1 Time Optimal Control of a Crane

1.1 Introduction

In this exercise, we consider a crane with mass $m$, line length $L$, excitation angle $\phi$, and horizontal trolley position $x$. Here, our control input is the acceleration $a_x$ of the trolley. Note that the velocity $v$ satisfies $\dot{v}(t) = a_x(t)$ as well as $\dot{x}(t) = v(t)$.

Now, the excitation angle $\phi$ of the pendulum satisfies

$$mL^2 \ddot{\phi}(t) + mL \cos(\phi(t))a_x(t) + mLg \sin(\phi(t)) = -b \dot{\phi}(t), \quad (1)$$

where $b$ is a positive damping constant. For this exercise we use the parameters $m = 1 \text{ kg}$, $L = 1 \text{ m}$, $b = 0.2 \text{ J s}$ as well as $g = 9.81 \frac{\text{m}}{\text{s}^2}$.

Our aim is to move the crane from one to another point as fast as possible. However, there are restrictions: first the control input $a_x(t)$ should during the maneuver be between $-5 \frac{\text{m}}{\text{s}^2}$ and $5 \frac{\text{m}}{\text{s}^2}$. Moreover, the crane should be started at $\phi(0) = 0$ and $x(0) = 0 \text{ m}$ being at rest. The target position is $x(T) = 10 \text{ m}$ and $\phi(T) = 0$ stopping again at rest. Here, $T$ is the end time.

1.2 Exercises:

1.2.1 Theoretical part (use pen and paper!)

1. As specified we would like to move the crane as fast as possible between the specified points. What is the mathematical definition of the objective function?

2. How can the restrictions from the introduction be formulated in form of constraints?

3. Summarize once again the objective and the constraints and write them down in the standard formulation of optimal control problems.
4. In order to discretize the control input, we choose a piecewise constant
control discretization with 40 pieces. How many degrees of freedom
does the above optimal control problem have? How many constraints
do you have?

1.2.2 Installation of ACADO (for Linux)

Google for "ACADO Toolkit", and download the current version of the
software. Unpack ACADO using the command:

\[
\texttt{\$ tar xfvz ACADOtoolkit-1.0beta.tar.gz}
\]

Go to the directory ACADOtoolkit-1.0beta and compile the package:

\[
\texttt{\$ cd ACADOtoolkit-1.0beta}
\]
\[
\texttt{\$ make}
\]

Make sure that the installation was successful by running an example:

\[
\texttt{\$ cd examples/ocp}
\]
\[
\texttt{\$ ./getting_started}
\]

1.2.3 Programming part

1. Copy and rename the file \texttt{examples/ocp/getting\_started.cpp} and
adapt your Makefile to reflect this.

2. Solve the above problem optimal control problem in ACADO. You
can use the code template below to learn the syntax. In addition, it
might help to briefly browse through the ACADO online tutorials at
\texttt{www.acadotoolkit.org} to look up the notation.

3. What is the result for the minimum time? Can you interpret the result
for the optimal control input?

4. By default ACADO uses an SQP algorithm with BFGS updates. Use
the ACADO command \texttt{acadoGetTime()} to measure how long the optimi-
ization takes. In the next step we would like to use an exact Hessian.
For this aim we set the option

\[
\texttt{algorithm.set( HESSIAN\_APPROXIMATION, EXACT\_HESSIAN );}
\]

How does the behavior of the algorithm change and how long does it
take?

5. Look up the storage-initialization possibilities on the ACADO web-
page. Try to initialize the code such that a smaller number of SQP
iterations is needed. What happens if you initialize in the numerical
solution found by the algorithm?
```cpp
#include <acado_optimal_control.hpp>
#include <gnuplot/acado2gnuplot.hpp>

int main()
{
    USING_NAMESPACE_ACADO;

    // VARIABLES:
    // 
    // 
    // DifferentialState
    // 
    // Position of the trolley: x;
    // the end time: T;
    // trolley acceleration: ax;

    Parameter T;
    Control ax;

    // DIFFERENTIAL EQUATION:
    // 
    // 
    // DifferentialEquation f(t_start, T);
    // The model equations
    f << ... // implement the model equations.

    // DEFINE AN OPTIMAL CONTROL PROBLEM:
    // 
    // 
    const int numberOfIntervals = 40;

    OCP ocp;
    ocp.minimizeMayerTerm(T);
    ocp.subjectTo(f, numberOfIntervals);
    ocp.subjectTo(AT_START, x == 0.0);
    ocp.subjectTo(AT_END, x == 10.0);
    ocp.subjectTo(0.0 <= T <= 10.0);
    ... // implement the other constraints.

    // DEFINE A PLOT WINDOW:
    // 
    // 
    GnuplotWindow window;
    window.addSubplot(x, "POSITION OF THE TROLLEY: x");
    window.addSubplot(v, "VELOCITY OF THE TROLLEY: v");
    window.addSubplot(phi, "EXCITATION ANGLE: phi");
    window.addSubplot(dphi, "ANGLULAR VELOCITY: dphi");
    window.addSubplot(ax, "THE CONTROL INPUT: aw");

    // DEFINE AN OPTIMIZATION ALGORITHM AND SOLVE THE OCP:
    // 
    OptimizationAlgorithm algorithm(ocp);
    algorithm << window;
    algorithm.solve();

    return 0;
}
```