Numerical Optimization, 2021 Fall
Homework 7 Solution

1 Armijo Line Search Condition

Consider the optimization problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \tag{1}$$

where $f \in C^2$. In addition, we have the following assumptions on f:

1. f is L-smooth
2. f is bounded below over $\boldsymbol{x} \in \mathbb{R}^n$

The Armijo line search condition is

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + c_1 \alpha_k \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k \tag{2}$$

with $c_1 \in (0, 1)$.

(1) Show that for a sufficiently small step size α_k, this condition must hold (provide the expression of this step size). [15pts]

For an arbitrary α, we achieve an upper bound of $f(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k)$ by the Lipschitz continuity of ∇f

$$f(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \alpha\boldsymbol{d}_k\rangle + \frac{\alpha^2}{2} L\|\boldsymbol{d}_k\|_2^2 \tag{3}$$

By making α further satisfying the sufficient decrease condition, it holds

$$f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \alpha\boldsymbol{d}_k\rangle + \frac{\alpha^2}{2} L\|\boldsymbol{d}_k\|_2^2 \leq f(\boldsymbol{x}_k) + c_1\langle \nabla f(\boldsymbol{x}_k), \alpha\boldsymbol{d}_k\rangle \tag{4}$$

Therefore, for any $\alpha \in \left[0, \frac{2(c_1-1)\langle\nabla f(\boldsymbol{x}_k), \boldsymbol{d}_k\rangle}{L\|\boldsymbol{d}_k\|_2^2}\right]$, the Armijo line search condition is satisfied. And then, the backtracking procedure must end up with

$$\alpha_k \geq 2\gamma\frac{(c_1-1)\langle\nabla f(\boldsymbol{x}_k), \boldsymbol{d}_k\rangle}{L\|\boldsymbol{d}_k\|_2^2} \tag{5}$$

where γ is the decay constant of the line search.

(2) Let $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{d}_k$ with $\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k)$, show that $\|\nabla f(\boldsymbol{x}_k)\|_2 \to 0$. [15pts]

Combining the Armijo line search condition with (5) and substituting $\boldsymbol{d}_k$ by $-\nabla f(\boldsymbol{x}_k)$ gives

$$\begin{aligned} f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) &= f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \alpha_k\boldsymbol{d}_k) \\ &\geq -c_1\alpha_k\langle\nabla f(\boldsymbol{x}_k), \boldsymbol{d}_k\rangle \\ &\geq c_1\frac{2\gamma(1-c_1)\langle\nabla f(\boldsymbol{x}_k), \boldsymbol{d}_k\rangle}{L\|\boldsymbol{d}_k\|_2^2}\langle\nabla f(\boldsymbol{x}_k), \boldsymbol{d}_k\rangle \\ &= \frac{2\gamma c_1(1-c_1)\|\nabla f(\boldsymbol{x}_k)\|_2^2}{L} \end{aligned} \tag{6}$$

By rearranging (6) and summing up both sides from 1 to K, we have

$$\sum_{k=0}^{K} \|\nabla f(\boldsymbol{x}_k)\|_2^2 < \frac{L}{c_1(1-c_1)} \sum_{k=0}^{K} (f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})) \leq \frac{L}{2\gamma c_1(1-c_1)} \left(f(\boldsymbol{x}^0) - \underline{f}\right) \tag{7}$$

Thus, $\|\nabla f(\boldsymbol{x}_k)\|_2 \to 0$ as $K \to \infty$.

2 Exact Line Search

Consider minimizing the following quadratic function

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} \tag{8}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is positive definite and $\boldsymbol{b} \in \mathbb{R}^n$. Let $\boldsymbol{d}_k$ be a descent direction at the kth iteration. Suppose that we search along this direction from $\boldsymbol{x}_k$ for a new iteration, and the line search is exact.

(1) Please find the step size α with respect to $\boldsymbol{d}_k$. [15pts]

Keep in mind that our goal is to choose $\alpha > 0$ to minimize $f(\boldsymbol{x}_{k+1})$. Toward achieving this, we let

$$\begin{aligned} g(\alpha) &= f(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k) \\ &= \frac{1}{2}(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k)^T\boldsymbol{Q}(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k) - \boldsymbol{b}^T(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k) \end{aligned} \tag{9}$$

It should be noticed that $g(\alpha)$ is quadratic and convex since $\boldsymbol{Q}$ is positive definite. Then,

$$g(\alpha) = a\alpha^2 + d\alpha + c \tag{10}$$

with

$$\begin{aligned} a &= \frac{1}{2}\boldsymbol{d}_k^T Q\boldsymbol{d}_k \\ d &= (Q^T\boldsymbol{x}_k - \boldsymbol{b})^T\boldsymbol{d}_k \\ c &= \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}_k - \boldsymbol{b}^T\boldsymbol{x}_k \end{aligned} \tag{11}$$

Therefore, $-\frac{d}{2a} = \text{argmin}_{\alpha>0}\, g(\alpha)$. This leads to

$$\alpha = \frac{(\boldsymbol{b} - \boldsymbol{Q}^T\boldsymbol{x}_k)^T\boldsymbol{d}_k}{\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{d}_k} \tag{12}$$

(2) If $\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k)$, write down the step size α. [5pts]

$$\begin{aligned} \alpha &= \frac{(\boldsymbol{Q}^T\boldsymbol{x}_k - \boldsymbol{b})^T\nabla f(\boldsymbol{x}_k)}{\nabla f(\boldsymbol{x}_k)^T\boldsymbol{Q}\nabla f(\boldsymbol{x}_k)} \\ &= \frac{\|\nabla f(\boldsymbol{x}_k)\|_2^2}{\nabla f(\boldsymbol{x}_k)^T\boldsymbol{Q}\nabla f(\boldsymbol{x}_k)} \end{aligned} \tag{13}$$

Hint: Consider how would you solve the following one-dimensional minimization problem

$$\min_{\alpha>0} \quad f(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k) \tag{14}$$

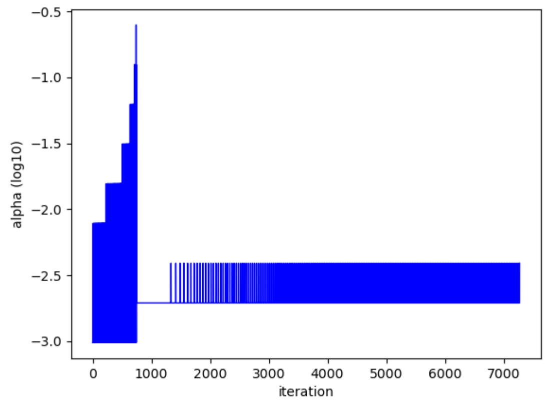
3 Coding

Please solve the following problem using *Gradient Descent* and *Newton's Method* respectively in combination with *Armijo Backtracking Line Search*. [50pts]

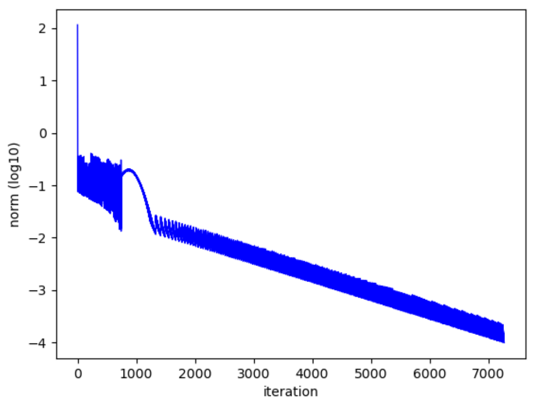
$$\min_{x_1,x_2} 100\left(x_2 - x_1^2\right)^2 + (1-x_1)^2 \tag{15}$$

Use two starting points ($\boldsymbol{x}^0 = [1.2, 1.2]^T$ and $\boldsymbol{x}^0 = [-1.2, 1]^T$) to initiate your algorithms. For each case, draw the curves of the step size α^k and $\|\nabla f(\boldsymbol{x}^k)\|_\infty$ of the two algorithms, with respect to the iteration number k.
Your program may be implemented with Python or MATLAB. Please paste your curves down below, then put your code in an independent file and submit it to Blackboard alongside the PDF.
Hint: You may set initial step size $\alpha_0 = 1$, constant $c_1 = 10^{-4}$, and termination condition as $\|\nabla f(\boldsymbol{x}^k)\|_\infty \leq 10^{-4}$.

We can approximate from the programm that **the minimum value is 0, obtained at (1, 1)**.

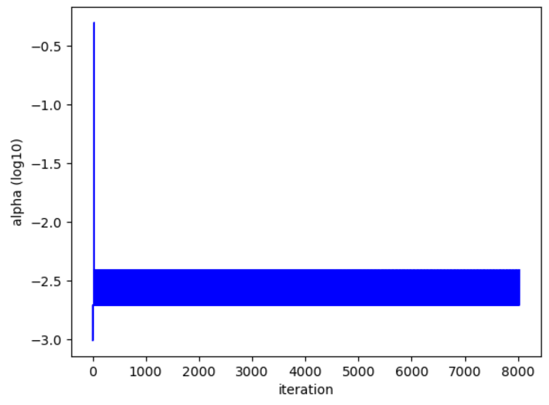


(a) α

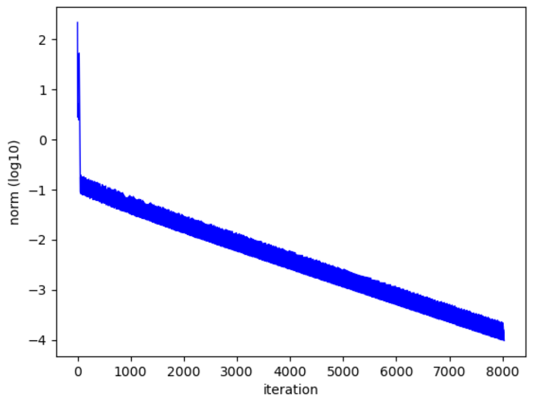


(b) $\|\nabla f(\boldsymbol{x}^k)\|_\infty$

Figure 1: Results of Gradient Descent with $\boldsymbol{x}^0 = [1.2, 1.2]^T$.

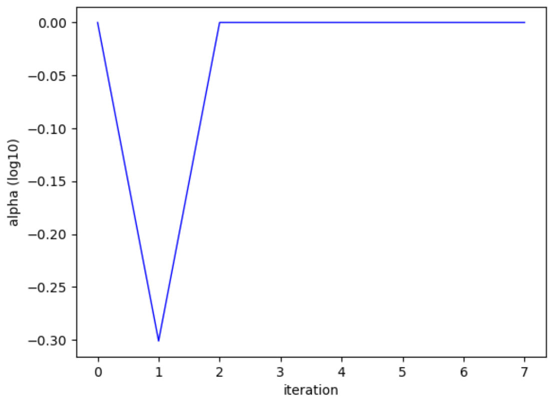


(a) α

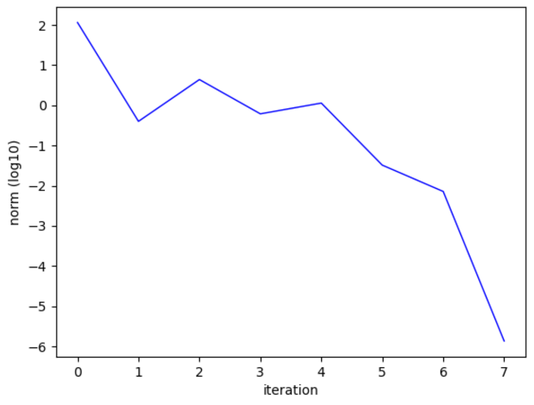


(b) $\|\nabla f(\boldsymbol{x}^k)\|_\infty$

Figure 2: Results of Gradient Descent with $\boldsymbol{x}^0 = [-1.2, 1]^T$.

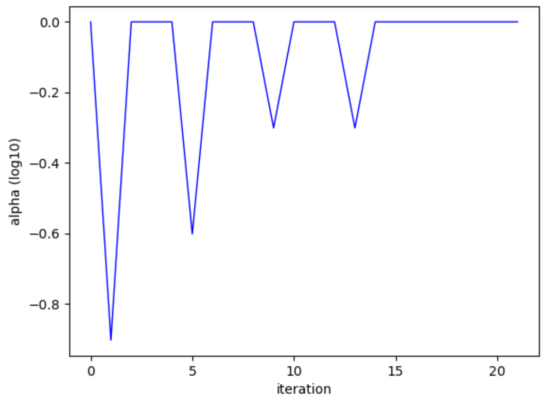


(a) α

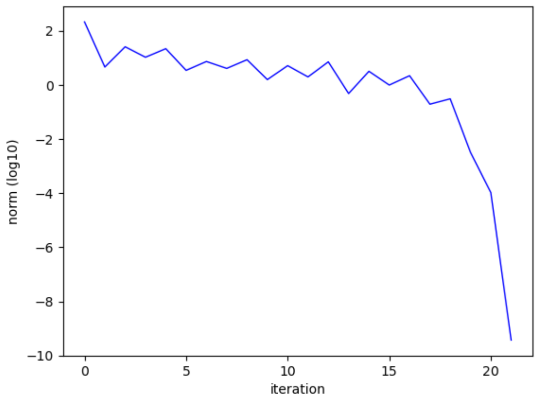


(b) $\|\nabla f(\boldsymbol{x}^k)\|_\infty$

Figure 3: Results of Newton's Method with $\boldsymbol{x}^0 = [1.2, 1.2]^T$.



(a) α



(b) $\|\nabla f(\boldsymbol{x}^k)\|_\infty$

Figure 4: Results of Newton's Method with $\boldsymbol{x}^0 = [-1.2, 1]^T$.

Code:

```
import numpy as np
import argparse
from numpy import linalg as LA
import matplotlib.pyplot as plt


# Calculate the objective function
def calcObj(point):
   return 100 * ((point[1] - point[0] ** 2) ** 2) + (1 - point[0]) ** 2


# Calculate the gradient at current point
def calcJacobian(point):
   dx1 = -2 * (1 - point[0]) + 200 * (point[1] - point[0] ** 2) * (-2 * point[0])
   dx2 = 200 * (point[1] - (point[0] ** 2))
   return np.array([dx1, dx2])


# Calculate the Hessian matrix
def calcHessian(x):
   return np.array([[-400 * x[1] + 1200 * x[0] ** 2 + 2, -400 * x[0]],
                    [-400 * x[0],              200]])


# Find alpha^k with backtracking
def findAlpha(point, direction):
   alpha = 1
   c = 1e-4
   rho = 0.5
   f = calcObj(point)
   grad = calcJacobian(point)
   while calcObj(point + alpha * direction) > f + c * alpha * np.dot(grad.T,
       direction):
      alpha = rho * alpha
   return alpha


# Visualize path
def drawPath(points):
   # Unzip points
   x1s = np.array([p[0] for p in points])
   x2s = np.array([p[1] for p in points])
   max_x1, max_x2 = max(x1s) + 0.1, max(x2s) + 0.1
   min_x1, min_x2 = min(x1s) - 0.1, min(x2s) - 0.1
   step = 0.001
   X1, X2 = np.meshgrid(np.arange(min_x1, max_x1, step), np.arange(min_x2, max_x2,
       step))
   F = 100 * ((X2 - X1 ** 2) ** 2) + (1 - X1) ** 2

   plt.contourf(X1, X2, F, 50); plt.colorbar(); plt.contour(X1, X2, F)
   plt.plot(x1s, x2s, color="black", linewidth=1, linestyle="-")
   plt.xlabel("x1"); plt.ylabel("x2")
   plt.show()


# Visualize alpha/gradient updates
def drawUpdates(pairs, ylabel):
   i = np.array([p[0] for p in pairs])
   value = np.array([p[1] for p in pairs])

   plt.plot(i, np.log10(value), color="blue", linewidth=1, linestyle="-")
   plt.xlabel("iteration"); plt.ylabel(ylabel)
   plt.show()


def main(x1, x2, method):
   x = np.array([x1, x2]) # The initial point
   tol = 1e-4           # Tolerance before exiting
   epochs = 100000      # Maximum epochs

   x_f_record = []
   alpha_record = []
   norm_record = []
   for i in range(epochs):
      f = calcObj(x)
      grad = calcJacobian(x)
      if method == "gd":
         direction = - grad
      elif method == "newton":
         direction = - np.dot(np.linalg.inv(calcHessian(x)), calcJacobian(x))
      else:
         raise AssertionError("Method undefined!")
      alpha = findAlpha(x, direction)      # Learning rate

      x_f_record.append([x, f])
      norm_record.append([i, LA.norm(grad, np.inf)])
      alpha_record.append([i, alpha])

      print("Iteration:", i, x_f_record[i])
      if LA.norm(grad, np.inf) < tol:
         break
```

```
        else:
            x = x + np.dot(alpha, direction)

    print("With " + method + ", the minimum is: {0:.4} at point ({1:.4}, {2:.4})"
            .format(x_f_record[-1][1], x_f_record[-1][0][0], x_f_record[-1][0][1]))
    drawPath([x_f[0] for x_f in x_f_record])
    drawUpdates(norm_record, ylabel="norm (log10)")
    drawUpdates(alpha_record, ylabel="alpha (log10)")


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Initial point.')
    parser.add_argument('method', type=str, help='solving method: gd or newton')
    parser.add_argument('x1', type=float, help='value for x1')
    parser.add_argument('x2', type=float, help='value for x2')
    args = parser.parse_args()
    main(args.x1, args.x2, method=args.method)
```