MATH/CS 5486 Homework 1

Submitted by: Vishnu Dutt Sharma

A. Analysis

$$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2 \tag{1}$$

where,

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad x \in [-10, 10] \times [-10, 10] \tag{2}$$

1. $\nabla f(x)$ and $\nabla^2 f(x)$

$$\begin{aligned} \nabla f(x) &= \begin{bmatrix} \frac{\partial}{\partial x_1} f(x) \\ \frac{\partial}{\partial x_2} f(x) \end{bmatrix} \\ &= \begin{bmatrix} 2(x_1 + 2x_2 - 7) \times 1 + 2(2x_1 + x_2 - 5) \times 2 \\ 2(x_1 + 2x_2 - 7) \times 2 + 2(2x_1 + x_2 - 5) \times 1 \end{bmatrix} \\ &= \begin{bmatrix} 10x_1 + 8x_2 - 34 \\ 8x_1 + 10x_2 - 38 \end{bmatrix} \end{aligned} \tag{3}$$

$$\begin{aligned} \nabla^2 f(x) &= \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} f(x) & \frac{\partial^2}{\partial x_1 \partial x_2} f(x) \\ \frac{\partial^2}{\partial x_2 \partial x_1} f(x) & \frac{\partial^2}{\partial x_2^2} f(x) \end{bmatrix} \quad (using\ the\ results\ for\ \nabla f(x)\ above) \\ &= \begin{bmatrix} 10 & 8 \\ 8 & 10 \end{bmatrix} \end{aligned} \tag{4}$$

2. For a differtiable function $f(x)$, $x^\star$ is called a stationary point if all partial derivatives (derivative for one dimensional case) are equal to zero at this point.
   At stationary point, we would have $\nabla f(x) = 0$. i.e.

$$\begin{aligned} \nabla f(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(x) \\ \frac{\partial}{\partial x_2} f(x) \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 10x_1 + 8x_2 - 34 \\ 8x_1 + 10x_2 - 38 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 10 & 8 \\ 8 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &= \begin{bmatrix} 34 \\ 38 \end{bmatrix} \\ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &= \frac{1}{100 - 64} \begin{bmatrix} 10 & -8 \\ -8 & 10 \end{bmatrix} \begin{bmatrix} 34 \\ 38 \end{bmatrix} \\ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &= \frac{1}{36} \begin{bmatrix} 36 \\ 108 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \end{aligned} \tag{5}$$

   Hence, the stationary point $(x^\star)$ is $x = [1, 3]^T$.

3. Yes, this point is a strict local minimum $(f(x^\star) = 0)$, as in the given domain this function does not have a lower or equal value.

   Theoretical Proof:
   From (3),

$$\begin{aligned} \nabla f(x^\star) &= \begin{bmatrix} 10x_1 + 8x_2 - 34 \\ 8x_1 + 10x_2 - 38 \end{bmatrix}_{(at\ x_1=1, x_2=3)} \\ &= \begin{bmatrix} 10 + 24 - 34 \\ 8 + 30 - 38 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \tag{6}$$

and from (4),

$$\nabla^2 f(x^\star) = \begin{bmatrix} 10 & 8 \\ 8 & 10 \end{bmatrix} \tag{7}$$

Checking if $\nabla^2 f(x^\star)$ is positive definite. Let $X = [x, y]^T$ be a point in domain of the objective function.

$$\begin{aligned} \begin{bmatrix} x & y \end{bmatrix} \nabla f(x^\star) \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 10 & 8 \\ 8 & 10 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ &= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 10x + 8y \\ 8x + 10y \end{bmatrix} \\ &= 10x^2 + 16xy + 10y^2 = 2(5^2 + 8xy + 5y^2) \\ &= 2(x^2 + y^2 + 4x^2 + 4xy + 4y^2) = 2(x^2 + y^2 + (2x + 2y)^2) \\ &= 2x^2 + 2y^2 + 2(2x + 2y)^2 > 0 \;\; \forall [x, y]^T \in \mathbb{R} \setminus 0 \end{aligned} \tag{8}$$

Thus $\nabla^2 f(x^\star)$ is positive definite.

From the results above, we know that (a) $\nabla^2 f(x)$ is continuous in an open neighbourhood of $x^\star$ (as it is constant throughout), (b) $\nabla f(x^\star) = 0$, and (x) $\nabla^2 f(x^\star)$ is positive definite. As it satisfies *second-order sufficient conditions (Theorem 2.4)* [1], $x^\star$ is a strict local minimizer of $f(x)$.

4. Yes, in the given domain, this point $(x^\star)$ is the global minimize as no other point has a lower value value this point.
   Theoretical Proof:
   From (4), $\nabla^2 f(x) > 0 \;\; \forall x \in [-10, 10] \times [-10, 10]$, $f(x)$ is a convex function. Also, since $\nabla f(x)$ exists in the domain, it is differentiable over this domain. As we have already proven that $x^\star$ is a stationary point (also a strict local minimizer) of $f(x)$, according to *Theorem 2.5* [1], $x^\star$ is also global minimizer of $f(x)$.

B. Computation

1. Results for Steepest Descent and Newton's Method are shown in Figure 1 and Figure 2 respectively. The status of convergence is also shown in the title. The implementations are submitted as file **hw1.py**.
   Note: In implementation of steepest gradient, it was found that if non-absolute value of error is used, then the algorithm may converge even though x_k is away from the $x^\star$ due to $f(x_k) - f(x_{k+1})$ going negative (due to the large value of α_k). As absolute value of error provided slightly better results in part 2, absolute value has been used to compare with the tolerance to check for convergence.

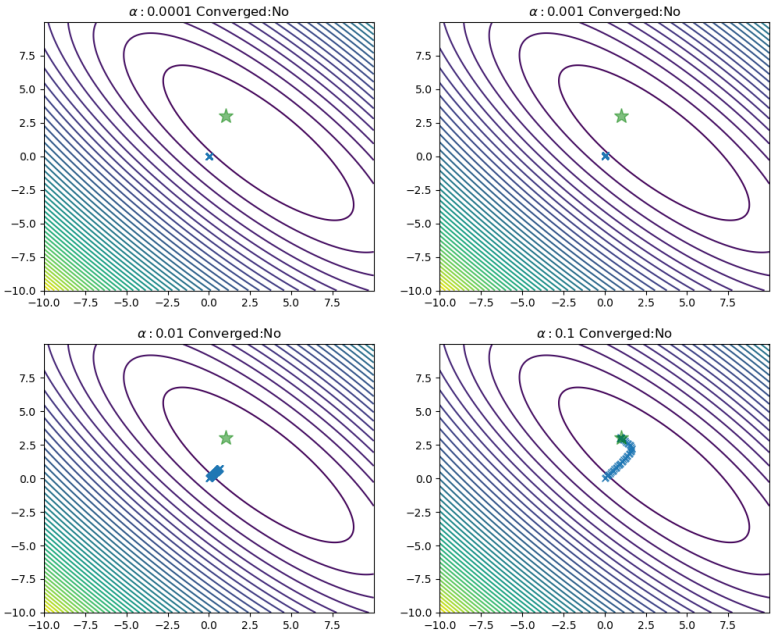

Figure 1: Iterations for steepest decent for different values of α

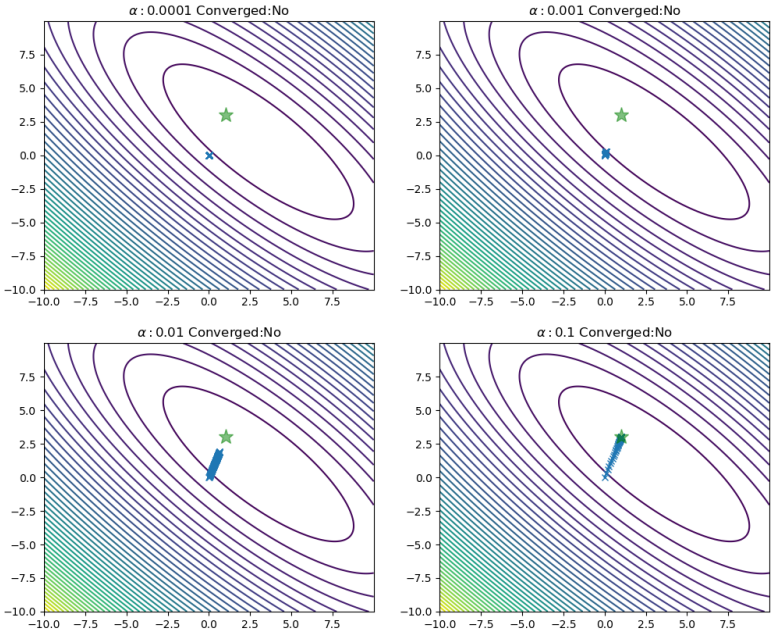

Figure 2: Iterations for Newton's Method for different values of α

2. For different values of α_k i.e.
   (a) $\alpha_k = 1$, (referred to as mode *const*)
   (b) $\alpha_k = ||\nabla f(x_k)||$, (referred to as mode *norm*) and
   (c) $\alpha_k = 1/(k+1)$, (referred to as mode *count*),
   the results are shown in Figure 3.
   As observed in the figure, the algorithm fails to converge,for all the cases but the reasons vary.

   i. $\alpha_k = 1$
   Algorithm fails to converge and exits after 100 iterations. It reaches near the stationary point and bounces back and forth around it. The reason is that after reaching near $x^\star$, the algorithm oversteps it due to comparatively high value of α_k. We also see the error may become negative due the the large step-length α_k (hence using absolute values of error for comparison with the tolerance in code).
   Final error in this case is 2.1693.
   If raw value of error is used instead of absolute value, algorithm stops after 3 iterations with error= -2.31183.

   ii. $\alpha_k = ||\nabla f(x_k)||$
   Algorithm fails to converge and exits due to x_{k+1} going out of domain, after 1 step. In this case, the expression $\alpha_k p_k$ evaluates to $\nabla f(x)$. Hence, x_1 evaluates to $\nabla f(x_0) = [-34, -38]^T$, which sends the next update point x_{k+1} out of the domain.
   Final error in this case is -20736.0.
   If raw value of error is used instead of absolute value, algorithm goes out of domain after 1 steps with error= -20736.0.

   iii. $\alpha_k = 1/(k+1)$
   Algorithm fails to converge and exists after 100 iterations. In this case, the step length reduces with time. Although this is a better strategy than $\alpha_k = 1$, as it would take smaller step near $x^\star$, this strategy takes more time to reach the optimum due to diminishing step sizes. It can also be observed from the 3 (a) and (c). Examination of the steps shows that in this case also algorithm bounces around $x^\star$.
   Final error in this case is -0.00040329. It suggests that the algorithm overstepped $x^\star$.
   If raw value of error is used instead of absolute value, algorithm stops after 26 steps with error -0.0005093.

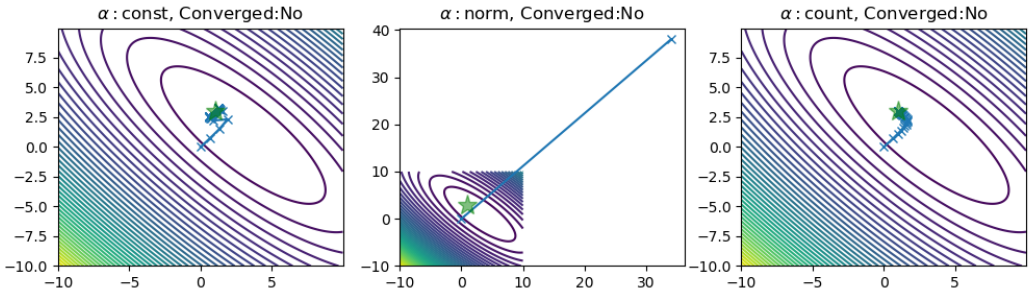

Figure 3: Optimization iterations for steepest descent with α_k= (a) 1, (b) $\nabla f(x_k)$, and (c) $1/(k+1)$, from left to right

3. For Newton's line search method, setting $\alpha_k = 1$, makes the algorithm converge in 1 step (in implementation it may look like 2 steps: one step to reach minimizer, another to confirm that this point is a minimum by checking the error). Figure 4 shows the iterations. This quick convergence is observed as the update by Newton's method here is similar to the closed form solution for finding the stationary point(see Eq. 5). Error in this case is 0.

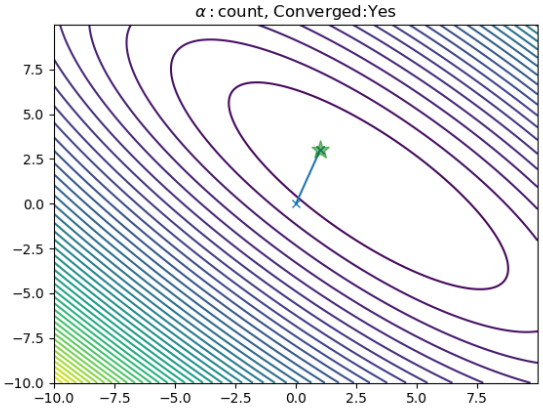

Figure 4: Optimization iterations for Newton's line search method with $\alpha = 1$

.

Zoom Methods:

Using the Zoom methods with $C_1 = 10^{-4}$, $C_2 = 0.9$, and $\alpha_{max} = 5$ improves the performance of Steepest Descent by reducing the number of iterations to reach the minimizer. When absolute value of error is used to compare with tolerance, the algorithm keeps bouncing around $x^\star$. In earlier trials, after comparing the unprocessed (non-absolute) error with tolerance, the algorithm converges at fixed value of $\alpha = 0.1$ after 38 iterations with error -0.0073236. However, with zoom the algorithm converges after 31 iterations with error 2.98443e-12.

The optimization trajectory is shown in Figure 5

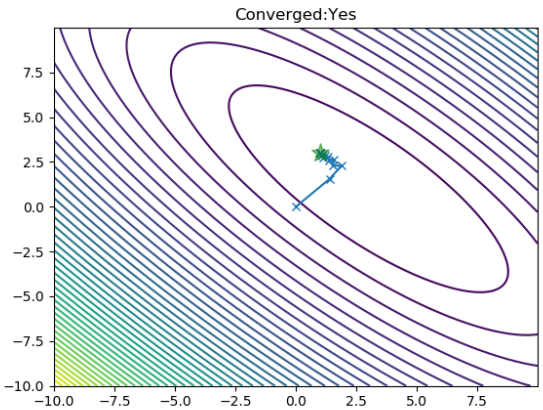

Figure 5: Optimization iterations for Steepest Descent with Zoom method

.

The number of iterations for Newton's method increase with Zoom method (10 iterations) as compared to when $\alpha = 1$, however it still converges faster than steepest descent. The optimization trajectory is shown in Figure 6

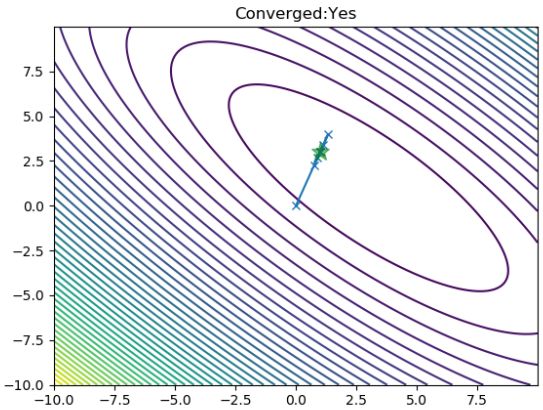

Figure 6: Optimization iterations for Newton's method with Zoom method

.

References

[1] Jorge Nocedal and Stephen Wright. Numerical optimization. Springer Science & Business Media, 2006.