Exercises for Lecture Course on Numerical Optimization (NUMOPT)
Albert-Ludwigs-Universität Freiburg – Winter Term 2015-2016

Exercise 7: Calculation of Derivatives

(to be completed during exercise session on Dec 9, 2015 or sent by email to dimitris.kouzoupis@imtek.uni-freiburg.de before Dec 11, 2015)

Prof. Dr. Moritz Diehl, Dimitris Kouzoupis and Andrea Zanelli

Aim of this exercise is to gain experience with all derivative computations discussed in the class.

Exercise Tasks

1. **Control of a dynamic system:** Our goal is to drive the state of a one-dimensional discrete time system to the origin using scalar, piecewise constant controls u_k in N time intervals. More precisely, we are interested in solving the optimization problem:

$$\underset{U,X}{\text{minimize}} \sum_{k=0}^{N-1} u_k^2 + x_N^2\, q \tag{1a}$$

$$\text{subject to: } x_0 = \hat{x} \tag{1b}$$

$$x_{k+1} = x_k + \frac{T}{N}((1-x_k)x_k + u_k), \quad k = 0, \dots N-1, \tag{1c}$$

where $\hat{x}$ is the initial condition of the system, T the terminal time and vectors $U = (u_0, \dots, u_{N-1})$, $X = (x_0, \dots, x_N)$ denote the state and control trajectories respectively. The objective (1a) expresses our aim to bring the terminal state x_N to zero, using the least amount of effort in terms of control actions u_k. The equality constraints (1b) and (1c) uniquely determine the state trajectory $x_0, \dots x_N$ given controls $u_0, \dots, u_{N-1}$. Therefore we can write (1) in the equivalent unconstrained form:

$$\underset{U}{\text{minimize}} \sum_{k=0}^{N-1} u_k^2 + \Phi(U), \tag{2}$$

using the nonlinear function $\Phi(\cdot) : \mathbb{R}^N \to \mathbb{R}$. This function is implemented for you in MATLAB, using elementary operations. You can call it as `f = Phi(U,param)` where $U \in \mathbb{R}^N$ is a certain control trajectory and param a structure with the problem parameters, similarly to the previous exercise. For the purposes of this task, you will need to construct the following structure:

```
param.N  = 50;   % number of discretization steps
param.x0 = 2;    % initial condition on state
param.T  = 5;    % terminal time
param.q  = 50;   % weight on terminal state
```

   (a) Use your code from last week to differentiate $\Phi(U)$ with finite differences. Add the derivative of the quadratic term $\sum_{k=0}^{N-1} u_k^2$ to your result to get the Jacobian of your objective function.

(1 point)

   (b) Using the same syntax, write a function `[F,J] = i_trick(fun,x,param)` that calculates the Jacobian of $\Phi(U)$ using the imaginary trick.

(1 point)

(c) Now let's implement both forward and backward modes of Automatic Differentiation. Before you start coding, which of the two you think it would perform faster in our example and why?

(1point)

(d) Write a MATLAB function `[F,J] = Phi_FAD(U,param)` that returns the function evaluation and the Jacobian of $\Phi(U)$ using the forward mode of AD. Start by copying the code from the given function `Phi`.

(2 points)

(e) Write a MATLAB function `[F,J] = Phi_BAD(U,param)` that implements the backward mode of AD. Once you have everything implemented, run the script `test_derivatives.m` to check (and demonstrate) that your results are correct.

(2 points)

(f) Now solve the optimization problem in (1) using the BFGS method with globalization similarly to the previous exercise (you can simply adapt your code from last week). Plot the state and controls as a function of time to confirm that the systems behaves as expected.

(2 points)

(g) Measure the total time spent in the derivative calculations for the different functions you have implemented. Set $N = 200$ and $q = 200$ to make the difference in performance more clear. Which method performs the best for this problem? Does this comply with your answer in (1c)?

(1 point)

(h) **Extra:** A powerful tool for AD (among many other features) that we may want to use in future exercises is *CasADi*. Installation intstructions (only CasADi, not optistack) can be found here:

`http://optistack.casadi.org/`

Using casadi we can build the Jacobian of our nonlinear function as a symbolic expression within a few lines only. Complete the template `casadi.m` to calculate the Jacobian of $\Phi(U)$. You can also check how much faster casadi is than all your previous implementations.

(2 bonus points)

2. **Optimal perturbation for finite differences:** Assume we have a twice continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$ and we want to evaluate its derivative $f'(x_0)$ at x_0 with finite differences. Further assume that in a neighborhood $\mathcal{N}(x_0)$ it holds:

$$|f''(x)| \leq f''_{\max}, \quad |f(x)| \leq f_{\max} \tag{3}$$

with $\mathcal{N}(x_0) := \{x \,|x_0 - \delta \leq x \leq x_0 + \delta\}$, $\delta > t$ and t the perturbation in the finite difference approximation. The function $f(x)$ can be represented on a computing system with an accuracy ϵ_{mach}, i.e., it is perturbed by noise $\epsilon(x)$:

$$\tilde{f}(x) = f(x)(1 + \epsilon(x)) \quad |\epsilon(x)| \leq \epsilon_{\text{mach}}.$$

(a) Compute a bound ψ on the error of the finite difference approximation of $f'(x_0)$

$$\left| \frac{\tilde{f}(x_0 + t) - \tilde{f}(x_0)}{t} - f'(x_0) \right| \leq \psi(t, f_{\max}, f''_{\max}, \epsilon_{\text{mach}}).$$

(2 points)

(b) Which value t^* minimizes this bound and which value has the bound at t^*?

(1 point)

(c) **Extra:** Do a similar analysis for the central differences where $\tilde{f}'(x_0) = \frac{\tilde{f}(x_0+t) - \tilde{f}(x_0-t)}{2t}$.

(2 bonus points)

This sheet gives in total 13 points and 4 bonus points.