Friedrich-Alexander-Universität Erlangen-Nürnberg
CML: Control, Machine Learning, and Numerics
Assignment 1
Due Date: 6:00pm, May 18, 2023. Assignment submission address: ycsong.math@gmail.com

1. Detect and classify (with a short justification) the stationary point of $f : \mathbf{R}^2 \to \mathbf{R}$ with $f(x_1, x_2) = x_1^2 - x_2^2$.

2. Consider the gradient descent method.
   Show with the help of an example that if the step length α is badly chosen, the gradient descent might diverge. [**Hint:** The iteration for gradient descent is given by $x_{k+1} = x_k - \alpha \nabla f(x_k)$. Consider the 1D case with a quadratic function for instance.]

3. In this exercise we investigate that Newton is really a *local* method and does not converge from arbitrary starting points $x_0 \in \mathbf{R}$. Let $f : \mathbf{R} \to \mathbf{R}$ with $f(x) = \sqrt{x^2 + 1}$.

   (1) Investigate for which starting points x_0 Newton does converge and for which starting points Newton will diverge.

   (2) What are possible procedures to globalize Newton's method and to increase the convergence radius?

4. Consider the following optimization problem

$$\min_{x_1, x_2} f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$$

   It is easy to see that the exact solution to the above problem is $(1, 1)^T$.

   Implementing the following methods using Matlab (or other software you prefer) to solve the above optimization problem:

   - Gradient descent method with constant step sizes: 0.1, 0.01, 0.001, 0.0001.
   - Gradient descent method with backtracking line search.
   - Classic Newton method (i.e. step size is 1).
   - Newton method with backtracking line search.

   Parameters setting-up: initial value $x^0 = [0, 0]^T$, maximum iteration: 50000, stoping criterion: norm of gradient $< 10^{-8}$. (You can verify your result by comparing the output with the exact solution)

   Please report all the results of the four methods listed above, and discuss briefly what you can conclude from the results. (For backtracking line search, please choose the parameters carefully such that the algorithm converges as fast as possible)

Friedrich-Alexander-Universität Erlangen-Nürnberg
CML: Control, Machine Learning, and Numerics
Solution of Assignment 1

1. The stationary point $(0,0)$ (check yourself why this is the only stationary point and how to obtain it) is a saddle point.

   - Justification 1: Following x_1 we identify a minimum at $(0,0)$ and following x_2 we identify a maximum at $(0,0)$. Consequently, the curvatures are different and thus there is a saddle point.
   - Justification 2: Compute the Hessian:

     $$\nabla^2 f(0,0) = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}$$

     The Hessian has two different eigenvalues and therefore, a saddle point can be classified. As alternative, one can compute the determinant: since $\det\nabla^2 f(0,0) = -4 < 0$, a saddle point is obtained.

2. Choose for example $f(x) = \frac{x^2}{2}$. Then $\nabla f(x) = f'(x) = x$. The general iteration reads:

   $$x_{k+1} = x_k + \alpha_k p_k$$

   where $p_k = -\nabla f(x_k)$. Here, $p_k = -x_k$. This yields:

   $$x_{k+1} = x_k - \alpha_k x_k = x_k(1 - \alpha_k).$$

   If $(1 - \alpha_k) \geq 1$, gradient descent does not converge. This is intuitively clear because the minimum is 0 and the sequence $\{x_k\}_k$ will produce increasing x_k with

   $$x_0 < x_1 < x_2 < \dots$$

   if $(1 - \alpha_k) > 1$ or just stagnates

   $$x_0 = x_1 = x_2 = \dots$$

   for $(1 - \alpha_k) = 1$.

3. - Answer to (1): We have:

     $$\nabla f(x) = \frac{x}{\sqrt{x^2+1}}, \quad \nabla^2 f(x) = \frac{1}{(\sqrt{x^2+1})^{3/2}}$$

     The Newton direction reads:

     $$\nabla^2 f(x_k) p_k = -\nabla f(x_k)$$

     Inserting the specific expressions:

     $$\frac{1}{(\sqrt{x_k^2+1})^{3/2}} p_k = -\frac{x_k}{\sqrt{x^2+1}}$$

     Thus for the update we have:

     $$p_k = -x_k((x_k)^2 + 1).$$

     The second step of Newton's method is to form the new iterate:

     $$x_{k+1} = x_k + p_k = x_k - x_k((x_k)^2 + 1) = -(x_k)^3$$

     For $|x_0| < 1$ we have (very) fast convergence of Newton's method. For your own studies, you might verify the rate of convergence for this example (normally in the literature Newton's method is considered to be quadratically convergent). In this example, however, Newton converges even with rate 3!! For $|x_0| = 1$ we have an oscillating behavior between $-x_0$ and x_0. For $|x_0| > 1$ we observe divergence. All these results are also intuitively clear since the minimum of $f(x)$ is $(0,1)$. Thus, the sequence $\{x_k\}_k$ must converge to 0 and this is obviously not the case for $|x_0| \geq 1$.

- Answer to (2) (one possibility out of many - but extremely naive for this example). A classical procedure to increase Newton's convergence radius (i.e., to globalize Newton's method) is to add a line search parameter α_k. Thus the iteration reads:

$$x_{k+1} = x_k + \alpha_k p_k = x_k - \alpha_k x_k((x_k)^2 + 1) = (1 - \alpha_k)x_k - \alpha_k(x_k)^3.$$

Choose for example $\alpha_k = 0.5$. Then:

$$x_{k+1} = 0.5x_k - 0.5(x_k)^3.$$

Choose for instance now as $x_0 = 1.5$. Then we obtain for the first iterate:

$$x_1 = -0.93750.$$

Compute for some further steps for the second and so forth. Here we see that indeed $x_0 = 1.5$ works. Consequently, we see that in comparison to the standard Newton method with the 'natural' step length $\alpha_k = 1$, we could increase the convergence radius from $|x_0| < 1$ to (at least) $|x_0| = 1.5$.

4. Code for gradient descent method

```
function gradient_descent_assignment
f=@(x) 100*(x(2)-x(1)^2)^2+(x(1)-1)^2;
fp=@(x) [-400*(x(2)-x(1)^2)*x(1)+2*(x(1)-1); 200*(x(2)-x(1)^2)];
[x,xk]=SteepestDescent(f,fp,[0;0]);
F=@(x,y) 100*(y-x.^2).^2+(x-1).^2; % for plotting purposes
Fp1=@(x,y) -400*(y-x.^2).*x+2*(x-1);
Fp2=@(x,y) 200*(y-x.^2);
[X,Y]=meshgrid(-1.5:0.1:1.5,-0.5:0.1:1.5);
contour(X,Y,F(X,Y),50)
hold on
quiver(X,Y,Fp1(X,Y),Fp2(X,Y),5)
plot(xk(1,:),xk(2,:),'-o')
hold off
end
```

```
function [x,xk]=SteepestDescent(f,fp,x0,tol,maxiter,tau,be,alinit)
if nargin<8, alinit=1; end
if nargin<7, be=0.5; end
if nargin<6, tau=0.99; end
if nargin<5, maxiter=40000; end
if nargin<4, tol=1e-8; end
x=x0;
xk=x0;
p=-feval(fp,x);
k=0;
while norm(p)>tol && k<maxiter
al=alinit;
while feval(f,x+al*p)>feval(f,x)-al*be*p'*p
al=tau*al;
end
x=x+al*p;
% x=x+0.01*p;  %%constant stepsize
```

```
p=-feval(fp,x);
k=k+1;
xk(:,k+1)=x;
end
niter=k
solution=x
end
```

Code for Newton method:

```
function Newton_assignment
f=@(x) 100*(x(2)-x(1)^2)^2+(x(1)-1)^2;
fp=@(x) [-400*(x(2)-x(1)^2)*x(1)+2*(x(1)-1); 200*(x(2)-x(1)^2)];
fpp=@(x) [-400*x(2)+1200*x(1)^2+2, -400*x(1); -400*(x(1)), 200];
[x,xk,k]=Newton(f,fp,fpp,[0;0]);
F=@(x,y) 100*(y-x.^2).^2+(x-1).^2; % for plotting purposes
[X,Y]=meshgrid(-1.5:0.1:1.5,-1:0.1:1.5);
contour(X,Y,F(X,Y),50,LineWidth=1)
hold on
plot(xk(1,:),xk(2,:),'-o')
hold off
end

function [x,xk,k]=Newton(f,fp,fpp,x0,tol,maxiter,tau,be,alinit)
if nargin<9, alinit=1; end
if nargin<8, be=0.1; end
if nargin<7, tau=0.6; end
if nargin<6, maxiter=50000; end
if nargin<5, tol=1e-8; end
x=x0;
xk=x0;
p=-feval(fpp,x)\feval(fp,x);
k=0;
while norm(feval(fp,x))>tol && k<maxiter
al=alinit;
while feval(f,x+al*p)>feval(f,x)-al*be*p'*p
al=tau*al;
end
x=x+al*p;
% x=x+p; %%%constant stepsize
p=-feval(fpp,x)\feval(fp,x);
k=k+1;
xk(:,k+1)=x;
end
end
```