TD 2 Optimisation: méthodes de descente

Exercice 1

1. Représenter graphiquement quelques lignes de niveau de la fonction

$$f(x,y) = 2(x-1)^2 + y^2 + 1$$

puis représenter pour un point sur une des lignes de niveau, le gradient en ce point et un exemple de direction de descente.

2. Soit f une fonction différentiable de $\mathbb{R}^n$ dans $\mathbb{R}$ et $x \in \mathbb{R}^n$. On suppose que $d \in \mathbb{R}^n$ est tel que

$$||\nabla f(x) + d|| \leq ||\nabla f(x)||$$

Montrer que d est une direction de descente de f en x.

3. Soit f une fonction différentiable et convexe de $\mathbb{R}^n$ dans $\mathbb{R}$. Soient x et y dans $\mathbb{R}^n$ tels que $f(y) < f(x)$. Montrer que $y - x$ est une direction de descente de f en x.

Exercice 2

On cherche à minimiser sur $\mathbb{R}^3$ la fonction :

$$J(x,y,z) = x^4 + 2y^4 + z^4 - 2x + y - z$$

1. Donner la solution exacte au problème considéré
2. On veut utiliser la méthode de Newton avec un pas $\alpha = 1$ pour approcher la solution à partir du point $X_0 = (1,1,1)$. Que vaut la direction de descente à la première itération ? Vérifier qu'il s'agit bien d'une direction de descente.
3. Que vaut X_1 ?

Exercice 3

On considère le programme suivant :

```
function y=J(x)
  y=10*(x(2)-x(1)^2)^2+(x(1)-1)^2;
endfunction
//
function y=nablaJ(x)
  y=[-40*x(1)*(x(2)-x(1)^2)+2*(x(1)-1);20*(x(2)-x(1)^2)];
endfunction
//
```

```
function alpha=backtrackA(x,J,Jx,p,gx,betaA,tau,alphainit)
alpha=alphainit;
count=1;
while(J(x+alpha*(p))>Jx+betaA*alpha*(p'*gx))&(count<20)
  count=count+1;
  alpha=tau*alpha;
end
endfunction
//
function p=BFGSdescent(B,gx)
p=linsolve(B,gx); // solve Bp=-gx
endfunction
//
betaA=0.0001; alphainit=1; tau=0.7; epsilon=1E-3;
//
n=2; x=[0;1]; // starting point
//
disp('initial value of x');disp(x);
gx=nablaJ(x);
Jx=J(x);
count=1;
B=eye(n,n);
//
while (norm(gx)>epsilon)&(count<200)
 p=BFGSdescent(B,gx); // p=-gx if steepest descent
 alpha=backtrackA(x,J,Jx,p,gx,betaA,tau,alphainit); // linesearch
 xold=x;//
 gxold=gx;
 x=x+alpha*(p);
  Jx=J(x);
 gx=nablaJ(x);
 count=count+1;
 y=(gx-gxold);s=x-xold;
 B=B+(1/(s'*y))*y*y'-(1/(s'*(B*s)))*B*s*s'*B;
end
disp('final value of x:');disp(x)
disp('iteration number');disp(count)
//
```

1. Expliquer sous forme mathématique le principe de l'algorithme d'optimisation utilisé.
2. Que donne le résultat de la première itération ?
3. On applique l'algorithme précédent pour une fonction de $\mathbb{R}$ dans $\mathbb{R}$ dérivable dont on recherche le minimum. Montrer que la méthode revient à résoudre l'équation $f'(x) = 0$ par la méthode de la sécante.