SOLUTION 7 – Advanced Numerical Analysis

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

1 ► Quiz

Prof. D. Kressner
M. Steinlechner

a) false, take $A > 0$ and $\|x\| = (x^T A x)^{1/2}$

b) true, if $\pm\langle \nabla f(x), h \rangle = \langle \nabla f(x), \pm h \rangle \leq 0$, then $\langle \nabla f(x), h \rangle = 0$ for all h.

c) false, since x could be a saddle point

d) true, the global minimizer of $\alpha \mapsto f(x_0 - \alpha(x-b))$ is $\alpha^* = 1$ ($\nabla f(x) = (x-b)$)

e) false, see Problem 3

f) true, since $(f(x_k))$ is monotonically decreasing and bounded from below by $\min_x \{f(x) : f(x) \leq f(x_0)\}$ which exists since f is continuous and this set hence compact by assumption.

true, by continuity of f

g) false, the limit could still be a saddle point

h) true, by Theorem 4.8 and the remarks after, the sequence $(\nabla f(x_k))$ converges to zero. But we have to assume that f is twice continuously differentiable in order to ensure Lipschitz-continuity of ∇f on every compact domain

false,

2 ► Curvature condition for Quasi-Newton methods

In a Quasi-Newton method, after the $(k+1)$-th iteration, a symmetric positive definite matrix B_{k+1} is sought to satisfy $B_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$. A necessary condition for such a matrix to exist is

$$(\mathbf{x}_{k+1} - \mathbf{x}_k)^T (\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)) > 0. \qquad (\star)$$

(a) Proof that for a strictly convex function $(\star)$ always holds.

a) A function f is strictly convex iff ∇f is strictly monotonically increasing.
For a vector-valued function $g: D \to \mathbb{R}^n$, $D \subset \mathbb{R}^n$, strictly monotonically increasing is defined as

$$(\underline{x} - \underline{y})^T (g(\underline{x}) - g(\underline{y})) > 0 \quad \forall \underline{x}, \underline{y} \in D.$$

This is exactly condition $(\star)$ for $g(\underline{x}) := \nabla f(\underline{x})$. □

(b) Show that the *strong Wolfe curvature condition*

$$|\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k| \leq -c_2 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k$$

with $c_2 \in (0,1)$ implies $(\star)$ for $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$.

b) Inserting $\underline{x}_{k+1} = \underline{x}_k + \alpha_k \underline{p}_k$ into $(\star)$:

$$\alpha_k \underline{p}_k^T \left(\nabla f(\underline{x}_{k+1}) - \nabla f(\underline{x}_k)\right) = \alpha_k \left(\underline{p}_k^T \nabla f(\underline{x}_{k+1}) - \underline{p}_k^T \nabla f(\underline{x}_k)\right) \overset{?}{>} 0$$

Distinguish the following three cases:

① $\underline{p}_k^T \nabla f(\underline{x}_{k+1}) = 0$: $\Rightarrow \underbrace{-\alpha_k}_{<0} \underbrace{\underline{p}_k^T \nabla f(\underline{x}_k)}_{<0, \text{ as } \underline{p}_k \text{ is descent direction!}} > 0$ ✓

② $\underline{p}_k^T \nabla f(\underline{x}_{k+1}) > 0$: Then, strong Wolfe yields $\underline{p}_k^T \nabla f(\underline{x}_{k+1}) \leq -c_2 \underline{p}_k^T f(\underline{x}_k)$.

$$\Rightarrow \alpha_k \left(\underline{p}_k^T \nabla f(\underline{x}_{k+1}) - \underline{p}_k^T \nabla f(\underline{x}_k)\right) \geq \alpha_k \left(\underline{p}_k^T \nabla f(\underline{x}_{k+1}) + \frac{1}{c_2} \underline{p}_k^T \nabla f(\underline{x}_{k+1})\right)$$

$$= \underbrace{\alpha_k \left(1 + \frac{1}{c_2}\right)}_{>0} \underbrace{\left(\underline{p}_k^T \nabla f(\underline{x}_{k+1})\right)}_{>0} > 0$$ ✓

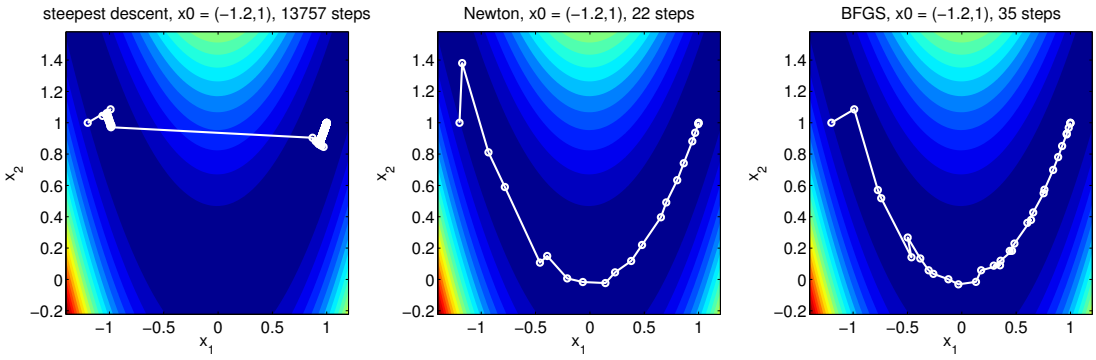
③ $\underline{p}_k^T \nabla f(\underline{x}_{k+1}) < 0$: From strong Wolfe: $-\underline{p}_k^T \nabla f(\underline{x}_{k+1}) \leq -c_2 \underline{p}_k^T f(\underline{x}_k)$

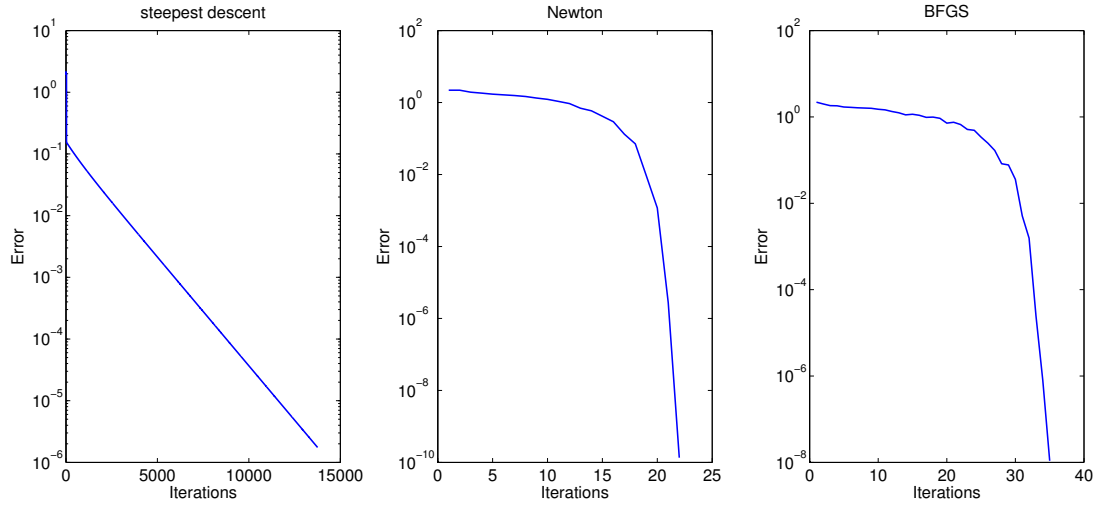
$$\Leftrightarrow \underline{p}_k^T \nabla f(\underline{x}_{k+1}) \geq c_2 \underline{p}_k^T f(\underline{x}_k)$$

$$\Rightarrow \alpha_k \left(\underline{p}_k^T \nabla f(\underline{x}_{k+1}) - \underline{p}_k^T \nabla f(\underline{x}_k)\right) \geq \alpha_k \underbrace{(c_2 - 1)}_{<0} \underbrace{\left(\underline{p}_k^T \nabla f(\underline{x}_k)\right)}_{<0, \text{ as } \underline{p}_k \text{ is descent direction}} > 0$$ ✓ □

3 ▶ Steepest descent vs. Newton vs. BFGS

Plots for the difficult starting guess $x_0 = (-1.2, 1)^T$:






```
1  function rosenbrock

3  % Rosenbrock function and derivatives
   f = @(x) 100*(x(2) - x(1)^2)^2 + (1 - x(1))^2;
5  df = @(x) [-400*(x(2) - x(1)^2 )*x(1) - 2*(1 - x(1)); 200*(x(2) - x(1)^2) ];
   ddf = @(x) [ -400*x(2) + 1200*x(1)^2 + 2, -400*x(1); -400*x(1), 200 ];
7
   alpha0 = 1;
9  beta = 0.5;
   c1 = 1e-4;
11 tol = 1e-6;

13 %Choose starting value here
   %x0 = [1.2,1.2]';
15 x0 = [-1.2,1]';

17 %Run steepest descent
   X1 = steepdesc(f,df,x0,c1,alpha0,beta,tol);
19
   %Run Newton
21 X2 = newton(f,df,ddf,x0,c1,alpha0,beta,tol);

23 %Run BFGS
   X3 = BFGS(f,df,x0,c1,alpha0,beta,tol);
25
```

```
% Visualization
xrange = linspace( -1.5,1.5,200);
yrange = linspace( -0.5,1.5,200);
[A,B] = meshgrid( xrange, yrange );
vecA = A(:); vecB = B(:);
for i = 1:length(vecA)
    z(i,1) = f( [vecA(i); vecB(i)] );
end
Z = reshape(z, 200, 200);

%Plot steps
scrsz = get(0,'ScreenSize');
figure('Position',[1 scrsz(4)/2-200 scrsz(3)/1.2 scrsz(4)/2+100])

%set linewidth and fontsize
set(0,'defaultLineLineWidth',1.5)
set(0,'defaultAxesFontSize',16)

subplot(1,3,1)
contourf(A,B,Z,20,'linestyle','None');
axis square
hold on
plot( X1(1,:), X1(2,:), '-wo', 'linewidth',2)
xlabel('x_1')
ylabel('x_2')
title(sprintf('steepest descent, x0 = (%g,%g), %i steps',x0(1),x0(2),length(X1(1,:))));

subplot(1,3,2)
contourf(A,B,Z,20,'linestyle','None');
axis square
hold on
plot( X2(1,:), X2(2,:), '-wo', 'linewidth',2)
xlabel('x_1')
ylabel('x_2')
title(sprintf('Newton, x0 = (%g,%g), %i steps',x0(1),x0(2),length(X2(1,:))));

subplot(1,3,3)
contourf(A,B,Z,20,'linestyle','None');
axis square
hold on
plot( X3(1,:), X3(2,:), '-wo', 'linewidth',2)
xlabel('x_1')
ylabel('x_2')
title(sprintf('BFGS, x0 = (%g,%g), %i steps',x0(1),x0(2),length(X3(1,:))));

%Plot errors
scrsz = get(0,'ScreenSize');
figure('Position',[100 scrsz(4)/2-300 scrsz(3)/1.2 scrsz(4)/2+100])

E1 = X1 - ones(2,length(X1(1,:)));
E2 = X2 - ones(2,length(X2(1,:)));
E3 = X3 - ones(2,length(X3(1,:)));

subplot(1,3,1)
semilogy(sqrt(sum(E1.^2,1)));
title('steepest descent');
xlabel('Iterations')
ylabel('Error')

subplot(1,3,2)
semilogy(sqrt(sum(E2.^2,1)));
title('Newton');
xlabel('Iterations')
ylabel('Error')

subplot(1,3,3)
semilogy(sqrt(sum(E3.^2,1)));
title('BFGS');
xlabel('Iterations')
```

```
95  ylabel('Error')
97  end
99  % STEEPEST DESCENT
    % ================================================================
101 function X = steepdesc(f,df,x0,c1,alpha0,beta,tol)
103 X(:,1) = x0;
    k = 1;
105 xk = x0;
    gk = df(xk);
107
    while norm(gk) > tol && k < 1e6  % maxiter just as a safeguard
109
        %search directions
111     pk = -gk;
113     % start backtracking
        alpha = alpha0;
115     while f(xk + alpha*pk) > f(xk) + c1*alpha*gk'*pk
            alpha = alpha*beta;
117     end
119     %Perform step
        xk = xk + alpha*pk;
121     X(:,k+1) = xk;
123     % prepare for next step
        k = k+1;
125     gk = df(xk);
    end
127 end
129 % NEWTON
    % ================================================================
131 function X = newton(f,df,ddf,x0,c1,alpha0,beta,tol)
133 X(:,1) = x0;
    k = 1;
135 xk = x0;
    gk = df(xk);
137
    while norm(gk) > tol && k < 1e6  % maxiter just as a safeguard
139
        Hk = ddf(xk);
141
        %search directions
143     pk = -Hk \ gk;
145     % start backtracking
        alpha = alpha0;
147     while f(xk + alpha*pk) > f(xk) + c1*alpha*gk'*pk
            alpha = alpha*beta;
149     end
151     %Perform step
        xk = xk + alpha*pk;
153     X(:,k+1) = xk;
155     % prepare for next step
        k = k+1;
157     gk = df(xk);
    end
159 end
161 % BFGS
    % ================================================================
163 function X = BFGS(f,df,x0,c1,alpha0,beta,tol)
```

```
X(:,1) = x0;
k = 1;
xk = x0;
gk = df(xk);

Bk = eye(size(x0,1));

while norm(df(xk)) > tol && k < 1e6  % maxiter just as a safeguard

    %search directions
    pk = -Bk \ gk;

    % start backtracking
    alpha = alpha0;
    while f(xk + alpha*pk) > f(xk) + c1*alpha*gk'*pk
        alpha = alpha*beta;
    end

    %Perform step
    sk = alpha*pk;
    xk = xk + sk;
    X(:,k+1) = xk;

    % prepare for next step
    k = k+1;
    yk = gk;
    gk = df(xk);
    yk = gk - yk;

    if k == 1 % adjust bad initial guess
        Bk = B0*(yk'*sk)/(yk'*yk);
    end

    %BFGS update
    zk = Bk * sk;
    Bk = Bk + (yk*yk')/(yk'*sk) - (zk*zk')/(sk'*zk);

end
end
```