Fakultät für Luft- und Raumfahrttechnik
Institut für Mathematik und Rechneranwendung
Prof. Dr. M. Gerdts, Dr. A. Dreves

Universität der Bundeswehr München

Frühjahrstrimester 2019

4. Übung in Optimierung

13) Programmieraufgabe: Implementieren Sie das folgende Gradientenverfahren:

(i) Wähle $x^{[0]} \in \mathbb{R}^n$ und setze $\sigma = 10^{-4}, \beta = 0.5, i := 0$.

(ii) Falls $\|\nabla f(x^{[k]})\| \leq 10^{-4}$: STOP.

(iii) Setze $d^{[i]} := -\nabla f(x^{[i]})$.

(iv) Setze $\alpha_i := 1$. Solange

$$f(x^{[i]} + \alpha_i d^{[i]}) > f(x^{[i]}) + \sigma \alpha_i \nabla f(x^{[i]})^\top d^{[i]}$$

setze $\alpha_i := \beta\, \alpha_i$.

(v) Setze $x^{[i+1]} := x^{[i]} + \alpha_i d^{[i]}$, $i := i + 1$ und gehe zu (ii).

14) Programmieraufgabe: Implementieren Sie das folgende globalisiertes Newton-Verfahren:

(i) Wähle $x^{[0]} \in \mathbb{R}^n$ und setze $\sigma = 10^{-4}, \beta = 0.5, \varrho = 10^{-8}, p = 2.1, i := 0$.

(ii) Falls $\|\nabla f(x^{[k]})\| \leq 10^{-6}$: STOP.

(iii) Bestimme, falls möglich die Suchrichtung $d^{[i]}$ als Lösung von

$$H_f(x^{[i]})\, d = -\nabla f(x^{[i]}]).$$

Wurde keine Lösung gefunden, oder ist die Bedingung

$$\nabla f(x^{[i]})^\top d \leq -\varrho \|d^{[i]}\|^p$$

verletzt, so setze $d^{[i]} := -\nabla f(x^{[i]})$.

(iv) Setze $\alpha_i := 1$. Solange

$$f(x^{[i]} + \alpha_i d^{[i]}) > f(x^{[i]}) + \sigma \alpha_i \nabla f(x^{[i]})^\top d^{[i]}$$

setze $\alpha_i := \beta\, \alpha_i$.

(v) Setze $x^{[i+1]} := x^{[i]} + \alpha_i d^{[i]}$, $i := i + 1$ und gehe zu (ii).

15) Testen Sie beide Verfahren mit folgenden Funktionen:

(i) Rosenbrock-Funktion $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$, Startvektor $(-1.2, 1)$.

(ii) Funktion von Himmelblau $f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$, Startvektor $(4, 2.5)$.

Lösungen

13)
```
function Grad(f,df,x)
% Gradientenverfahren
sigma=1e-4;
beta=0.5;
eps=1e-4;
k=0;
fx=feval(f,x);
dfx=feval(df,x);
while norm(dfx) > eps && k<=10000
   d=-dfx;
   fxd=feval(f,x+d);
   j=0;
   dfxd=dfx*d';
   while fxd > fx + sigma * beta^j * dfxd
       j=j+1;
       xd=x+beta^j*d;
       fxd=feval(f,xd);
   end
   x=xd;
   fx=fxd;
   dfx=feval(df,x);
   k=k+1;
end
fprintf('%5.0f %5.1e %5.1e\n',k,fx,norm(dfx));
disp(x);
end
```

```
14) function Newton(f,df,Hesf,x)
    % Newtonverfahren
    sigma=1e-4;
    beta=0.5;
    rho=1e-8;
    p=2.1;
    eps=1e-6;
    kmax=200;
    k=0;
    fx=feval(f,x);
    dfx=feval(df,x);
    while norm(dfx) > eps && k<=kmax
       Hf=feval(Hesf,x);
       if condest(Hf) > 1e+15
           d=-dfx;
       else
           d=(-Hf\dfx')';
           if dfx*d' > -rho*norm(d)^p
               d=-dfx;
           end
       end
       xd=x+d;
       fxd=feval(f,xd);
       j=0;
       dfxd=dfx*d';
       while fxd > fx + sigma * beta^j * dfxd
           j=j+1;
           xd=x+beta^j*d;
           fxd=feval(f,xd);
       end
       x=xd;
       fx=fxd;
       dfx=feval(df,x);
       k=k+1;
    %   fprintf('%5.0f %10.6f %10.6f %5.1e %5.1e\n',k,x(1),x(2),fx,norm(dfx));
    end
    fprintf('%5.0f %5.1e %5.1e\n',k,fx,norm(dfx));
    disp(x);
    end
```

15)
```
function fx = R(x)
fx=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
end

function dfx=DR(x)
dfx=[-400 * x(1) *(x(2)-x(1)^2)+2*(x(1)-1), 200*(x(2)-x(1)^2)];
end

function dfx=D2R(x)
dfx=[-400 *(x(2)-x(1)^2)+800*x(1)^2+2, -400*x(1);  -400*x(1), 200];
end

function fx = H(x)
fx=(x(1)^2+x(2)-11)^2+(x(1)+x(2)^2-7)^2;
end

function dfx=DH(x)
dfx=[4*x(1)*(x(1)^2+x(2)-11)+2*(x(1)+x(2)^2-7),
     2*(x(1)^2+x(2)-11)+4*x(2)*(x(1)+x(2)^2-7)];
end

function dfx=D2H(x)
dfx=[12*x(1)^2+4*x(2)-42, 4*(x(1)+x(2));
     4*(x(1)+x(2)),       4*x(1)+12*x(2)^2-26];
end
```

```
Grad('R', 'DR', [-1.2;1])          Newton('R', 'DR','D2R', [-1.2;1])
 8058 6.3e-09 1.0e-04                21 3.7e-21 4.5e-10
    0.9999                             1.0000
    0.9998                             1.0000

Grad('H', 'DH', [4;2.5])           Newton('H', 'DH','D2H', [4;2.5])
   18 2.1e-11 5.6e-05                5 1.7e-26 1.5e-12
    3.0000                             3.0000
    2.0000                             2.0000
```

Alternativ kann man in Matlab automatisch die Ableitungen symbolisch berechnen lassen, und diese als Funktionen definieren. Dazu kann man wie folgt vorgehen:

```
function Rosen
x(1:2)=sym('x',[2,1]);
f=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
g=gradient(f,x)';
H=hessian(f,x);
matlabFunction(f,'file','AR','vars',{x});
matlabFunction(g,'file','AGR','vars',{x});
matlabFunction(H,'file','AHR','vars',{x});
end

function Himmelblau
x(1:2)=sym('x',[2,1]);
f=(x(1)^2+x(2)-11)^2+(x(1)+x(2)^2-7)^2;
g=gradient(f,x)';
H=hessian(f,x);
matlabFunction(f,'file','AH','vars',{x});
matlabFunction(g,'file','AGH','vars',{x});
matlabFunction(H,'file','AHH','vars',{x});
end
```