Conjugate gradient method for quadratic programming problems

Elizaveta Mochalova

August 7, 2024

1 Introduction

The aim of this project was to implement a Python program for solving convex quadratic optimization problems. To do this first some random convex quadratic optimization problems of the form

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T Q x + c^T x$$

were generated and later solved using two distinct optimization methods: a simple gradient descent method with exact line search and the conjugate gradient method.

2 Implementation

The implementation is divided into three main components: generating random convex quadratic optimization problems, developing a simple gradient descent method, and implementing the conjugate gradient method.

2.1 Generation of the quadratic problem

The first step in the project involves the generation of random convex quadratic optimization problems. This is achieved by:

- generating a vector $c \in \mathbb{R}^n$ of values within the range $[-1, 1]$
- generating a matrix $A \in \mathbb{R}^{n \times n}$ of values within the range $[-1, 1]$
- computing $Q = A^T A$ to get a semi-definite matrix which is a necessary condition for convexity.

2.2 Algorithms

Two optimization techniques were implemented to solve generated quadratic problems.

Gradient Descent Method: this method iteratively updates the solution in the direction of the negative gradient of the objective function. The step size is determined using an exact line search to ensure optimal progress towards the minimum at each iteration and it and it is calculated as

$$\alpha^* = -\frac{\nabla f(x_k)^T d_k}{d_k^T Q d_k}$$

Conjugate Gradient Method: this method accelerates convergence by generating search directions that are conjugate to each other.

During the implementation part of the project a different stopping criterion was used instead of the one provided in the general definition of the algorithms. The number of iterations is set to be equal to the dimensionality of the problem.

Algorithm 1 Gradient Descent Method

```
$x_0 \in \mathrm{I\!R}^{\mathrm{n}}, \mathrm{Q} \in \mathrm{I\!R}^{\mathrm{n\times n}}$
$k = 0$
while $\nabla f(x_k) \neq 0$ do
    $d_k = -\nabla f(x_k)$
    $\alpha_k = -\frac{\nabla f(x_k)^T d_k}{d_k^T Q d_k}$
    $x_{k+1} = x_k + \alpha_k d_k$
    $k = k + 1$
end while
```

Algorithm 2 Conjugate Gradient Method

```
$x_0 \in \mathrm{I\!R}^{\mathrm{n}}, \mathrm{Q} \in \mathrm{I\!R}^{\mathrm{n\times n}}, \mathrm{c} \in \mathrm{I\!R}^{\mathrm{n}}$
$g_0 = Qx_0 + c$
$d_0 = -g_0$
$k = 0$
while $||g_k|| \neq 0$ do
    $\alpha_k = \frac{||g_k||^2}{d_k^T Q d_k}$
    $x_{k+1} = x_k + \alpha_k d_k$
    $g_{k+1} = g_k + \alpha_k Q d_k$
    $\beta_{k+1} = \frac{||g_{k+1}||^2}{||g_k||^2}$
    $d_{k+1} = -g_{k+1} + \beta_{k+1} d_k$
    $k = k + 1$
end while
```

3 Tests

Some tests were performed to evaluate the correctness of the implementation of the algorithms and the convergence of said implementations.

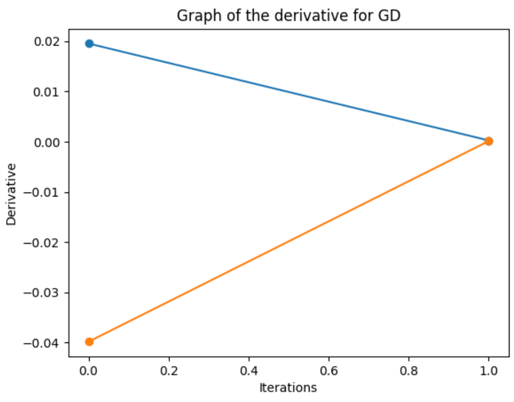
- For the first test it was necessary to check that the results given by the implemented methods were correct using the *check_solution* function. This function given the Q matrix and the vector c is able to calculate exact x for which the equation $Qx = -c$ is satisfied. Most of the time for the Conjugate Gradient Method the difference between the actual result and the result of the algorithm is of the order of 10^{-15} but it goes up the bigger the dimensionality of the problem is. For the Gradient Descent Method the difference is much bigger for the same amount of iterations.

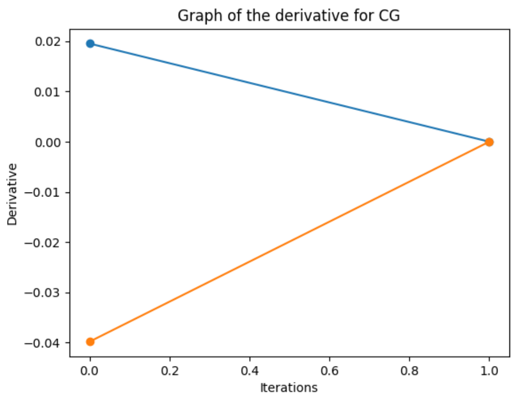
- The next test was to evaluate the convergence rate of the implementations of the algorithms, since in theory Conjugate Gradient Method should converge in a number of iterations equal to the dimensionality of the problem. For both methods this was done by iterating more times over the algorithm, for at most *dimensionality*^{3} times, because in a number of iterations equal to the dimensionality of the problem Gradient Descent Method almost always had the gradient's value bigger then 10^{-1}.

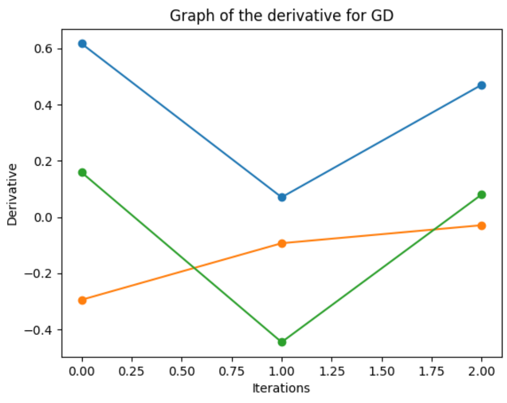
	dim = 3		dim = 5		dim = 10		dim = 20		dim = 50		dim = 75	
threshold	GD	CG	GD	CG	GD	CG	GD	CG	GD	CG	GD	CG
10^{-1}	10	2	23	1	94	8	5.292	20	6.081	57	12.339	91
10^{-2}	N/A	2	59	1	482	8	N/A	20	28.569	62	61.381	99
10^{-3}	N/A	2	97	1	868	9	N/A	20	51.479	64	110.421	101
10^{-5}	N/A	2	N/A	1	N/A	9	N/A	21	97.299	67	208.503	104
10^{-7}	N/A	2	N/A	1	N/A	9	N/A	21	N/A	70	306.579	106

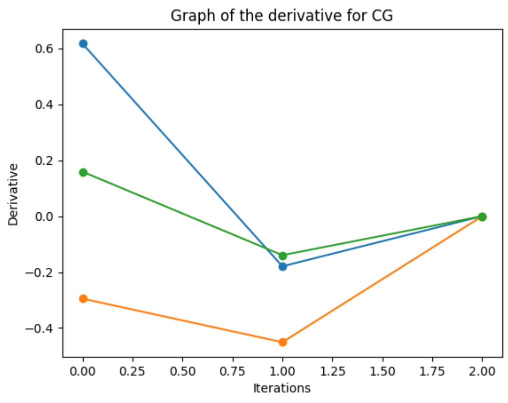
4 Results

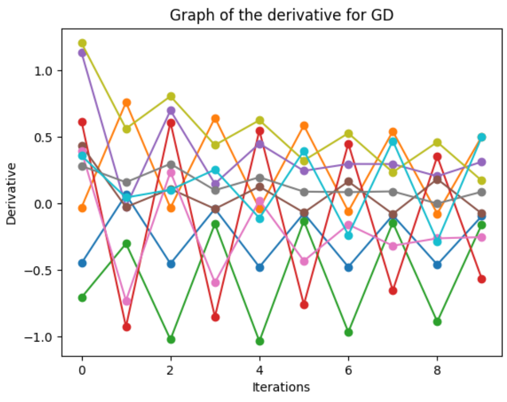
This project successfully implemented the procedures and methods for generating random convex quadratic optimization problems and solving them using gradient-based techniques. The results obtained from the implemented methods correspond to the theoretical expectations. The gradient descent method with exact line search demonstrated consistent convergence towards the minimum, while the conjugate gradient method showed superior performance in terms of the convergence speed. These outcomes validate the correctness and efficiency of the implemented algorithms, confirming that the procedural and methodological approaches adopted in this project are both accurate and effective, but some improvements can be done as discussed in the next section.

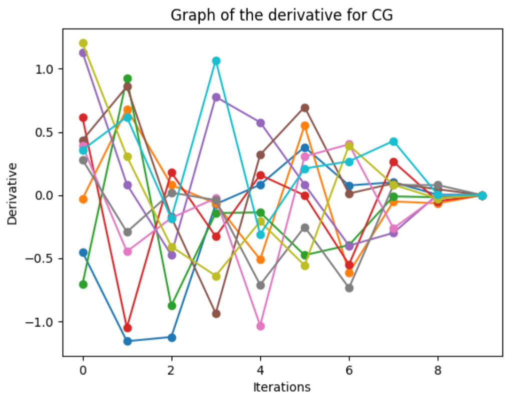


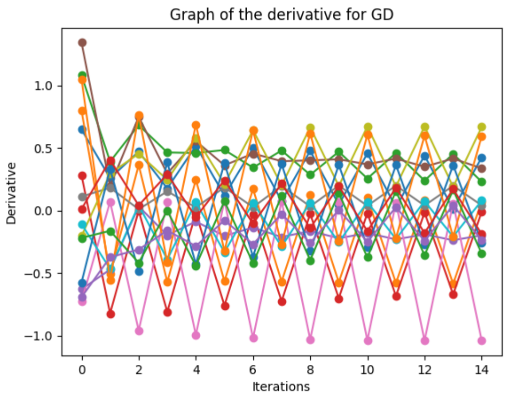


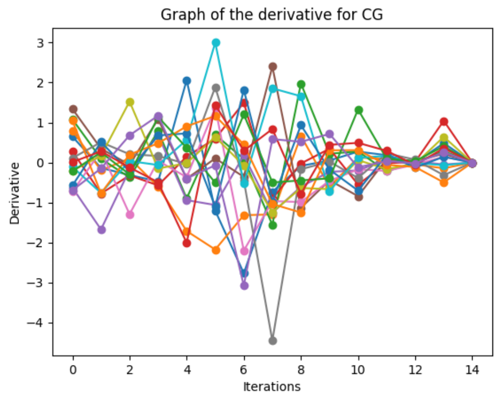


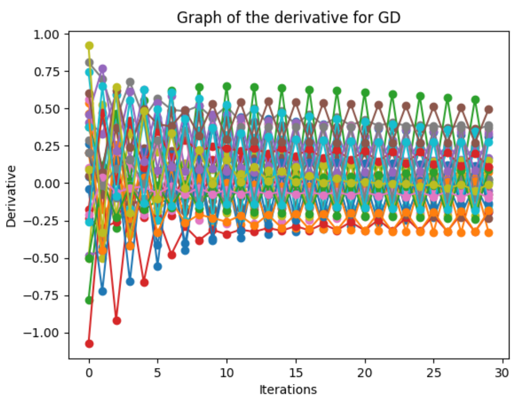


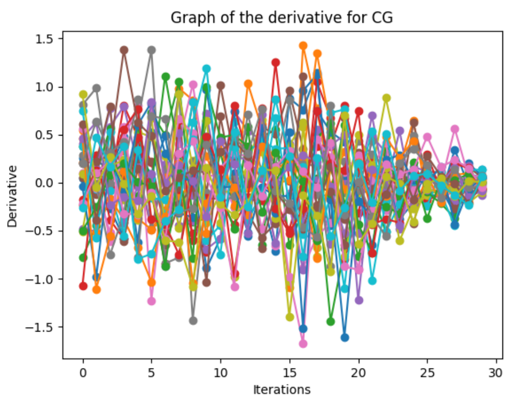


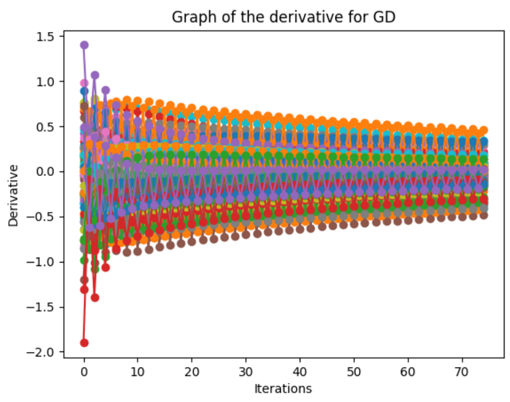


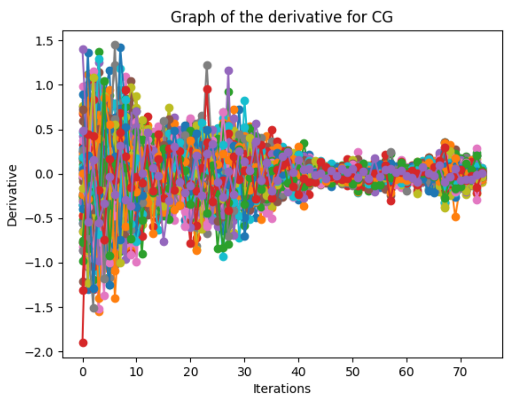












In these images it is possible to see graphs of the derivative of the function for Gradient Descent Method and Conjugate Gradient method on the left and on the right respectively. These are just some of the graphs produced for dimensionality of 2, 3, 10, 15, 30, 75. Only in the last two can you see that the number of iterations for CGM is not enough to converge fully, but in comparison the result of GDM is much worse with values more spread out.

5 Possible improvements

- Preconditioning. It is often used to improve the convergence rate, especially for poorly conditioned matrices. A preconditioner transforms the original problem into one that has more favourable properties for the conjugate gradient method, potentially reducing the number of iterations needed for convergence. In fact it can be seen with the function *condition_number* that when the number is too high Conjugate Gradient Method struggles to converge in less iterations than the dimension of the problem.
- Finite Precision Arithmetic. In practical implementations, due to finite precision arithmetic, the conjugate gradient method may not achieve the exact solution in iterations equal to the dimensionality of the problem. Round-off errors and numerical precision can affect the convergence behavior.

6 Future work

Some of the future work could involve extending the implementation to handle larger-scale problems and exploring other advanced optimization algorithms to further improve performance and applicability and to implement some of the improvements mentioned above.