Exercises for Lecture Course on Optimal Control and Estimation (OCE)
Albert-Ludwigs-Universität Freiburg – Summer Term 2014

Exercise 4 - AD and Single Shooting

Prof. Dr. Moritz Diehl and Greg Horn

The aim of this exercise is to implement reverse-mode AD and solve an optimal control problem with and without exact derivatives.

Dynamic System

We will use the same paper airplane system as in previous exercises. A matlab function has been provided for you which integrates the system in time, implementing:

$$x_{k+1} = f(x_k, u_k) \tag{1}$$

where $x = [p_x, p_z, v_x, v_z]$ as usual. As well as outputting x_{k+1}, this function also provides $\frac{\partial x_{k+1}}{x_k}$ and $\frac{\partial x_{k+1}}{u_k}$. This function is available at:

`https://github.com/ghorn/OCE-2014/blob/master/exercise4/integrate_airplane_ode.m`

Single Shooting NLP

We will be optimizing the following NLP:

$$\begin{aligned} \underset{U \in \mathbb{R}^{100}}{\text{minimize}} \quad & p_{z,N}(U) && \text{(2a)} \\ \text{subject to} \quad & -1^\circ \leq U_k \leq 10^\circ, \quad k = 0 \ldots N-1 && \text{(2b)} \end{aligned}$$

A matlab function `function [f, grad_f, X] = fobj(U)` has been provided at
`https://github.com/ghorn/OCE-2014/blob/master/exercise4/fobj.m`
This function computes $p_{z,N}$ (`f`) and a time history of states (`X`). It also returns $\frac{\partial p_{z,N}}{U}$ (`grad_f`), but this part is incomplete - you will implement it yourself.

Tasks

1. Use `fobj.m` to solve the NLP using `fmincon` letting matlab estimate derivatives. Your `fmincon` call should look like:

```
opts = optimset('display','iter','algorithm','interior-point','MaxFunEvals',100000);
alphasOpt = fmincon(@fobj, alphas0, [], [], [], [], lb, ub, [], opts);
```

   Use $\alpha_k = 0, \;\; k = 0 \ldots N-1$ as an initial guess. Plot p_x vs $-p_z$ and α vs time.

2. Using reverse mode AD, complete the missing part of `fobj.m` to compute `grad_f`.

3. Solve the NLP with `fobj.m` and `fmincon` again, this time using exact derivatives. Your `fmincon` call should look like:

```
opts = optimset('GradObj','on','display','iter','algorithm','interior-point');
alphasOpt = fmincon(@fobj, alphas0, [], [], [], [], lb, ub, [], opts);
```

   Use $\alpha_k = 0, \;\; k = 0 \ldots N-1$ as an initial guess. Plot p_x vs $-p_z$ and α vs time.