Oregon State University

Homework 6. Conjugate Gradient Method

Hyuntaek Oh

ohhyun@oregonstate.edu

Due: May 28, 2025

ECE599/ AI539 Nonlinear Optimization (Spring 2025)
Homework 6. Conjugate Gradient Method (Due: 11:59pm on May 28, Wednesday.)

Instruction: Students should provide enough detail of the logical procedure of deriving answers. Answers without sufficient justification will receive partial or no credit. You are welcome to use **Python** instead of MATLAB if you prefer.

Reading: Chapter 9 of the textbook (Luenberger and Ye).

1. Exercise 10 in Chapter 9 of the textbook. What does this result imply regarding convergence rate of a conjugate gradient method, especially in comparison with convergence rate of Gradient Descent? (You are allowed to use the inequality in the textbook hint for Exercise 10 without proving it. In addition, check the note on Chebyshev Polynomials, which was posted together with this problem set.)

   Exercise 10 in Chapter 9: Show that for the method of conjugate directions there holds

$$E(\mathbf{x}_k) \leq 4\left(\frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}\right)^{2k} E(\mathbf{x}_0),$$

   where $\gamma = a/A$ and a and A are the smallest and largest eigenvalues of $\mathbf{Q}$. *Hint:* In (27) select $P_{k-1}(\lambda)$ so that

$$1 + \lambda P_{k-1}(\lambda) = \frac{T_k\left(\frac{A+a-2\lambda}{A-a}\right)}{T_k\left(\frac{A+a}{A-a}\right)},$$

   where $T_k(\lambda) = \cos(k \text{ arc } \cos\lambda)$ is the kth Chebyshev polynomial. This choice gives the minimum maximum magnitude on $[a, A]$. Verify and use the inequality.

$$\frac{(1-\gamma)^k}{(1+\sqrt{\gamma})^{2k} + (1-\sqrt{\gamma})^{2k}} \leq \left(\frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}\right)^k.$$

   According to the Theorem 1 in the textbook, the point x_{k+1} generated by the conjugate gradient method satisfies:

$$E(x_{k+1}) = \min_{P_k} \frac{1}{2}(x_0 - x^*)^T Q[I + QP_k(Q)]^2(x_0 - x^*)$$

   where the minimum is taken with respect to all polynomials P_k of degree k.

Suppose that the vector $\mathbf{x}_0 - \mathbf{x}^*$ is written in the eigenvector expansion

$$\mathbf{x}_0 - \mathbf{x}^* = \xi_1 \mathbf{e}_1 + \xi_2 \mathbf{e}_2 + ... + \xi_n \mathbf{e}_n$$

where the $\mathbf{e}_i$'s are normalized eigenvectors of $\boldsymbol{Q}$. Then, since $Q(\mathbf{x}_0 - \mathbf{x}^*) = \lambda_1 \xi_1 \mathbf{e}_1 + \lambda_2 \xi_2 \mathbf{e}_2 + ... + \lambda_n \xi_n \mathbf{e}_n$ and since the eigenvectors are mutually orthogonal, we have

$$E(\mathbf{x}_0) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x})^T \boldsymbol{Q}(\mathbf{x}_0 - \mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} \lambda_i \xi_i^2$$

where the λ_i's are the corresponding eigenvalues of $\boldsymbol{Q}$. Applying it to the point generation by the conjugate gradient method, we find that for any polynomial P_k of degree k there holds

$$E(x_{k+1}) \leq \frac{1}{2}\sum_{i=1}^{n} [1 + \lambda_i P_k(\lambda_i)]^2 \lambda_i \xi_i^2.$$

It then follows that

$$E(\mathbf{x}_{k+1}) \leq \max_{\lambda_i} [1 + \lambda_i P_k(\lambda_i)]^2 \frac{1}{2}\sum_{i=1}^{n} \lambda_i \xi_i^2$$

and hence finally

$$E(\mathbf{x}_{k+1}) \leq \max_{\lambda_i} [1 + \lambda_i P_k(\lambda_i)]^2 E(\mathbf{x}_0).$$

For simplifying the term $[1 + \lambda_i P_k(\lambda_i)]$, we define it as $\rho_k(\lambda)$ So, the key is to find a polynomial term $\rho_k(\lambda)$ that minimizes the maximum $|\rho_k(\lambda)|$ over the interval [a, A], subject to $\rho_k(0) = 1$.

Based on the Chebyshev Polynomials of the first kind:

$$T_k(x) = \cos(k \cos^{-1}(x)), \quad \text{if } |x| \leq 1.$$

To apply them to the interval [a, A], $\lambda \in [a, A]$ is mapped to $x \in [-1, 1]$:

$$x = \frac{2\lambda - (A + a)}{A - a}, \text{ so that } \lambda = \frac{(A - a)x + (A + a)}{2}.$$

This means:

$$\rho_k(\lambda) = \frac{T_k(x)}{T_k(x_0)}, \text{ where } x_0 = \frac{A + a}{A - a} > 1, \quad |T_k(x)| \leq 1, \text{ for } x \in [-1, 1].$$

Then, we get:

$$1 + \lambda P_{k-1}(\lambda) = \frac{T_k(\frac{A+a-2\lambda}{A-a})}{T_k(\frac{A+a}{A-a})}$$

where $\lambda = 0 \Rightarrow \rho_k(0) = 1$

From the above, the worst-case magnitude of $\rho_k(\lambda)$ *is bounded by:*

$$\max_{\lambda \in [a,A]} |\rho_k(\lambda)| \leq \frac{1}{T_k(\frac{A+a}{A-a})} = \frac{1}{|T_k(x_0)|}, \quad x_0 > 1$$

Since $x_0 = \frac{A+a}{A-a} > 1$, *the hyperbolic identity for Chebyshev polynoimals is used:*

$$T_k(x) = \frac{1}{2}\left[(x + \sqrt{x^2-1})^k + (x - \sqrt{x^2-1})^k\right]$$

Now, compute it in terms of $\gamma = \frac{a}{A}$ *and use the inequality in the Problem 1:*

$$x_0 = \frac{A+a}{A-a} = \frac{1+\gamma}{1-\gamma}$$

$$\frac{(1-\gamma)^k}{(1+\sqrt{\gamma})^{2k} + (1-\sqrt{\gamma})^{2k}} \leq \left(\frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}\right)^k$$

The right side of the inequality above can be written:

$$\nu^k = \left(\frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}\right)^k \Rightarrow \frac{(1-\gamma)^k}{(1+\sqrt{\gamma})^{2k} + (1-\sqrt{\gamma})^{2k}} \leq \nu^k$$

This helps to bound the denominator of the Chebyshev expression:

$$|p_k(\lambda)| = \frac{1}{T_k(x_0)} \leq \frac{(1-\gamma)^k}{(1+\sqrt{\gamma})^{2k} + (1-\sqrt{\gamma})^{2k}} \leq \nu^k \Rightarrow |p_k(\lambda)|^2 \leq \nu^{2k} \Rightarrow E(x_k) \leq \nu^{2k} E(x_0)$$

Then, multiplying by a constant factor to relax the bound:

$$E(x_k) \leq 4\nu^{2k} E(x_0) = 4\left(\frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}\right)^{2k} E(x_0)$$

Therefore, the method of conjugate directions holds

$$E(\mathbf{x}_k) \leq 4\left(\frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}\right)^{2k} E(\mathbf{x}_0),$$

where $\gamma = a/A$ *and* a *and* A *are the smallest and largest eigenvalues of* $\boldsymbol{Q}$.

From the result above, the convergence rate of the conjugate gradient method depends on exponentially on $\sqrt{\gamma} = \sqrt{a/A}$. *On the other hand, Gradient Descent (GD) on the same quadratic function yields:*

$$E(x_k) \leq \left(\frac{A-a}{A+a}\right)^{2k} E(x_0) = \left(\frac{1-\gamma}{1+\gamma}\right)^{2k} E(x_0)$$

Both bounds are respectively:

$$\textit{CG method:} \quad E(x_k) \leq 4\left(\frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}\right)^{2k} E(x_0)$$

$$\textit{GD method:} \quad E(x_k) \leq \left(\frac{1-\gamma}{1+\gamma}\right)^{2k} E(x_0)$$

So, this implies that The conjugate Gradient method converges much faster than Gradient Descent when solving quadratic problems (especially for small $\gamma = \frac{a}{A}$*). Because CG's convergence rate depends on* $\sqrt{\gamma}$*, while Gradient Descent depends on* γ*, CG reduces the error much more rapidly, particularly when* $\gamma \ll 1$.

2. (MATLAB experiment) Consider the following minimization problem:

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \mathbf{b}^T\mathbf{x} + \frac{1}{2}\mu(\mathbf{c}^T\mathbf{x})^2. \tag{1}$$

where μ is a large positive constant. As discussed in the past lectures, the above formulation is used to find an approximate solution to minimizing $\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \mathbf{b}^T\mathbf{x}$ subject to $\mathbf{c}^T\mathbf{x} = 0$. Use $\mathbf{Q}, \mathbf{b}$, and $\mathbf{c}$ given in $HW6\_data.mat$, which was posted together with this problem set. For your information, the matrix $\mathbf{Q}$ is a 1000 x 1000 positive definite matrix.

(a) Using MATLAB, solve the first order necessary condition to identify a local minimum point $\mathbf{x}^*$ (for the two cases: $\mu = 1$ and $\mu = 1000$). Argue that this local minimum point is the unique global minimum point.
(*Remark*: Note that when n is very large, this method of obtaining the solution is infeasible due to the memory and computation cost associated with directly solving the first order condition.)

To find a local minimum of the given function:

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} + \frac{1}{2}\mu(\boldsymbol{c}^T\boldsymbol{x})^2.$$

, where $Q \in \mathbb{R}^{n\times n}$ is symmetric and positive definite, $\mu > 0$, and $c, b \in \mathbb{R}^n$
The first-order necessary condition for optimality is solved, which requires the gradient of the objective function to be zero:

$$\nabla f(x) = 0 \Rightarrow Qx - b + \mu(c^T x)c = 0.$$

By rearranging the equation in terms of x, a local minimum point would be:

$$(Q + \mu c^T c)x = b \Rightarrow x^* = (Q + \mu c^T c)^{-1}b$$

According to MATLAB implementation for $\mu = 1$ and $\mu = 1000$, the results of the norm of the gradient are $8.1577e^{-14}$ and $5.5050e^{-13}$ respectively.

To check whether x^* is the unique global minimum point or not, $Q + \mu cc^T$ needs to be symmetric positive definite, meaning that $f(x)$ is strictly convex, and the critical point is the unique global minimizer.

Since Q and cc^T are symmetric, $Q + \mu cc^T$ is also symmetric. Then, for all $x \neq 0$:

$$x^T(Q + \mu cc^T)x = x^T Qx + \mu x^T cc^T x = x^T Qx + \mu(c^T x)^2$$

$$x^T(Q + \mu cc^T)x = x^T Qx + \mu(c^T x)^2 > 0, \quad \forall x \neq 0$$

So, $Q + \mu cc^T$ is positive definite. Furthermore, its Hessian is $\nabla^2 f(x) = Q + \mu cc^T > 0$, meaning that $f(x)$ is strictly convex over $\mathbb{R}^n$. Thus, any stationary point is the unique global minimum.

(b) Implement Gradient Descent with Backtracking Line Search to find a local minimum point. Use $\mathbf{x}_0 = \mathbf{0}$ as the initial point. Explain the parameters of the algorithm you implemented. For $\mu = 1$, plot (i) $||\nabla f(\mathbf{x}_k)||$ versus k, (ii) $f(\mathbf{x}_k)$ versus k, and (iii) $||\mathbf{x}_k - \mathbf{x}^*||$ versus k. For $\mu = 1000$, plot (i) $||\nabla f(\mathbf{x}_k)||$ versus k, (ii) $f(\mathbf{x}_k)$ versus k, and (iii) $||\mathbf{x}_k - \mathbf{x}^*||$ versus k. Interpret the plots. Note that the Gradient Descent method may need a large number of iterations for convergence when μ is large; explain why this is expected.

- *Explain the parameters of the algorithm you implemented.*

  As shown in Figure 1, the parameters are used to implement the algorithm:
  - $\mu = 1, 1000$*: The penalty parameter in the objective function. Large* μ *makes the objective slow down convergence for Gradient Descent.*
  - $\tau = 1.3$*e-5: The stopping criterion based on the gradient norm.*
  - $\alpha = 1$*: The initial step size used at each iteration before backtracking.*
  - $\eta = 2$*: The shrinkage factor used in backtracking.*
  - $\epsilon = 0.4$*: The Armijo (sufficient decrease) condition parameter, where the value is in the range* $\epsilon \in (0, 0.5)$*.*

```
mu = 1;
% mu = 1000;
tau = 1.3e-5;
alpha = 1;
eta = 2;
epsilon = 0.4;
```

Figure 1: Parameters

- *For* $\mu = 1$*, Plot* $||\nabla f(\mathbf{x}_k)||$ *versus* k*, and Interpret the plot*

  Figure 2 shows that the gradient norm decreases nearly exponentially (linear in log-scale), from around 10^1 *down to below* 10^{-5} *within 140 iterations. This indicates consistent progress toward first-order optimality. The small oscillations are typical in backtracking line search and do not hinder overall convergence. The algorithm is effective in reducing the gradient to a very small magnitude, suggesting convergence to a critical point.*

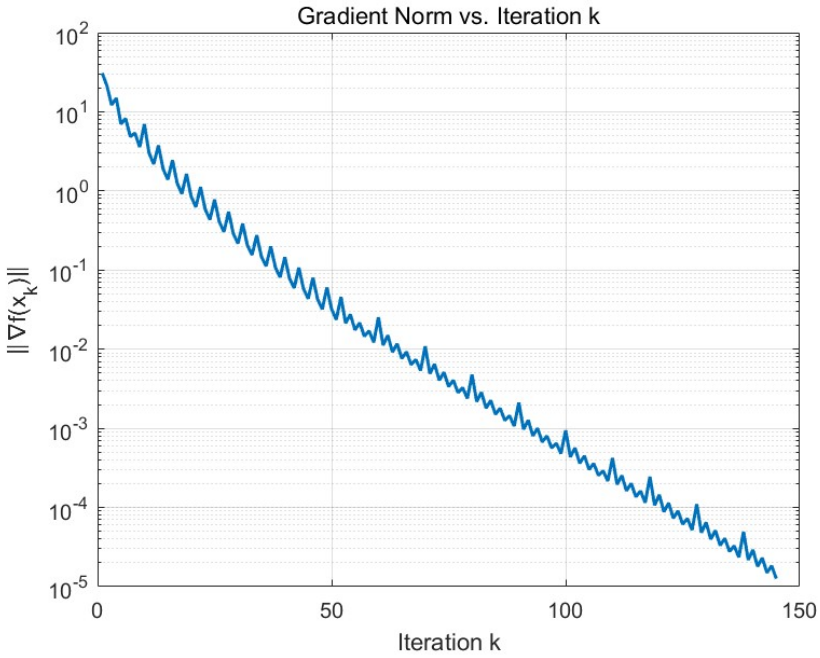

Figure 2: For $\mu = 1$, the norm of gradient in Gradient Descent versus iteration k

- *For $\mu = 1$, Plot $f(\mathbf{x}_k)$ versus k, and Interpret the plot*

  As can be seen in Figure 3, the objective value drops sharply in the first 20 iterations and then flattens out, stabilizing around -0.45. Rapid early decrease indicates that the algorithm finds a good descent direction and step size quickly. The near-flat tail of the curve confirms convergence to the optimal value. This plateau behavior is expected for well-conditioned problems with a modest penalty $\mu = 1$.

- *For $\mu = 1$, Plot $||\mathbf{x}_k - \boldsymbol{x}^*||$ versus k, and Interpret the plot*

  Figure 4 displays that the distance to the known optimal solution decreases steadily and almost linearly on a log scale, reaching 10^{-8} accuracy in fewer than 150 iterations. This also shows global convergence and suggests linear convergence rate (due to its straight line in log scale). The result confirms that the computed solution x_k is very close to the exact solution x^*.

- *For $\mu = 1000$, Plot $||\nabla f(\mathbf{x}_k)||$ versus k, and Interpret the plot*

  According to Figure 5, the gradient norm decreases steadily on a log scale but re-

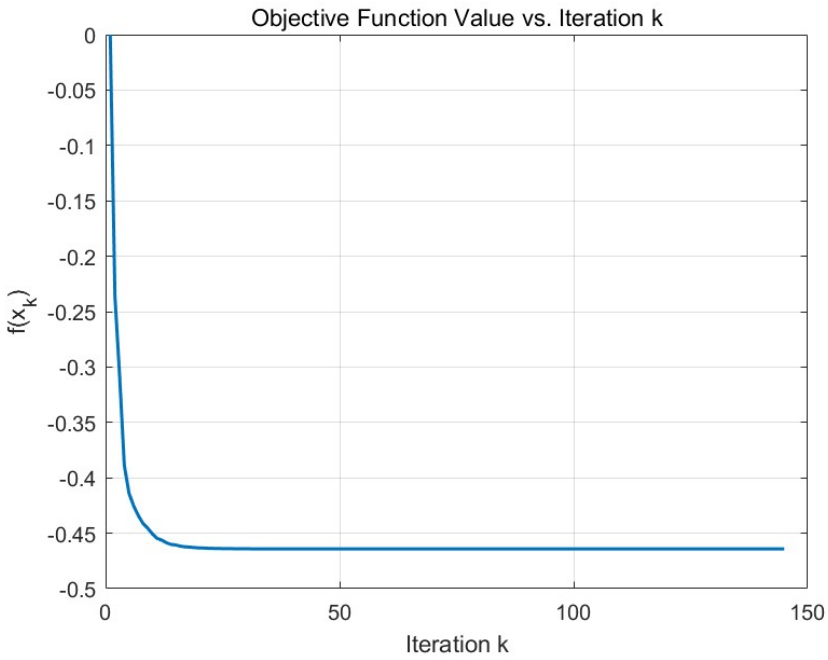

Figure 3: For $\mu = 1$, the function values in Gradient Descent versus iteration k

quires nearly 5500 iterations to reach below 10^{-5}. The convergence is significantly slower than in the $\mu = 1$ case, which converged in 150 iterations. This is expected because a large μ cause to an ill-conditioned objective, meaning that the level curves become elongated, and gradient descent struggles to make fast progress. The shape of the graph shows oscillatory behavior common in poorly conditioned problems with backtracking.

- *For $\mu = 1000$, Plot $f(\mathbf{x}_k)$ versus k, and Interpret the plot*

  Figure 6 shows that the function value decreases rapidly in the beginning, and then plateaus slowly toward the minimum (around -0.45) across thousands of iterations. The early rapid descent indicates that the algorithm makes useful initial progress, and then the function flattens, further confirming the slow convergence induced by high curvature in certain directions.

- *For $\mu = 1000$, Plot $||\mathbf{x}_k - \mathbf{x}^*||$ versus k, and Interpret the plot*

  As shown in Figure 7, the distance to the optimal solution decreases linearly on a log scale, indicating global convergence, but at a much slower rate than in the $\mu = 1$ case.

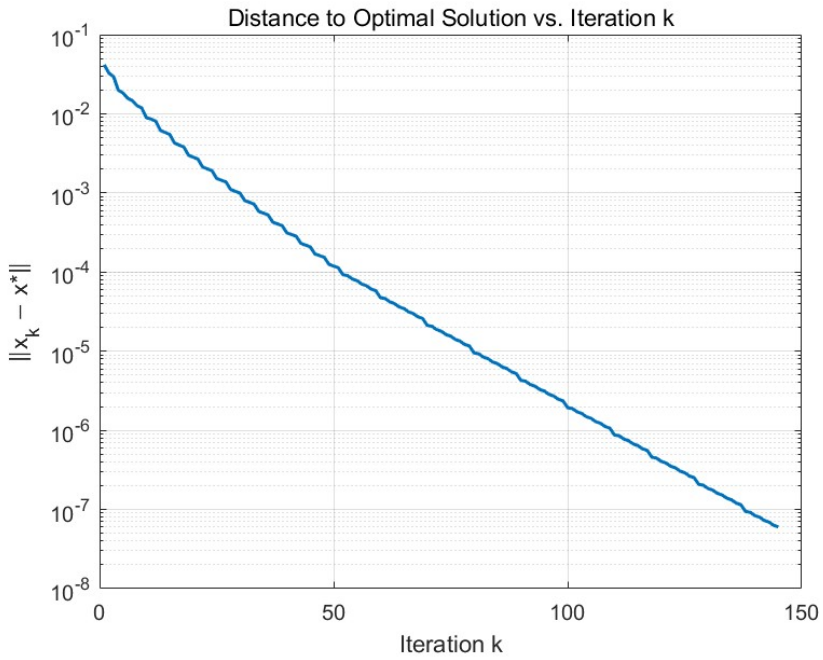

Figure 4: For $\mu = 1$, the distance to optimal point in Gradient Descent versus iteration k

Although the iterates are converging to the solution x^*, the number of iterations is large. This confirms that gradient descent eventually achieves good accuracy (down to 10^{-8}), but requires thousands of iterations to do so.

- *The Gradient Descent method may need a large number of iterations for convergence when μ is large; explain why this is expected.*

  When μ becomes large, the penalty term $\frac{1}{2}\mu(\boldsymbol{c}^T\boldsymbol{x})^2$ dominates the objective function, causing the Hessian $Q + \mu cc^T$ to become ill-conditioned. This means that the eigenvalues of the Hessian vary greatly in magnitude, resulting in a high condition number. As a consequence, the level sets of the objective function become highly elongated. Gradient Descent, which follows the direction of steepest descent, tends to zig-zag across these narrow valleys, making slow progress toward the minimum. Therefore, as μ increases, Gradient Descent requires more iterations to converge.

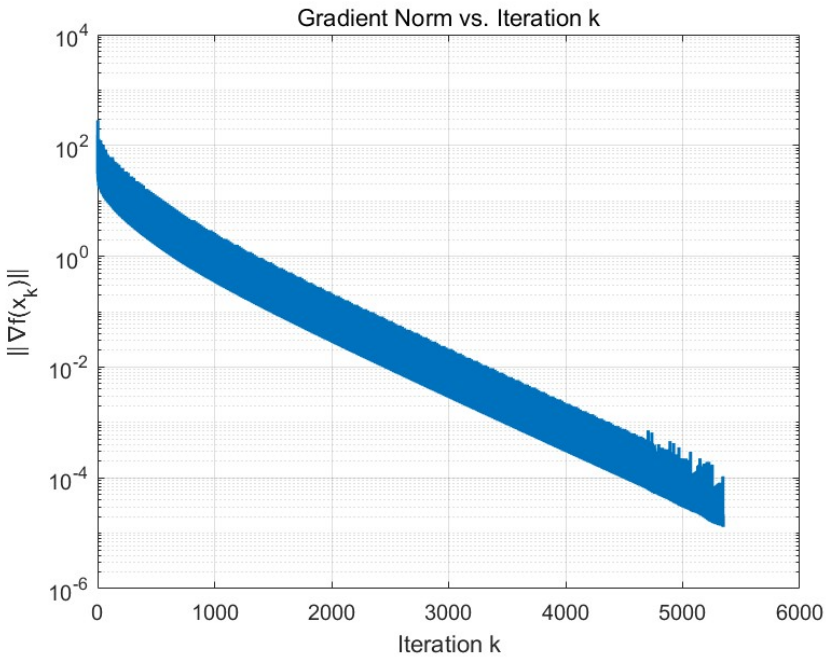

Figure 5: For $\mu = 1000$, the norm of gradient in Gradient Descent versus iteration k

(c) Implement PARTAN with Backtracking Line Search. Specifically, implement the following algorithm (here $\mathbf{g}(\mathbf{x})$ denotes $\nabla f(\mathbf{x})^T$):

- **Step 1.** Starting at $\mathbf{x}_0$, let $\mathbf{x}_1 = \mathbf{x}_0 - \alpha_0 \mathbf{g}(\mathbf{x}_0)$ (α_0 found by Backtracking Line Search).
- **Step 2.** (PARTAN update) For $k = 1, ..., n-1$
  i. $\mathbf{y}_k = \mathbf{x}_k - \alpha_k \cdot \mathbf{g}(\mathbf{x}_k)$ (α_k found by Backtracking Line Search)
  ii. Check whether $\mathbf{y}_k - \mathbf{x}_{k-1}$ is a descent direction at $\mathbf{y}_k$, i.e., whether $\mathbf{g}(\mathbf{y}_k)^T(\mathbf{y}_k - \mathbf{x}_{k-1}) < 0$. If yes, set $\mathbf{d}_k = \mathbf{y}_k - \mathbf{x}_{k-1}$. If not[1], set $\mathbf{d}_k = -(\mathbf{y}_k - \mathbf{x}_{k-1})$.
  iii. $\mathbf{x}_{k+1} = \mathbf{y}_k + \beta_k \cdot \mathbf{d}_k$ (β_k found by Backtracking Line Search)
  iv. If $\mathbf{x}_{k+1}$ satisfies the stopping criterion, return $\mathbf{x}_{k+1}$ and terminate. Otherwise, continue.
- **Step 3.** (Restart) Set $\mathbf{x}_0$ to $\mathbf{x}_n$ and go to **Step 1**.

Use $\mathbf{x}_0 = \mathbf{0}$ as the initial point. Let $\{\bar{x}_k\}$ denote the sequence of points generated by PARTAN. For $\mu = 1$, plot (i) $||\nabla f(\mathbf{x}_k)||$ versus k, (ii) $f(\mathbf{x}_k)$ versus k, and (iii) $||\mathbf{x}_k - \mathbf{x}^*||$ versus k. For $\mu = 1000$, plot (i) $||\nabla f(\mathbf{x}_k)||$ versus k, (ii) $f(\mathbf{x}_k)$ versus k, and (iii) $||\mathbf{x}_k - \mathbf{x}^*||$ versus k. Interpret the plots. Compare the convergence behavior with that of Gradient Descent in Part (b) and provide interpretations.

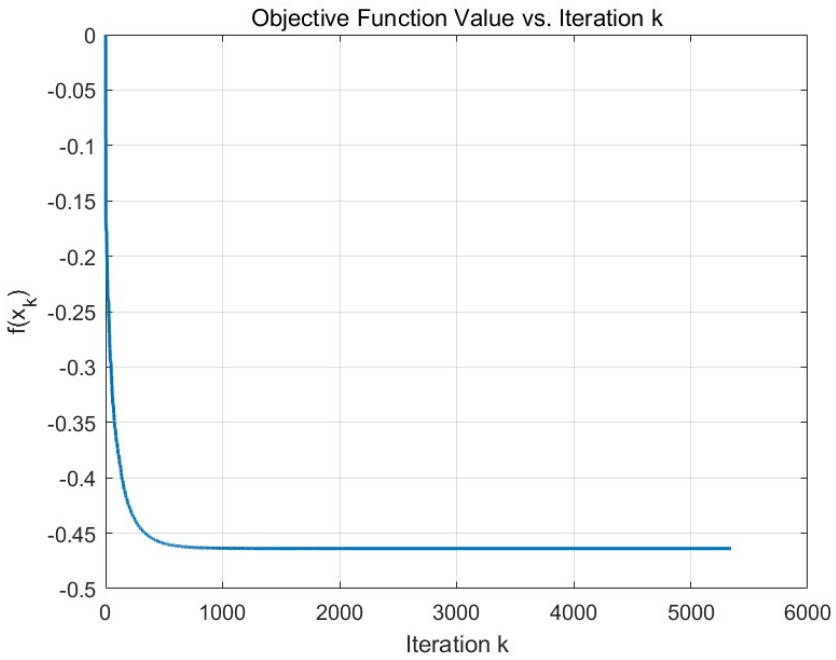

Figure 6: For $\mu = 1000$, the function values in Gradient Descent versus iteration k

- *For $\mu = 1$, Plot $||\nabla f(\mathbf{x}_k)||$ versus k, and Interpret the plot*

  As can be seen in Figure 8, the gradient norm decreases rapidly over just 50 iterations, from 10^{-1} to below 10^{-5}. This indicates fast convergence toward a first-order stationary point. The curve is fairly smooth and steadily decreasing like log-linear trend, suggesting that PARTAN is effectively accelerating the descent.

- *For $\mu = 1$, Plot $f(\mathbf{x}_k)$ versus k, and Interpret the plot*

  In Figure 9, the function value drops steeply within the first 10 iterations and quickly stabilizes near the minimum (≈ -0.45). Most of the objective value decrease happens early, showing that PARTAN is highly efficient. The early flat tail implies that the algorithm reaches the vicinity of the global minimum rapidly. This method is significantly faster than Gradient Descent for the same problem.

- *For $\mu = 1$, Plot $||\mathbf{x}_k - \boldsymbol{x}^*||$ versus k, and Interpret the plot*

  Figure 10 shows that the distance to the optimal solution decreases steadily and nearly linearly in the log scale, reaching 10^{-8} accuracy in under 50 iterations. This supports

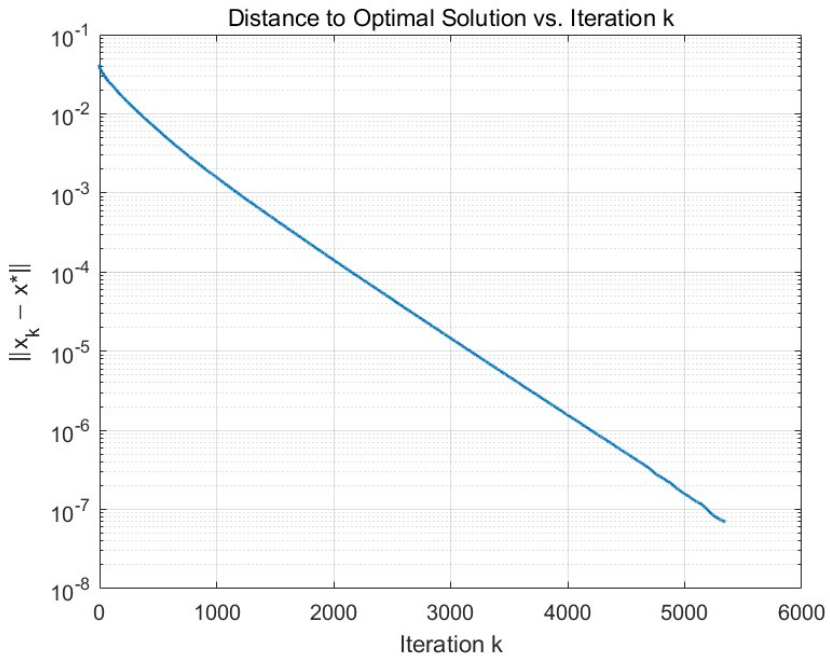

Figure 7: For $\mu = 1000$, the distance to optimal point in Gradient Descent versus iteration k

global convergence with fast rate. The convergence is superior to Gradient Descent, both in speed and precision.

- *For $\mu = 1000$, Plot $||\nabla f(\mathbf{x}_k)||$ versus k, and Interpret the plot*

  As can be seen in Figure 11, this plot shows how the gradient norm decays over the course of the algorithm. Although the overall trend is downward (indicating convergence), the gradient norm exhibits significant fluctuations and requires nearly 700 iterations to reach the termination threshold. Nevertheless, the steady decline across iterations suggests that PARTAN is able to make consistent progress.

- *For $\mu = 1000$, Plot $f(\mathbf{x}_k)$ versus k, and Interpret the plot*

  Figure 12 displays that the function value decreases rapidly during the initial phase (first 100 iterations) and then flattens as it approaches the optimal value near -0.45. This early drop indicates that PARTAN is efficient at finding productive descent directions. The long plateau phase afterward implies that while the solution is close to optimal, refining it to high precision still requires many iterations due to the penalized structure of the problem. Nonetheless, the behavior confirms that PARTAN avoids the

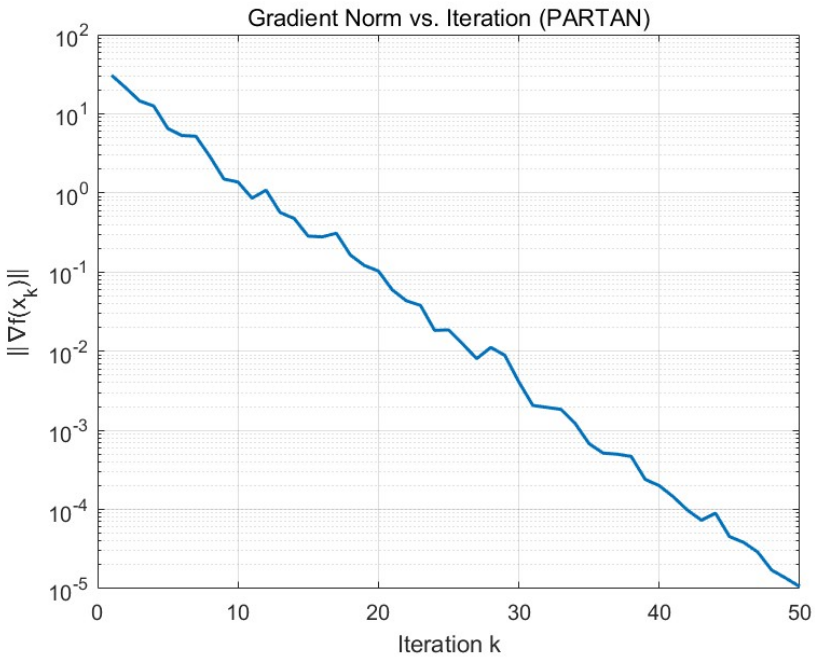

Figure 8: For $\mu = 1$, the norm of gradient in PARTAN versus iteration k

severe slowdowns that typically affect Gradient Descent in high-penalty regimes.

- *For $\mu = 1000$, Plot $||\mathbf{x}_k - \mathbf{x}^*||$ versus k, and Interpret the plot*

  According to Figure 13, the plot presents the norm of the error, which decreases approximately from 10^{-1} to 10^{-8}. The log-linear trend with small curvature suggests global convergence and roughly linear convergence rate. The distance reaches before iteration 700. This performance demonstrates that PARTAN can effectively produce highly accurate solutions faster than standard Gradient Descent, which required significantly more iterations for the same μ.

- *Compare the convergence behavior with that of Gradient Descent in Part (b) and provide interpretations.*

  For the case of $\mu = 1$, PARTAN outperforms Gradient Descent in both speed and efficiency. As described above, while Gradient Descent's convergence is in 140 iterations, PARTAN's convergence is in under 50 iterations. The reason why PARTAN's performance is significantly better is that it exploits previous step directions to accelerate convergence and avoids the redundant zig-zagging behavior that slows down

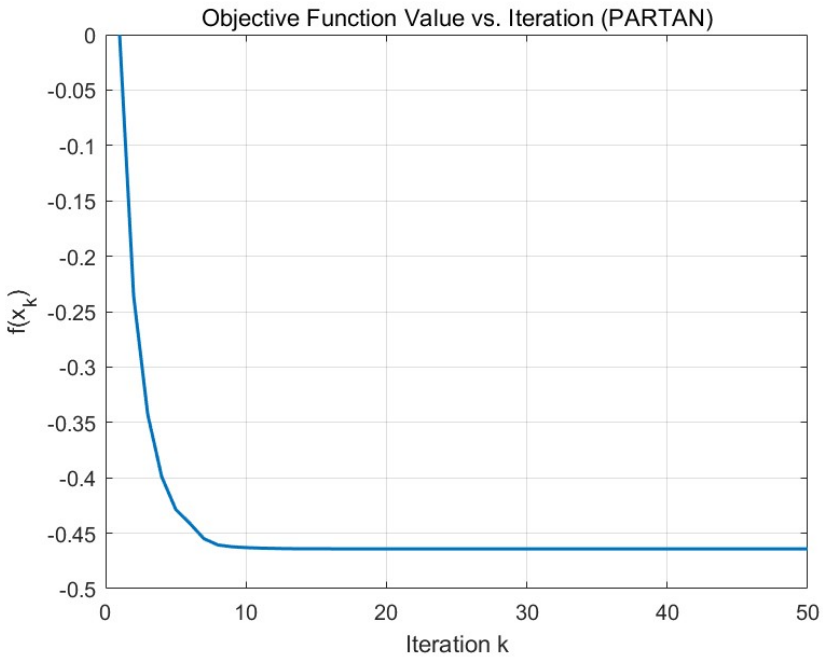

Figure 9: For $\mu = 1$, the function values in PARTAN versus iteration k

Gradient Descent.

For the case of $\mu = 1000$ (large μ), Gradient Descent becomes inefficient due to poor alignment between steepest descent directions and optimal directions. It struggles with zig-zagging and inefficient directions and requires nearly 5500 iterations to reduce gradient norm below threshold. In contrast, PARTAN still maintains steady progress and convergences in significantly fewer iterations (under 700, about 1/9 the iterations of GD), showing that overall trend is downward and steady convergence. The extrapolation step in PARTAN mitigates curvature effects and leads to more efficient updates.

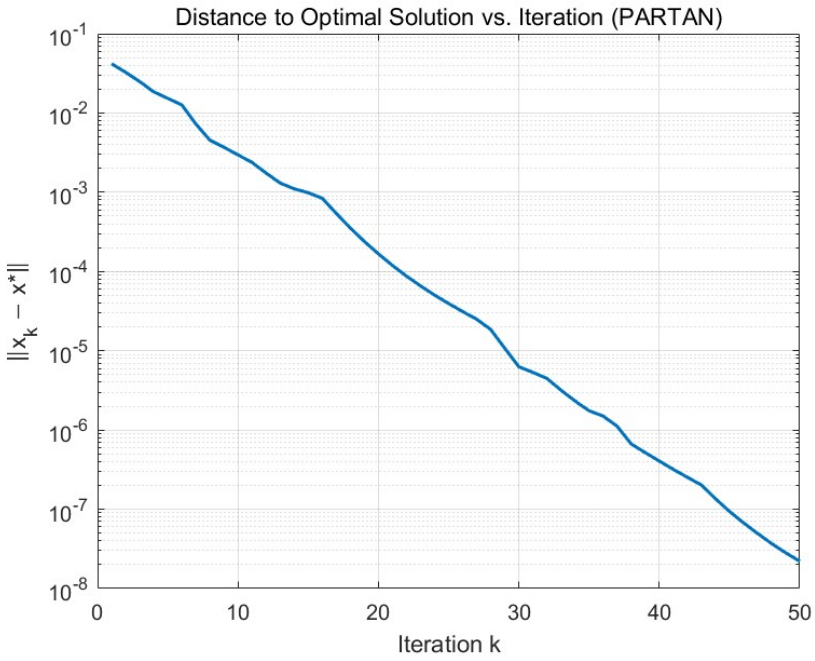

Figure 10: For $\mu = 1$, the distance to optimal point in PARTAN versus iteration k

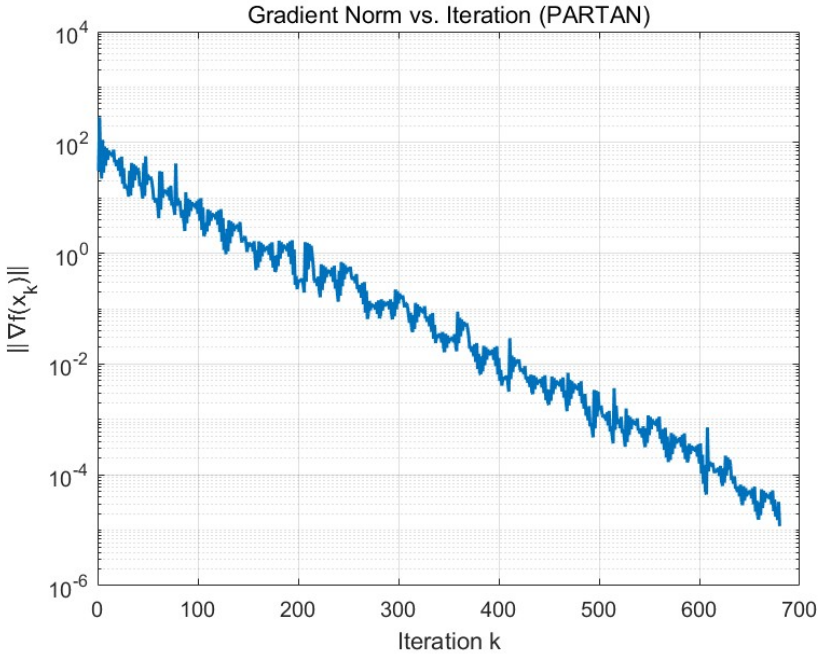

Figure 11: For $\mu = 1000$, the norm of gradient in PARTAN versus iteration k

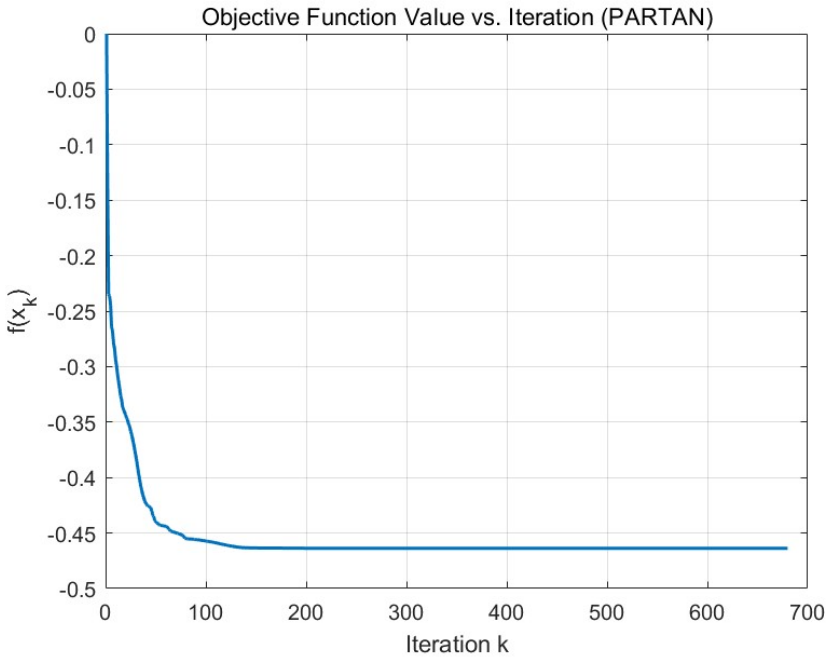

Figure 12: For $\mu = 1000$, the function values in PARTAN versus iteration k

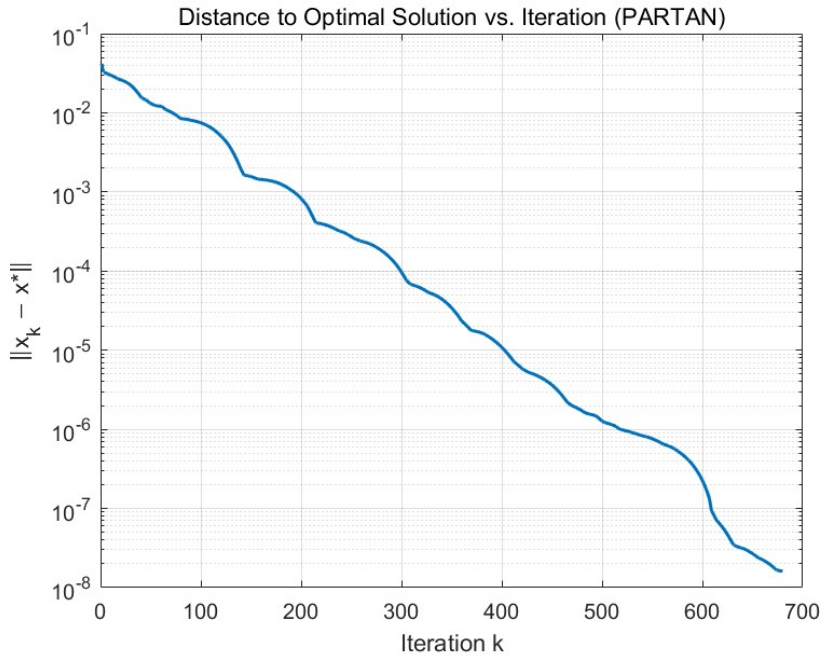

Figure 13: For $\mu = 1000$, the distance to optimal point in PARTAN versus iteration k