Exercises for Lecture Course on Optimal Control and Estimation (OCE)
Albert-Ludwigs-Universität Freiburg – Summer Term 2015

Exercise 7 - AD and the Sequential Approach (deadline: 22.6.2015, 2:15pm)

Prof. Dr. Moritz Diehl and Jonas Koenemann

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 paper airplane system as we already had in Exercise Sheet 2. Recall that the state of the system is given by $x = [p_x, p_z, v_x, v_z]^\top$ and the control by $u = [\alpha]$. A Matlab function has been provided for you which integrates the system in time, implementing:

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

where the continuous time system dynamics have the form:

$$f(x,u) = \begin{pmatrix} v_x \\ v_z \\ F_x/m \\ F_z/m \end{pmatrix} \tag{2}$$

with

$$\vec{F} = \vec{F}_{\text{lift}} + \vec{F}_{\text{drag}} + \vec{F}_{\text{gravity}}. \tag{3}$$

As well as outputting x_{k+1}, this function also provides $\frac{\partial x_{k+1}}{\partial x_k}$ and $\frac{\partial x_{k+1}}{\partial u_k}$. This function is available at the course page in the file `integrate_airplane_ode.m`.

For more details about the system dynamics have a looks at Exercise Sheet 2 again.

The timestep of the Euler integrator is $h = 0.02$. In this exercise we want to find controls for the airplane so that it gets a maximum velocity in upwards direction at the end of the horizon (after 2 seconds). The horizon length is $N = 100$.

NLP in Sequential Approach (Single Shooting)

We will be optimizing the following NLP:

$$\begin{aligned} \underset{U \in \mathbb{R}^{100}}{\text{minimize}} \quad & \phi(U) = v_{z,N}(U) && \text{(4a)} \\ \text{subject to} \quad & -1^\circ \le U_k \le 10^\circ, \quad k = 0 \dots N-1 && \text{(4b)} \end{aligned}$$

A matlab function `function [phi, grad_phi, X] = phi_obj(U)` has been provided at the course page in the file `phi_obj.m`.

This function computes $v_{z,N}$ (`phi`) and a time history of states (`X`). It also returns $\frac{\partial v_{z,N}}{\partial U}$ (`grad_phi`), but this part is incomplete - you will implement it yourself.

Tasks

1. Use `phi_obj.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(@phi_obj, alphas0, [], [], [], [], lb, ub, [], opts);
   ```

   Use $\alpha_k = 0, \;\; k = 0 \dots N-1$ as an initial guess. Plot p_x vs $-p_z$ and α vs time. How much time and iterations does the solver need to converge?

   (3 points)

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

   (5 points)

3. Solve the NLP with `phi_obj.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(@phi_obj, alphas0, [], [], [], [], lb, ub, [], opts);
```

Use $\alpha_k = 0,\ k = 0 \dots N-1$ as an initial guess. Plot p_x vs $-p_z$ and α vs time. How much time and iterations does the solver need to converge?

(2 points)