# 1 One-page report

The table below shows the recorded values of $x_1, ..., x_8$ from all 5 methods. The starting point (1, 4) is also included. After the table, the progress of all 5 methods would be commented on and explained. Note that for line search, the initial step length is set 1.

| | steepest descent | Newton | BFGS | coordinate search | nonlinear conjugate gradient |
|---|---|---|---|---|---|
| $x_0$ | (1.0, 4.0) | (1.0, 4.0) | (1.0, 4.0) | (1.0, 4.0) | (1.0, 4.0) |
| $x_1$ | (1.89, 3.55) | (1.0, 1.0) | (1.75, 3.62) | (2.0, 4.0) | (1.75, 3.62) |
| $x_2$ | (1.77, 3.56) | | (1.87, 3.62) | (2.0, 4.0) | (1.88, 3.58) |
| $x_3$ | (1.82, 3.53) | | (1.74, 3.25) | (2.0, 4.0) | (1.82, 3.57) |
| $x_4$ | (1.4, 2.62) | | (0.91, 0.37) | (2.0, 4.0) | (1.72, 2.94) |
| $x_5$ | (1.63, 2.52) | | (1.33, 1.87) | (2.0, 4.0) | (0.48, 0.24) |
| $x_6$ | (1.51, 2.53) | | (1.17, 1.36) | (1.94, 4.0) | (0.48, 0.24) |
| $x_7$ | (1.55, 2.49) | | (1.04, 0.98) | (1.94, 3.94) | (0.74, 0.23) |
| $x_8$ | (1.49, 2.47) | | (1.05, 1.05) | (1.94, 3.88) | (0.82, 0.31) |

**steepest descent:** When the method was approaching the minimizer, it converged slowly. The method converged in a zig-zag way. The reason is that we implemented the line search here. For the exact line search, due to the chain rule, we can prove that the convergence direction is normal to that in the previous iteration. You could see the demonstration in the appendix. Since inexact line search was implemented here, the two directions don't need to be perpendicular to each other.

**Newton:** The Newton's method found the global minimizer in the first iteration. Even though computational expensive, it has the best performance than other methods in this case. It's because it relies on the Hessian matrix while other methods don't. With second derivative evaluations, it has quadratic convergence. The other reason is that the starting point is near the minimizer. Also, (1,1) is the only local minimizer of the function. It's good for the Newton's method to find it correctly. Sometimes it may be stuck in local minimizers instead of the global minimizer.

When starting from two different points, (-3, -2) and (0, 0.5), the convergence were quite different. For the point (-3, -2), since the starting point is far from the minimizer, the method converged in a very big zig-zag. If there are more than one local minimizer in the function, the method would easily be stuck in a local minimizer instead of the global one. When the starting point is (0, 0.5), the method soon didn't work. Because the Hessian matrix at (0, 0.5) is not positive-definite.

**BFGS:** The convergence direction of the BFGS method in each iteration was not always correct. Because the 'Hessian matrix' it used is not the real Hessian matrix. But since the constructed matrix was always positive-definite, we could observe that the overall convergence direction of BFGS methods was right. In our case, it has the best performance except for Newton's method.

**coordinate search:** In the first 5 iterations, the coordinate search stayed at the same point, and then it moved slightly away from the point. It's because at the point (2, 4), to find a smaller function value, the step length $\gamma$ decreased to a small value(halved every time). In addition, the change of coordinate was only in one direction in an iteration. Because the change in both the two directions may increase the difficulty to converge. It has the worst performance among the 5 methods.

**nonlinear conjugate gradient:** The nonlinear conjugate gradient method converged faster than the steepest gradient one because it made use of the previous searching direction. It's performance is worse than that of Newton's method, since it doesn't apply Hessian matrix. Compared with BFGS method, it is hard for it to correct itself quickly when the direction/step length is not satisfying, because of the negative impact imposed by the previous direction. In BFGS, the correction is like backing a car while in the nonlinear conjugate gradient method, it's like U-turn. Hence, its performance is slightly worse than that of BFGS method.

# 2 Convergence of CG method

## 2.1 condition number and the smallest eigenvalue

| | condition number | the smallest eigenvalue | determinant |
|---|---|---|---|
| $tau = 0.01$ | 1.078 | 0.96217 | 0.92173 |
| $tau = 0.05$ | 2.3844 | 0.5886 | $2.4384 \times 10^{-5}$ |
| $tau = 0.09$ | 37.9236 | 0.051877 | $7.2832 \times 10^{-36}$ |
| $tau = 0.3$ | 1354.1374 | -0.0046001 | $-2.926531356691791 \times 10^{109}$ |

The table above demonstrates the change of condition number, the smallest eigenvalue and determinant with the increase of $\tau$. It's can be concluded that as $\tau$ increases, the condition number increases and the smallest eigenvalue and determinant decrease. In addition, when $\tau = 0.3$, the determinant of the matrix is negative, which means $A$ is no longer positive definite. It can also be reflected by the fact that the smallest eigenvalue of $A$ is negative. The reason is that when $\tau$ is increasing, the correlation of column vectors in the matrix becomes stronger, resulting in a higher condition number. A high condition number means the matrix is ill-conditioned. And for the same reason, with the increase of the correlation of column vectors, the determinant of $A$ is becoming smaller and smaller.

## 2.2 CG method

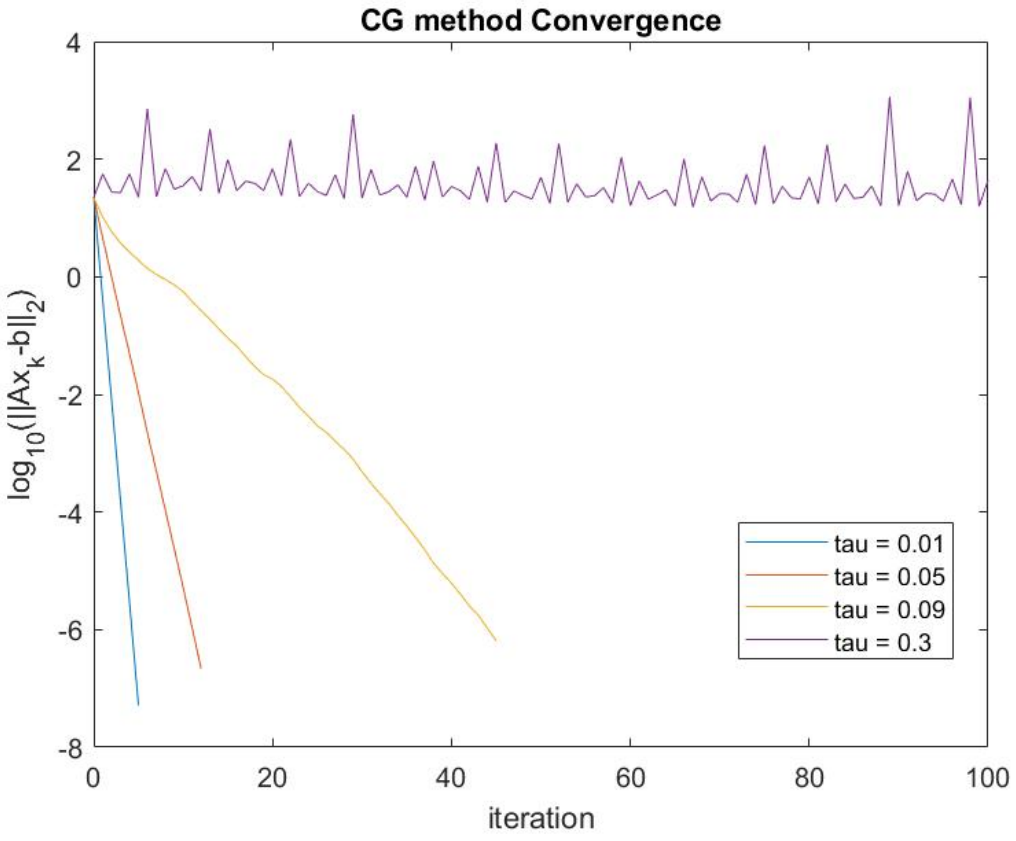


Figure 1: CG method convergence

Figure 1 shows the convergence curves with $\tau = 0.01$, 0.05, 0.09, and 0.3. From the figure we can learn that when $\tau = 0.01$, CG method converged the fastest. With the increase of $\tau$, the convergence rate decreased. When $\tau = 0.3$, the method didn't converge. If $A$ has only r distinct eigenvalues, then the CG method will terminate at the solution in at most r iterations. Thus in a sense, the convergence speed is determined by the distribution of eigenvalues. The higher $\tau$ is, the higher the condition number would be, resulting in a dispersed distribution of

eigenvalues, leading to a lower convergence speed. If the eigenvalues are closed to each other, the method would converge fast.

### 2.3 CG method when $\tau = 3$

CG method requires the matrix $A$ to be positive-definite. Because only when $A$ is positive-definite, the linear system has minimizer. We have proved that when $\tau = 3$, the matrix $A$'s determinant is negative. Therefore, the CG method doesn't converge when $\tau = 3$.

## 3 Necessary condition

### 3.1 KKT conditions

Regarding the equality constrained problem,

$$min_x \frac{1}{2}((x_1 - 1)^2 + x_2^2) \tag{1}$$

which is subject to $-x_1 + \beta x_2^2 = 0$, the Lagrangian function should be:

$$\frac{1}{2}((x_1 - 1)^2 + x_2^2) - \lambda \cdot (-x_1 + \beta x_2^2) \tag{2}$$

Then the KKT conditions ought to be:

$$\begin{bmatrix} x_1 + \lambda - 1 \\ (1 - 2\lambda\beta)x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3}$$

,

$$-x_1^* + \beta x_2^{*2} = 0 \tag{4}$$

and

$$\lambda^*(-x_1^* + \beta x_2^{*2}) = 0 \tag{5}$$

At point $x^* = [0, 0]^T$, when $\lambda^* = 1$, $x_1^* + \lambda^* - 1 = 0$, $(1 - 2\lambda^*\beta)x_2^* = 0$, $-x_1^* + \beta x_2^{*2} = 0$ and $\lambda^*(-x_1^* + \beta x_2^{*2}) = 0$. Therefore, this point satisfies the conditions.

### 3.2 the feasible set of the problem

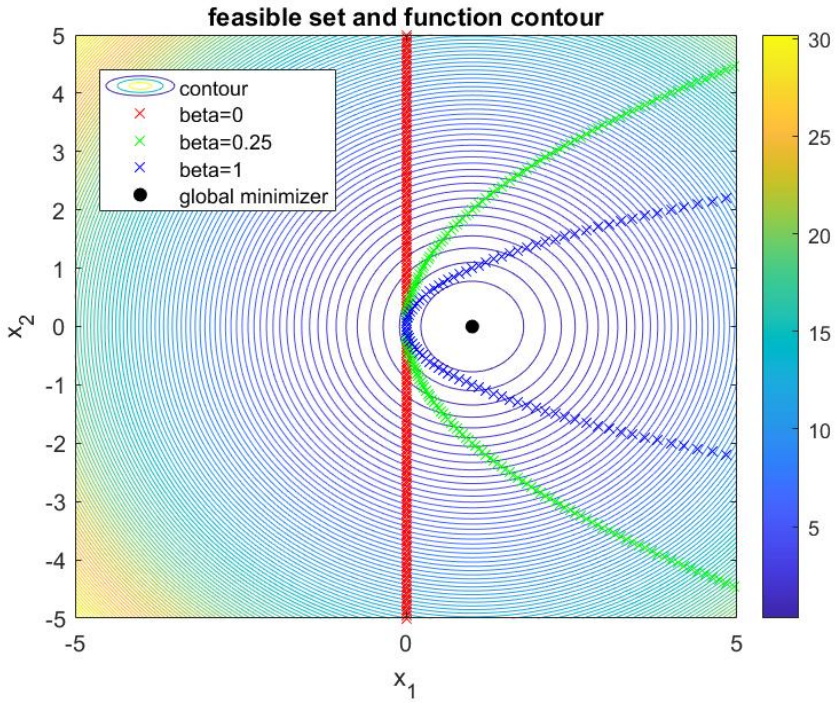


Figure 2: Feasible set and function contour

Figure 2 demonstrates the feasible set of the problem when $\beta = 0$, 0.25 and 1. In the feasible set, $x_1$ is a quadratic curve about $x_2$. With the increase of $\beta$, the quadratic curve became steeper. The black point in the figure is the global minimizer of the objective function without constraint.

### 3.3 further exploration

When $\beta = 1$, $x^*$ is no longer a minimizer, nor a maximizer. The minimizers are located in the blue quadratic curve in the figure, with the shortest distance to the black point in the figure. They are corresponding to lower function values.

## 4 Review: A regularized least-squares problem

### 4.1 the objective function in matrix-vector form

Since the summation of element multiplication could be replaced with matrix multiplication, it's clear that the objective function could be presented in matrix-vector form.

$$||Ax - b||_2^2 + \delta||Lx||_2^2 \tag{6}$$

where $A$ is $[a_1^T, a_2^T, ..., a_k^T]^T$ with size of $k \times n$, $x$ is a vector with size of $n \times 1$, $b = [b_1, b_2, ..., b_k]$ has size of $k \times 1$,

$$L = \begin{bmatrix} -1, 1, 0, 0, ..., 0 \\ 0, -1, 1, 0, ..., 0 \\ ... \\ 0, 0, 0, 0, ..., -1, 1 \end{bmatrix} \tag{7}$$

has a size of $(n-1) \times n$.

### 4.2 normal equation

According to the definition of normal equation, the normal equation in this question should be expressed as:

$$A'^T A' x = A'^T y \tag{8}$$

where $A' = [A, \sqrt{\delta}L]^T$, and $y = [b, \mathbf{0}]^T$. $\mathbf{0}$ is a vector consist of 0.

### 4.3 deblurring the image

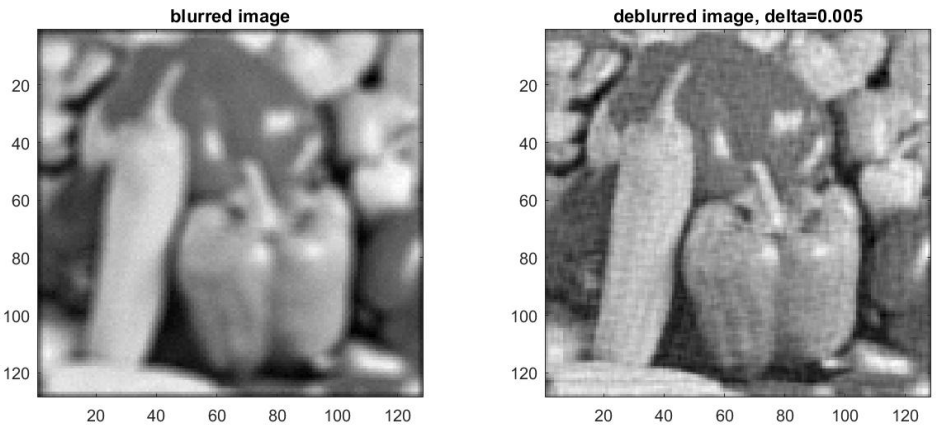


Figure 3: the result of deblurring

Figure 3 illustrates the result of the minimization problem. Solving the normal equation is equal to deblurring the image here.

## 4.4 change $\delta$

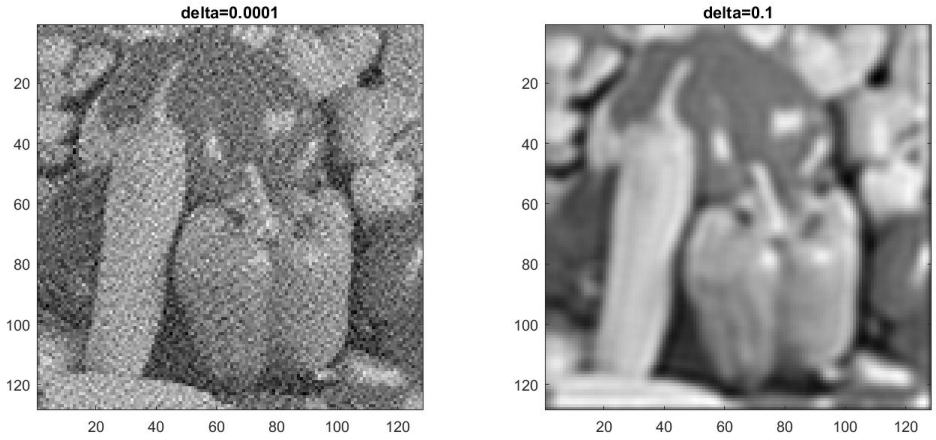


Figure 4: the result of deblurring with different $\delta$

Figure 4 presents the result of deblurring with different $\delta$. From the figure we could learn that with a larger $\delta$, the image becomes more smooth: it is blurred again. When $\delta$ is small, the deblurring result looks still not so good. It's because the objects no longer look continuous. There seem to be more noise points in the image.

Regarding the reason behind this, let's look at the objective function again. It's clear that $\delta$ is the parameter to control the importance of the regularization term or so-called penalty term. The penalty term in the objective function is set to keep the values of the adjacent pixels in the image close to each other. By enlarging $\delta$, the object looks more smooth, with the cost of deblurring effect, because the outline of the objects are no longer clear: pixels in different objects which near object boundaries tend to have similar values.

# 5 Appendix

## 5.1 Why zig-zag in the steepest gradient method

In the steepest gradient method, when using exact line search, we have:

$$\frac{\partial f(x^k - t_k \cdot \nabla f(x^k)/||\nabla f(x^k)||)}{\partial t_k} = 0 \tag{9}$$

where $t_k$ is the step length. By applying the chain rule,

$$\frac{\partial f(x^k - t_k \cdot \nabla f(x^k)/||\nabla f(x^k)||)}{\partial t_k} = \frac{\partial f(x^{k+1})}{\partial x^{k+1}} \cdot \frac{x^{k+1}}{\partial t_k} = \nabla f(x^{k+1}) \cdot \nabla f(x^k) = 0 \tag{10}$$

Obviously, the two convergence directions are perpendicular to each other.