2d.mop`. In the file `exercise1.py`, the Python commands required to solve the optimization problem are collected. Make sure you get the gist of all steps. Then, solve the optimization problem by typing
$>$ `run exercise1`
in the Python environment. This will initiate the solution of the optimization problem using JModelica.org and then plot the results.

(b) Examine the result. Is it consistent with the specification of the optimization problem? Interpret the solution from the optimization problem and compare with your intuition.

(c) Add the path constraint $y(t) \geq \cos x(t) - 0.2$ in the optimization problem and solve the new problem by typing
$>$ `run exercise1`
again. Plot the path constraint and check that the solution avoids the specified area in the XY-plane.

2. **Van der Pol oscillator** Consider the Van der Pol oscillator

$$\begin{aligned} \dot{x}_1 &= (1 - x_2^2)x_1 - x_2 + pu \\ \dot{x}_2 &= x_1 \end{aligned}$$

where p is a parameter with the nominal value 1;

(a) In the file `Vanderpol.mop` a start of a Modelica model describing the dynamics of the system is found. Complete the model and add the starting conditions on the states according to $(x_1(0), x_2(0)) = (0, 1)$.

(b) Assuming the Lagrange cost function

$$J = \int_0^{t_f} 10x_1^2 + 10x_2^2 + u^2 \, dt \ ,$$

with $t_f = 10$, create an optimization model `optVDP`, similar to that one used in Exercise 1 for the 2D double integrator.

(c) Solve the optimization problem by typing
$>$ `run exercise2`
in the Python environment and examine the solution.

(d) Add the constraint

$$0 \leq u \leq 1$$

to the optimization problem and solve the problem assuming the same cost function and initial conditions as in **(b)**. Also check what happens if you decrease the upper bound on u.

(e) Compute the minimum-time optimal solution assuming the terminal constraints $(x_1(t_f), x_2(t_f)) = (0, 0)$. To specify that the final time t_f is free, use `finalTime(free=true)` when declaring the optimization class, instead of specifying a value.

3. **Preparations for Laboratory Exercise 3** As part of the preparation for Laboratory Exercise 3, four home-assignments need to be done prior the lab occasion. During this computer exercise session you can discuss your solutions with the TA if you have any questions.