Introduction to Optimization
Exercise 1

Marc Toussaint
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

April 19, 2015

1 Boyd & Vandenberghe

Read sections 1.1, 1.3 & 1.4 of Boyd & Vandenberghe "Convex Optimization". This is for you to get an impression of the book. Learn in particular about their categories of convex and non-linear optimization problems.

2 Getting started

Consider the following functions over $x \in \mathbb{R}^n$:

$$f_{\text{sq}}(x) = x^\top C x \ , \tag{1}$$

$$f_{\text{hole}}(x) = 1 - \exp(-x^\top C x) \ . \tag{2}$$

For $C = \mathbf{I}$ (identity matrix) these would be fairly simple to optimize. The C matrix changes the *conditioning* ("skewedness of the Hessian") of these functions to make them a bit more interesting. We assume that C is a diagonal matrix with entries $C(i,i) = c^{\frac{i-1}{n-1}}$. We choose a conditioning[1] $c = 10$.

a) What are the gradients $\nabla f_{\text{sq}}(x)$ and $\nabla f_{\text{hole}}(x)$?

b) What are the Hessians $\nabla^2 f_{\text{sq}}(x)$ and $\nabla^2 f_{\text{hole}}(x)$?

c) Implement these functions and display them for $c = 10$ over $x \in [-1,1]^2$. You can use any language, but we recommend Python, Octave/Matlab, or C++ (iff you are experienced with numerics in C++). Plotting is oftem a quite laboring part of coding... For plotting a function over the 2D input on evaluates the function on a grid of points, e.g. in Python

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

space = np.linspace(-1, 1, 20)
X0, X1 = np.meshgrid(space, space)
Y = X0**2 + X1**2

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(X0, X1, Y)
plt.show()
```

Or in Octave:

```
[X0,X1] = meshgrid(linspace(-1,1,20),linspace(-1,1,20));
Y = X0.**2 + X1.**2;
mesh(X0,X1,Y);
save 'datafile' Y -ascii
```

Or you can store the grid data in a file and use gnuplot, e.g.:

```
splot [-1:1][-1:1] 'datafile' matrix us ($1/10-1):($2/10-1):3 with lines
```

d) Implement a simple fixed stepsize gradient descent, iterating $x_{k+1} = x_k - \alpha \nabla f(x_k)$, with start point $x_0 = (1,1)$, $c = 10$ and heuristically chosen α.

[1] The word "conditioning" generally denotes the ration of the largest and smallest Eigenvalue of the Hessian.

e) If you use Python or Octave, use an off-the-shelve optimization routine (ideally IP-opt). In Python, scipy.optimize is a standard go-to solution for general optimization problems.