Exercises

Continuous Optimization

Prof. Dr. Peter Ochs

www.mop.uni-saarland.de/teaching/OPT24

— Summer Term 2024 —

— Assignment 5 —

Exercise 1. [4 points]
Let f be a strictly convex quadratic function on $\mathbb{R}^n$. Show that Newton's Method with $\tau = 1$ converges in one step to the (unique) global minimizer starting from any point $x^{(0)} \in \mathbb{R}^n$.

Exercise 2. [6 points]
Prove the following

(a) For a symmetric positive definite matrix $Q \in \mathbb{R}^{n\times n}$, the eigen vectors $(v_1, \ldots, v_n)$ form a set of conjugate directions wrt to Q.

(b) Suppose $\{d^{(0)}, \ldots, d^{(k)}\}$ are conjugate directions with respect to the symmetric positive definite matrix Q, show that they are linearly independent.

(c) Let k be the iteration number and $\{d^{(0)}, \ldots, d^{(n-1)}\}$ be the Q conjugate directions. Show that $\langle d^{(i)}, \nabla f(x^{(k)})\rangle = 0$ for all $i = 0, \ldots, k-1$.

Exercise 3. [4 points]
Rewrite the BFGS update formula such that the computation requires only matrix–vector products but no matrix–matrix product, i.e. $O(n^2)$ operations rather than $O(n^3)$ operations are required when basic algorithms are used, where n is the problem dimension.

Exercise 4. [8 Points]
Consider the following optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f \in \mathcal{C}^2(\mathbb{R}^n)$. Consider the following definition.

Definition 1. Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix, $b \in \mathbb{R}^n$ and f as in (1). Let an algorithm $\mathcal{A}$ be applied to minimize $g(z) := f(Az + b)$ with initialization z_0, and the same algorithm is applied to minimize $f(x)$ with initialization $x_0 = Az_0 + b$. Then, the algorithm $\mathcal{A}$ is called scale invariant, if it results in iterates that are connected by

$$x^{(k)} = Az^{(k)} + b\,, \quad \text{for all } k \in \mathbb{N}\,.$$

Your tasks are the following.

(a) Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix and $b \in \mathbb{R}^n$. Consider the transformation $x = Az + b$ and denote $g(z) := f(Az + b)$. Calculate the gradient $\nabla g(z)$ in terms of $\nabla f(x)$, and the Hessian $\nabla^2 g(z)$ in terms of $\nabla^2 f(x)$. (3 points)

(b) Show that Newton's method is scale invariant. (3 points)

(c) Show that BFGS is scale invariant with appropriate initialization. (2 points)

Assume the step-size $\tau^{(k)} = 1$ for both the methods. *Hint: In BFGS method, if you use the initial inverse Hessian approximation $\tilde{H}^{(0)}$ with respect to the variable z, then choose initial inverse Hessian approximation H_0 with respect to the variable x, such that $H^{(0)} = A\tilde{H}^{(0)}A^T$.*

Exercise 5. [18 points]

This exercise continues Exercise 4 of Assignment sheet 4. We repeat the text from that exercise in the appendix (see last page of this sheet), for self completeness.

Let $u \in \mathbb{R}^N$, $\mathcal{A} \in \mathbb{R}^{N\times N}$, $\mathcal{D} \in \mathbb{R}^{2N\times N}$ and $f \in \mathbb{R}^N$. We consider the following optimization problem

$$\min_u \Psi(u)\,, \quad \Psi(u) := \frac{1}{2}\|\mathcal{A}u - f\|^2 + \frac{\mu}{2}\|\mathcal{D}u\|^2\,. \tag{2}$$

We use the preconditioning matrix $\mathcal{P} \in \mathbb{R}^{N\times N}$ to transform the problem to

$$\min_v \Phi(v)\,, \quad \Phi(v) := \Psi(\mathcal{P}v)\,, \tag{3}$$

For simplicity, we provide the following expressions, as required in Exercise 2:

$$\mathcal{Q} = \mathcal{A}^T\mathcal{A} + \mu\mathcal{D}^\top\mathcal{D} \text{ and } b = \mathcal{A}^\top f$$

We ignore the constant term in (2) while transforming to (3), as it does not affect the optimization problem.

There are several choices available for choosing the matrix $\mathcal{P}$. Here, we focus on two preconditioners, namely, Jacobi preconditioner and Cholesky decomposition based preconditioner.

Jacobi Preconditioning. The preconditioner is given by

$$\mathcal{P}_{i,i} = \frac{1}{\sqrt{\mathcal{Q}_{i,i}}}\,, \text{ for } i = 1,\ldots,N\,,$$

and $\mathcal{P}_{i,j} = 0$ for $i \neq j$ and $i, j \in \{1,\ldots,N\}$.

Cholesky Preconditioning. Let the Cholesky decomposition of $\mathcal{Q}$ be such that $\mathcal{Q} = \mathcal{L}\mathcal{L}^T$ holds, where $\mathcal{L}$ is a lower triangular matrix. The Cholesky decomposition based preconditioner is given by $P = (\mathcal{L}^{-1})^\top$. The command `np.linalg.cholesky` can be used to compute the Cholesky decomposition, however this command requires dense matrices.

The goal is to implement the preconditioned algorithms that operate directly on u variable. For this purpose, you only need the matrix $\mathcal{P}\mathcal{P}^\top$. You will find further instructions in the code and your tasks are the following.

(a) Implement the Conjugate Gradient Method with Jacobi and Cholesky decomposition techniques by filling in the TODOs in `ConjugateGradient.py` for a generic preconditioner $\mathcal{P}$. In order to compute the Cholesky decomposition, you may use `np.linalg.cholesky` function, for which you need to convert the involved matrices to dense format. Further instructions can be found in `ex04_03_image_deblurring.py`.

(b) Fill in the TODOs in `ex04_03_image_deblurring.py` and consolidate the results. *Note: Do not change the default options for the arguments.* Explain briefly your observations.

(c) Implement Newton's Method to solve (2), by filling in the TODOs in `NewtonMethod.py`.

(d) Fill in the TODOs in `ex04_04_plot_dimension_vs_time.py` to plot the time taken by Newton's method and Gradient Descent when the image dimension is increased. We allocate 20 seconds for each function, thus you may see `TimeoutError` towards the end of the execution.

Submission Instructions: This assignment sheet comprises the theoretical and programming parts.

- **Theoretical Part:** Write down your solutions clearly on a paper, scan them and convert them into a file named *theory(Name).pdf* where Name denotes the name of the student submitting on behalf of the group. Take good care of the ordering of the pages in this file. You are also welcome to submit the solutions prepared with LaTeX or some digital handwriting tool. You must write the full names of all the students in your group on the top of first page.
- **Programming part:** Submit your solution for the programming exercise with the filename `ex03_04_image_deblurring_solution_Name.py` where Name is the name of the student who submits the assignment on behalf of the group. You can only use python3.
- **Submission Folder:** Create a folder with the name *MatA_MatB_MatC* where MatA, MatB and MatC are the matriculation number (Matrikelnummer) of all the students in your group; depending on the number of people in the group. For example, if there are three students in a group with matriculation numbers 123456, 789012 and 345678 respectively, then the folder should be named: *123456_789012_345678*.
- **Submission:** Add all the relevant files to your submission folder and compress the folder into *123456_789012_345678.zip* file and upload it on the link provided on Moodle.
- **Deadline:** The submission deadline is 28.05.2023, 2:00 p.m. (always Tuesday 2 p.m.) via Moodle.

Image Deblurring Problem

We consider an optimization problem that can be used for image deblurring. Let $f \in \mathbb{R}^{n_x \times n_y}$ be a recorded gray-value image that is blurry and noisy. We model the degradation process as follows: There exists an image $h \in \mathbb{R}^{n_x \times n_y}$ (clean image or ground truth image) such that the observed image f is given by

$$f = \mathcal{A}h + n\,,$$

where $\mathcal{A}\colon \mathbb{R}^{n_x \times n_y} \to \mathbb{R}^{n_x \times n_y}$ is a (known) blurr operator and $n \in \mathbb{R}^{n_x \times n_y}$ is a realization of a pixel-wise independent and identically distributed normal distributed random variable (Gaussian noise). In this exercise, the blurr operator implements a convolution with a filter (a small image patch).

In order to reconstruct the clean image as good as possible, we seek for an image $u \in \mathbb{R}^{n_x \times n_y}$ that approximates the inverse of the blurring and looks like an image. Such a reconstruction can be found by minimizing the following optimization problem

$$\min_u \frac{1}{2}\|\mathcal{A}u - f\|^2 + \frac{\mu}{2}\|\mathcal{D}u\|^2\,. \tag{4}$$

A solution tries to minimize both terms as much as possible, i.e., it tries to find the best compromise between a small value of the first term, i.e., $\mathcal{A}u \approx f$, and a small value of the second term, i.e., u being smooth, steered by a positive parameter μ. For the definition of the forward difference operator $\mathcal{D}\colon \mathbb{R}^{n_x \times n_y} \to \mathbb{R}^{n_x \times n_y \times 2}$ and the context of image processing applications, we refer to Section 2.3 of the lecture notes, although this knowledge is not required for solving the following problem.

In the implementation, we consider the vectorized form of the above problem, i.e., we set $N = n_x n_y$ and let $u, f \in \mathbb{R}^N$, $\mathcal{A} \in \mathbb{R}^{N \times N}$, $\mathcal{D} \in \mathbb{R}^{2N \times N}$ and $\mu > 0$. The exact construction of $\mathcal{D}$ and $\mathcal{A}$ can be found in `make_derivatives2D` and `make_filter2D` function in `myimgtools.py`.