HOMEWORK 1
MAE 206A- OPTIMIZATION METHODS
INSTRUCTOR: PROF. SOLMAZ S. KIA

Problem 1. *This problem is a practice on use of necessary and sufficient conditions to determine minimizers of an unconstrained optimization problem.*

Consider

$$f(x) = 1.5x_1^2 + x_2^2 - 2x_1\,x_2 + 2x_1^3 + 0.5\,x_1^4,$$

as the cost function of an unconstraint optimization problem.

(a) Find all the stationary points of this function.
(b) Determine the type of each stationary point (local minimum, global minimum, maximum, saddle point).
(c) Plot the contours of f in $x_1 \in [-4, 2]$ and $x_2 \in [-4, 2]$. Display the stationary points also on this plot.

Problem 2. *This problem is a practice on use of necessary and sufficient conditions to determine minimizers of an unconstrained optimization problem.*

Least squares problem:

Given $A \in \mathbb{R}^{m\times n}$ with linearly independent columns, i.e., $\text{rank}(A) = n$, and $b \in \mathbb{R}^{m\times 1}$, verify that $x^\star$ in

$$x^\star = \underset{x\in\mathbb{R}^n}{\text{argmin}} \|Ax - b\|^2,$$

is $x^\star = (A^\top A)^{-1}A^\top b$. Explain why $x^\star$ is a unique strict global minimum of this optimization problem. You need to explain why $A^\top A$ is invertible.

Hint: $\|Ax - b\|^2 = (Ax - b)^\top (Ax - b)$

Problem 3. *Quadratic optimization problems are an important class of optimization problems which appear both in practice and various stages of some of the nonlinear optimization algorithms. In this problem you obtain a closed form solution for exact stepsize of the steepest descent algorithm to solve positive definite quadratic optimization problems.*

Consider the unconstrained optimization problem below

$$x^\star = \text{argmin}(\frac{1}{2}\,x^\top Q\,x + b^\top x + c),$$

where $x \in \mathbb{R}^n$ is the decision variable and $Q > 0$. Write down the steepest descent algorithm for this problem. Show that the stepsize $\alpha_k = \text{argmin}\, f(x_k - \alpha_k \nabla f(x_k))$

is

$$\alpha_k = \frac{\nabla f(x_k)^\top \nabla f(x_k)}{\nabla f(x_k)^\top Q \nabla f(x_k)}, \quad \text{where } \nabla f(x_k) = Q x_k + b.$$

Problem 4. *This problem explores the use of steepest descent algorithm and Matlab fmin function to solve an unconstrained optimization problem. The emphasis is on the role of stepsize on convergence of the algorithm.*

Let the cost function of the unconstrained optimization problem of interet be

$$f(x) = 2x_1^2 + x_1 x_2 + x_2^2 + x_2 x_3 + x_3^2 - 6x_1 - 7x_2 - 8x_3 + 9.$$

Recall that the steepest descent algorithm is $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$, $\alpha_k \in \mathbb{R}_{>0}$.

(a) Using $x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, write out the first iteration of the steepest descent algorithm and obtain the optimum value for α_0. What is the value of x_1 if you implement α_0? Verify that $f(x_1) < f(x_0)$?

(b) Write a MATLAB code to find the minimizer of $f(x)$ using steepest descent algorithm with starting point of $x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and using
- $\alpha_k = \operatorname{argmin} f(x_k - \alpha_k \nabla f(x_k))$.
- constant $\alpha = 0.1$.
- constant $\alpha = 0.5$.
- constant $\alpha = 1$.

Use $\|x_{k+1} - x_k\| \leq 10^{-6}$ as the stoping condition for your algorithm.

(c) How many steps it takes for the algorithm to converge for each choices of the stepsizes above?

(d) Use MATLAB's fmin function to solve the problem. How many steps this algorithm takes to converge?

Problem 5. *This problem demonstrates that Newton method $x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$ does not necessarily have global convergence property. Recall that global convergence means that starting from any initial conditions the algorithm converges to a minimizer.*

Consider an optimization problem with cost function below:

$$f(x) = \frac{1}{4}x^4 - x^2 + 2x$$

(a) Find the stationary point(s) of f.

(b) Obtain the minimizer(s) of this problem.

(c) Write a MATLAB code to solve this problem using Newton's method starting from initial conditions given below. For each initial condition run your code

for $N = 10$ steps. Complete the table below (write your answers up to 9 digit accuracy):

(d) Does Newton algorithm have global convergence for this problem?

k	x_k	x_k	x_k	x_k	x_k
0	-1.000000000	0.000000000	0.10000000	1.000000000	2.000000000
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

Note: this problem is an example to motivate the *guarded* Newton method (a modified Newton method)

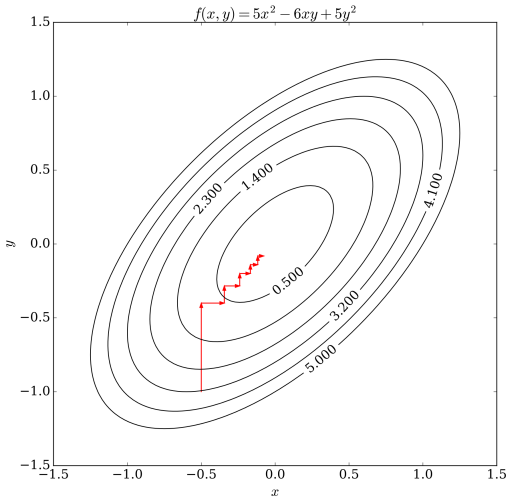
- Let's not take full Newton steps
- Introduce a step size $x_{k+1} = x_k - \alpha_k \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$.

The material below is just FYI and is not part of your homework assignment.

– – – – – – – – – – – A fun fact – – – – – – – – – – –

Theorem ("the zig-zag theorem") Let $\{x_k\}$ be the sequence generated by steepest descent algorithm. Then, for all k, $x_{k+1} - x_k$ is orthogonal to $x_{k+2} - x_{k+1}$.

(Optional) Prove this Theorem. (recall that two vectors a and b are orthogonal if and only if $a^\top b = b^\top a = 0$).



– – – – – – – – – – – – – – – Sample Code – – – – – – – – – – – – – – –

Find the minimizer of

$$f(x) = -\log(1 - x_1 - x_2) - \log x_1 - \log x_2$$

The minimizer of this function is

$$x^\star = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}, \quad f(x^\star) = 3.295836867.$$

You can use the following MATLAB codes to compute the gradient and Hessian of this function:

```
>> syms x1 x2;
>> f=-log(1-x1-x2)-log(x1)-log(x2);
>> gradf=gradient(f)

gradf =
-1/(x1 + x2 -1) -1/x1
-1/(x1 + x2 -1) -1/x2

>> Hf=hessian(f)
```

```
Hf =
[ 1/(x1 + x2 -1)^2 + 1/x1^2, 1/(x1 + x2 -1)^2]
[ 1/(x1 + x2 -1)^2, 1/(x1 + x2 -1)^2 + 1/x2^2]

>> ezsurf(f,[0,.5,0,.5])
```

A simple code for Newton's iteration:

```
syms x1 x2;
f=-log(1-x1-x2)-log(x1)-log(x2);
gradf=gradient(f);
Hf=hessian(f);
N=10; %number of Newton iterations
x=zeros(2,N); fval=zeros(N,1); er=zeros(N,1);
x(:,1)=[.8;.1]; %initial point
for k=1:N-1
fval(k)=subs(f,{x1,x2},{x(1,k),x(2,k)});
er(k)=norm([1/3;1/3]-x(:,k));
x(:,k+1)=x(:,k)-subs(Hf,{x1,x2},{x(1,k),x(2,k)})\subs(gradf,{x1,x2},{x(1,k),x(2,k)});
end
fval(k+1)=subs(f,{x1,x2},{x(1,k+1),x(2,k+1)});
er(k+1)=norm([1/3;1/3]-x(:,k+1));
format long
output=[(1:N)' x' er fval]
```