Optimisation Methods: Assignment 4

Deborah Sulem
deborah.sulem@usi.ch

Spring 2025

The purpose of this assignment is to consolidate your knowledge on line search methods and familiarise yourself with the methodologies for computing derivatives.

It is due for **Wednesday 9th April at 2 pm**. Only the second part (Python implementation) will be graded but you can also submit your answers for the first part and the latter will also be corrected (though not graded). For the first part, you need to submit a Python notebook on iCorsi and follow these instructions:

- Put the answers for each part of the question into separate cells.
- Before each cell, put a markdown header that says which part of the question comes in the following cell.
- Good coding style is part of the grade: add clear comments throughout the code when it is necessary.
- Before you submit your notebook, make sure it runs.

Part 0: reading

Read Chapters 6 and 8 of the book "Numerical Optimisation" (Nocedal & Wright).

Part 1: written exercises

Exercise 1
Prove that if $d = -B^{-1}\nabla f(x)$ *with* $B \succ 0$ *and* $\nabla f(x) \neq 0$ *then the angle* θ *between* d *and* $-\nabla f(x)$ *verifies:*

$$\cos(\theta) \geq \frac{\lambda_{min}(B)}{\lambda_{max}(B)} = \frac{1}{\kappa(B)},$$

where $\kappa(B)$ *is the condition number of* B*. Hint: you may use that for a symmetric matrix* A *and any vector* x*,*

$$\lambda_{min}(A)\|x\|^2 \leq x^T Ax \leq \lambda_{max}(A)\|x\|^2,$$

where $\lambda_{min}(A)$ *and* $\lambda_{max}(A)$ *are respectively the smallest and largest eigenvalues of* A*.*

Exercise 2
Show that if f twice continuously differentiable, and if the minimum eigenvalue of $\nabla^2 f(x)$ is larger than m and its maximum eigenvalue is smaller than M then ∇f is Lipschitz continuous with constant M and m-strongly convex.

Exercise 3
Prove that if f is c-strongly convex then

$$2c(f(x) - f(x^*)) \leq \|\nabla f(x)\|^2,$$

with x^* the global minimiser of f.

Exercise 4 (Finite difference)
Prove that the second-order partial derivatives can be approximated by the central difference formula

$$\frac{\partial^2 f}{\partial x_j \partial x_i}(x) \approx \frac{f(x + te_i + te_j) - f(x + te_i - te_j) - f(x - te_i + te_j) + f(x - te_i - te_j)}{4t^2}.$$

What is the approximation error of such formula if f is three times differentiable?

Exercise 5
Consider the function

$$f(x_1, x_2, x_3) = \frac{1}{x_1 x_3} - \log(x_1 x_2).$$

1. *Draw the computational graph of this function.*
2. *Consider the vector $x = (2, 3, -1)^T$. Compute the gradient of f at x using forward propagation. You need to use the notations of the lecture slides ($\hat{s}_k$).*
3. *Now compute the gradient using backpropagation. You need to use the notations of the lecture slides ($\bar{s}_k$).*

Part 2: programming problems

Problems 1 (Line search)
We consider the function

$$f(x) = x_1^2 + 2x_2^2.$$

1. *Define a function that computes $f(x)$ for any $x = (x_1, x_2)$.*
2. *Define a function that computes the gradient of f, $\nabla f(x)$ for any $x = (x_1, x_2)$.*
3. *Evaluate f and its gradient function at $x^{(0)} = (9, 1)^T$.*
4. *Define a function `wolfe_conditions` that verifies if the first and second Wolfe conditions are verified for some $\alpha > 0$, given a function, its gradient function, a current point x, a direction d and the parameters η and $\bar{\eta}$ of the Wolfe conditions. This function should take as input*

- *a function f (callable object)*
- *its gradient function ∇f (callable object)*
- *a current points x*
- *a direction d*
- *a step size α*
- *parameters η and $\bar{\eta}$.*

and should return a tuple of 2 booleans, indicating if each of the Wolfe conditions is verified.

5. *Test the* `wolfe_conditions` *function for the function f at $x^{(0)} = (9, 1)^T$ with $d^{(0)} = -\nabla f(x^{(0)})$, $\alpha = 0.05$, $\eta = 0.01$, $\bar{\eta} = 0.8$.*

6. *Define a function* `backtracking` *that implements the backtracking line search algorithm. This function should take as input*

   - *a function f (callable object)*
   - *its gradient function ∇f (callable object)*
   - *a current points x*
   - *a direction d*
   - *a maximum step size $\bar{\alpha}$*
   - *parameter η of the first Wolfe conditions.*

   and should return the found value of α^* and the new iterate $x_{new} = x + \alpha^* d$. It should also call the function `wolfe_conditions`*.*

7. *Test the* `backtracking` *function for the function f at $x^{(0)} = (9, 1)^T$ with $d^{(0)} = -\nabla f(x^{(0)})$, $\bar{\alpha} = 10$ and $\eta = 0.01$.*

Problems 2 (Unconstrained optimisation)
Consider the function $f(x) = x_1^3 - x_1 + x_2^3 - x_2$ and solve the following problem

$$\min_{x \in \mathbb{R}^2} f(x),$$

using your own implementation of each of the following algorithms:

- *Gradient descent with a fixed constant step size, chosen by you.*
- *Gradient descent with backtracking line search.*
- *Newton's method with backtracking line search.*

with starting point $x^{(0)} = (1, 1)^T$.

Problems 3 (Finite difference) 1. *Define a function* `forward_finite_difference` *that compute the forward finite difference approximation of the first derivative of a function f at a point x. The function* `forward_finite_difference` *must take as input a function f, a point x and a level t used to compute the finite difference.*

2. *Define a function f that compute the sin of any real number x. Then define another function that outputs the (analytical) derivative of f.*

3. *Compute the estimate of $f'(x)$ at $x = 1$ using the function* `forward_finite_difference` *for $t = 10^{-16}, 10^{-15}, \ldots, 10^{-2}, 10^{-1}$ and compute the approximation error for each t. Plot this error versus t in log-log scale.*

4. *Define a function* `central_difference` *that compute the central difference approximation of the first derivative of a function f at a point x. The function* `central_difference` *must take as input a function f, a point x and a level t used to compute the finite difference.*

5. *Compute the estimate of $f'(x)$ at $x = 1$ using the function* `central_difference` *for $t = 10^{-16}, 10^{-15}, \ldots, 10^{-2}, 10^{-1}$ and compute the approximation error for each t. Plot this error versus t in log-log scale and on the same plot, the error of the forward finite difference scheme. Comment the results.*