Nonlinear Control and Servo Systems (FRTN05)

Computer Exercise 5

Last updated: December 10, 2018

The following exercises should be solved with the optimization tool `NLOptcontrol.jl`[1] in the programming language `Julia`. Julia is a new programming language, designed for scientific computing. The syntax is similar to MATLAB, but the language is much more powerful, see `https://julialang.org/`. There are multiple examples on how to formulate and solve optimal control problems on the documentaion page for `NLOptcontrol.jl` here `https://juliampc.github.io/NLOptControl.jl/stable/`

This software 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. You are obviously not required to know the Julia language to do the lab, you will only do minor edits in the Julia files during the lab, and the assistant can help you with any questions.

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. Initialize the julia environment and packages by opening a terminal and type
   ```
   > ./setupjulia
   ```
4. In the Julia terminal, run
   ```
   julia> include("setup.jl")
   ```
   This will install all the necessary packages, precompile them, and wait for Julia commands or scripts to be executed. When you want to run a script, you can either copy it into the Julia terminal, or run

   ```
   julia> include("my/path/to/script.jl").
   ```

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

[1] The package can be found at `https://github.com/JuliaMPC/NLOptControl.jl`

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

$$\begin{aligned}
\ddot{x}(t) &= u_x(t) \\
\ddot{y}(t) &= u_y(t) \\
x(0) &= -1.5, \quad x(4.5) = 1.5 \\
y(0) &= 0, \quad y(4.5) = 0 \\
\dot{x}(0) &= 0, \quad \dot{x}(4.5) = 0 \\
\dot{y}(0) &= 0, \quad \dot{y}(4.5) = 0 \\
1 &\geq u_x(t)^2 + u_y(t)^2
\end{aligned}$$

$$y(t) \geq \cos x(t) - 0.2 \qquad \text{(Try both with and without)}$$

The dynamics of the double integrator and system constraints can be written in julia, when defining the states as $(x, \dot{x}, y, \dot{y})$, as:

```
using NLOptControl

# ========== PROBLEM DEFINITION ==============================
# Constraints on states
XL   = [-Inf,-Inf,-Inf,-Inf]
XU   = [Inf,Inf,Inf,Inf]
# Constraints on control, we will add ux^2+uy^2 <= 1, so these won't be active
CL   = [-1.0, -1.0]
CU   = [1.0, 1.0]
# Initial and final constraints on states
X0   = [-1.5, 0, 0, 0]
XF   = [1.5, 0, 0, 0]

# Create problem ================================================================
prob  = define(numStates = 4, numControls=2,
    X0=X0, XF=XF, CL=CL, CU=CU, XL=XL, XU=XU)
# Give name to states and controls
states!(prob, [:x, :vx, :y, :vy],
    descriptions=["x(t)", "vx(t)", "y(t)", "vy(t))"])

controls!(prob, [:ux, :uy], descriptions=["ux", "uy"])

# Setup model equations as julia **expressions** ================================
dx    = Array{Expr}(4)
dx[1] = :( vx[j] )
dx[2] = :( ux[j] )
dx[3] = :( vy[j] )
dx[4] = :( uy[j] )

## Add dynamics to model ========================================================
dynamics!(prob, dx)
```

and the description of the optimization problem, and the code for solving it using the following code

```
# Obstracle constraints
expr1 = :( ux[j]^2 + uy[j]^2 <= 1 )
expr2 = :( y[j] >= cos(x[j]) - 0.2 )

# Try with both [expr1] and [expr1, expr2]
constraints!(prob, [expr1])

# Number of grid points in discretization
N = 700

# Final time fixed, backward Euler discretization
configure!(prob; finalTimeDV=false, tf=4.5, integrationScheme=:bkwEuler, N=N)

# Set objective
obj = integrate!(prob, :( ux[j]^2 + uy[j]^2) )
# Set to minimize obj
@NLobjective(prob.ocp.mdl, Min, obj)

@time optimize!(prob)
```

(a) Read and understand the model in the file `DoubleIntegrator2D.jl` and the optimization problem description and solver in `exercise1.jl`. The second file also includes code for displaying the results. Make sure you get the gist of all steps. Then, solve the optimization problem by typing
`julia> include("exercise1.jl")`
in the Julia environment. This solve the problem 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
`julia> include("exercise1.jl")`
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.jl` a start of a 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 $t_f = 10$, edit the file `exercise2.jl`, similar to that one used in Exercise 1 for the 2D double integrator.

(c) Solve the optimization problem by typing
`julia> include("exercise2.jl")`
in the Julia 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 `finalTimeDV=true` in the `configure!` command, and remove the `tf=...` specification.

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.