Exercises

Continuous Optimization

Prof. Dr. Peter Ochs

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

— Summer Term 2024 —

— Assignment 4 —

Exercise 1. [5 points]
Consider an objective function $f(x) = 4x_1^2 + x_2^2 - 2x_1x_2$. Starting with $x^{(0)} = (-1, -1)$ and $d^{(0)} = (1, 0)$, find $x^{(1)}$ using exact line search. At $x^{(1)}$, find a direction $d^{(1)}$ that gives the optimal solution using exact line search.

Exercise 2. [6 points]
Let A, M be symmetric positive definite matrices. Show that, if (λ, v) is an eigen pair of $M^{-1}A$, then $(\lambda, E^\top v)$ is an eigen pair for the matrix $E^{-1}AE^{-\top}$, where E is a Cholesky factor of M, that is, $M = EE^\top$.

Exercise 3. [9 points]
Following proposition 9.8 from lecture notes, the directions of conjugate gradient method are given by

$$\begin{aligned} d^{(0)} &= -r^{(0)}, \\ d^{(k)} &= -r^{(k)} + \beta_k d^{(k-1)}, \ k = 1, \dots, n-1 \\ \beta_k &= \frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle r^{(k-1)}, r^{(k-1)} \rangle}. \end{aligned}$$

Show that $\operatorname{span}\big(r^{(0)}, \dots, r^{(k)}\big) = \operatorname{span}\big(d^{(0)}, \dots, d^{(k)}\big)$, $k = 1, \dots, n-1$.

Exercise 4. [20 points]
This exercise continues Exercise 4 of Assignment sheet 3. 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

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

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

$$\min_v \Phi(v), \quad \Phi(v) := \Psi(\mathcal{P}v), \tag{2}$$

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 (1) while transforming to (2), 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) We require `timeout-decorator` python package to keep track of maximum time taken by a function. You can install the package with the following command:

`pip install timeout-decorator`.

(b) Implement the Preconditioned Gradient Descent Method (Algorithm 8 in the lecture notes) with Jacobi and Cholesky decomposition techniques by filling in the TODOs in `GradientDescent.py` for a generic preconditioner $\mathcal{P}$. The step-size can be set to $\frac{2}{L+m}$, where L is the largest eigenvalue of the matrix $\mathcal{P}\mathcal{Q}\mathcal{P}^\top$ and m is the smallest eigenvalue of $\mathcal{P}\mathcal{Q}\mathcal{P}^\top$. Make sure to perform all the operations on u and not on v. (7 points)

(c) Implement the Conjugate Gradient Method (without pre-conditioner) by filling in the TODOs in `ConjugateGradient.py`.(7 points)

(d) 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. (6 points)

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 21.05.2023, 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

$$\min_u \frac{1}{2}\|\mathcal{A}u - f\|^2 + \frac{\mu}{2}\|\mathcal{D}u\|^2\,. \tag{3}$$

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`.