EN530.603 Applied Optimal Control
Homework #8

November 12, 2021

Due: November 19, 2021

Professor: Marin Kobilarov

This homework will have you use the ACADO Toolkit: a framework for automatic control and dynamics optimization (`http://acado.github.io/index.html`). You will use ACADO to solve a few optimal control problems. Follow the instructions at `http://acado.github.io/matlab_overview.html` to install the ACADO Toolkit Matlab interface. We will refer to the root directory of your ACADO installation as `<ACADO_ROOT>`. Note: In Step 3 it is not necessary to download ACADO using Git. You can just download the .zip file if it is easier for you.

1. Optimal control with a car. ACADO allows users to symbolically specify dynamics, constraints, and cost functions. The framework can then use automatic differentiation to compute derivatives without the user specifying them directly. Using the provided template file `hw8_car_template.m`, symbolically define the dynamics for a car model given by

$$\begin{aligned} \dot{p_x} &= v\cos(\theta) \\ \dot{p_y} &= v\sin(\theta) \\ \dot{\theta} &= v\tan(\delta) \\ \dot{v} &= u_a \\ \dot{\delta} &= u_\delta \end{aligned}$$

   Each state of your system should be declared as a `DifferentialState` object and each control should be defined as a `Control` object. A symbolic differential equation object `f` is defined using the command:

```
f = acado.DifferentialEquation(0, t_f);
```

   where `t_f` is a variable storing a fixed final time. The dynamics of your system can be defined by adding symbolic differential equations to `f`. For example,

```
DifferentialState v;
Control u_a;
f.add(dot(v) == u_a);
```

   Use the examples in `<ACADO_ROOT>/interfaces/matlab/examples/ocp` for guidance.

   (a) In ACADO, the user must first define an optimal control problem:

```
ocp = acado.OCP(0.0, t_f, num_steps);
```

where `num_steps` defines the number of of steps in the optimal control problem. The user then adds desired costs and constraints to the `ocp` object.

Symbollically add a quadratic cost on the controls $u = (u_a, u_\delta)$. To do so, use the function

```
ocp.minimizeLSQ(u, 0);
```

(b) Add path constraints to the optimal control problem so that

$$\begin{aligned} -5 \leq v \leq 5 \\ -5 \leq u_a \leq 5 \\ -\pi/4 \leq \delta \leq \pi/4 \\ -\pi/6 \leq u_\delta \leq \pi/6 \end{aligned}$$

Path constraints can be added using the function `ocp.subjectTo`, e.g.

```
ocp.subjectTo(-5 <= v <= 5);
```

Don't forget to add the dynamics as a constraint using `ocp.subjectTo(f)`.

(c) Add boundary constraints to the optimal constrol problem so that

$$\begin{aligned} p_x(t_0) &= -10.0 \\ p_y(t_0) &= 1.0 \\ v(t_0) &= 0.0 \\ \theta(t_0) &= 0.0 \\ \delta(t_0) &= 0.0 \end{aligned}$$

$$\begin{aligned} p_x(t_f) &= 0.0 \\ p_y(t_f) &= 0.0 \\ v(t_f) &= 0.0 \\ \theta(t_f) &= 0.0. \end{aligned}$$

Boundary constraints can be added using, e.g.

```
ocp.subjectTo('AT_START', p_x == -10.0);
ocp.subjectTo('AT_END', p_x == 0.0);
```

(d) Configure the settings of the optimal control algorithm. Get access to the algorithm settings object using `algo=acado.OptimizationAlgorithm(ocp)`. Then, using the `algo.set` function, set the KKT convergence tolerance to 10^{-8}. This determines how closely the conditions for optimality have to be satisfied before the optimization exits. Furthermore, set the discretization type to 'MULTIPLE_SHOOTING' and set the max number of optimization iterations to 500. Enter the command `help acado.OptimizationAlgorithm.set` for details.

(e) Using `t_f=10` and `num_steps=64`, run the optimal control algorithm and plot the car state and control trajectories.

2. Time optimal obstacle avoidance with a car

(a) Using your script for (2) as a starting point, add a symbolic time parameter to ACADO, e.g. `Parameter T`. Change your differential equation and optimal control problem variables to use the symbolic time `T` instead of a fixed final time `t_f`.

(b) Add a term to your cost function that minimizes the final time using the function `ocp.minimizeMayerTerm(T)` (a Mayer term is just a terminal cost). Modify your quadratic control cost to minimize only u_δ.

(c) Add a path constraint to avoid a circular obstacle with radius 1 placed at the position $(-7, 0)$. Constrain the final time `T` to be between 3 and 10 seconds.

(d) Without proper initialization, the optimization is unlikely to succeed. Using the control solution from problem 2, initialize the solution to this problem using `algo.initializeControls(u0)`, where `u0` is a matrix containing the control trajectory from problem 2 (which gets stored in the variable `out.CONTROLS` after running the optimization script).

(e) Run the optimal control algorithm and plot the state and control history. Plot the (p_x, p_y) trajectory along with the obstacle. Output the optimal final time.

Note: upload your code using the File upload link on the website; in addition submit a printout of your code (only the part that was written/modified by you) and generated plots. Please hand in your homework to our TAs during the Friday TA session, or alternatively drop it off in the drop-off basket in front of Hackerman 117. If by chance you cannot make it to campus on Friday, you can also submit a pdf electronically (although a hard copy is strongly preferred).