ECP 2ème année
Semestre 4, Electif 11, Année 2010
Mr DUMAS

NUMERICAL OPTIMIZATION AND APPLICATIONS

examen de rattrapage, 3 hours

Two Scilab programs must be sent (one for each exercise) at the adress `dumas@ann.jussieu.fr` *at the end of the exam with the subjet ECP2010 and with a file name of the type* `ECP2010-firstname-name.sci`.

It will be appreciated that the programs are commented.

EXERCISE 1

Let f a given function C^1 from $\mathbb{R}^n$ to $\mathbb{R}$, such that $\lim\limits_{||x||\to+\infty} f(x) = +\infty$. Denote g the gradient function of f defined from $\mathbb{R}^n$ to $\mathbb{R}^n$. We assume that g is Lipschitz continous on every subset $S_{x_0} = \{x \in \mathbb{R}^n, f(x) \leq f(x_0)\}$. In this case, it is easy to prove that f has a global minima x^* for which $g(x^*) = 0$. The objective is here to construct a sequence $(x_k)_{k\in\mathbb{N}}$ that converge to a local(or global) minima of f. Starting from $x_0 \in \mathbb{R}^n$, the sequence is defined by the relation :

$$x_{k+1} = x_k + t_k d_k$$

where $d_k = -g(x_k) = -g_k$ is the descent direction of the gradient and t_k is constructed by the following algorithm :

- Step 0: start from $t = t_{init} > 0$. Denote $t_g = 0$ and $t_d = 0$.
- Step 1: test t with the criteria a) b) and c):
  - if a) then $t = t_k$.
  - if b) (t too big), denote $t_d = t$ and go to step 2.
  - if c) (t trop small), denote $t_g = t$ and go to step 2.
- Step 2:
  - if $t_d = 0$ then compute a new $t \to \lambda t$
  - if $t_d > 0$ then compute a new $t \to \frac{t_g+t_d}{2}$.
- Step 3: go to step 1 with the new t.

with a) b) and c) the following conditions :

a) t is acceptable if $q(t) \leq q(0) + m_1 t q'(0)$ *and* $q'(t) \geq m_2 q'(0)$.
b) t is too big if $q(t) > q(0) + m_1 t q'(0)$.
c) t is too small if $q(t) \leq q(0) + m_1 t q'(0)$ *and* $q'(t) < m_2 q'(0)$

where $q(t) = f(x_k + t d_k)$ and where m_1 and m_2 are such that $0 < m_1 < m_2 < 1$.

It can be proven that this algorithms always converges in a finite number of iterations and thus that the sequence $(x_k)_{k\in\mathbb{N}}$ is well defined.

1. Implement with Scilab the previous algorithm in order to construct the sequence $(x_k)_{k\in\mathbb{N}}$ associated to a given cost function f with its gradient function g.

2. Apply the previous algorithm to approximate the global minima of the 'banana shape' function :

$$f(x,y) = 100(y - x^2)^2 + (y-1)^2$$

with well chosen values of t_{init}, λ, m_1 and m_2.

EXERCISE 2

The objective is here to modify the following genetic algorithm program, written in Scilab and also available at `http://www.ann.jussieu.fr/~dumas/GA-real2010.sci` :

```
Npop=60; L=2; Ngen=100; pc=0.6; pm=0.2;xmin=-5;xmax=5;
sigma=(xmax-xmin)/100;
function y=J(x)   //
   y=sum(x.^2-cos(2*%pi*x))+length(x);  // rastrigin
endfunction
// initialisation //
A= xmin+(xmax-xmin)*rand(Npop,L+1);
 val=[];oldbestval=0; for
gen=1:Ngen
// evaluation  + elitism  //////////////////////////////////////
for i=1:Npop
  A(i,L+1)=J(A(i,1:L));
end
//  elitism
[bestval,pos]=min(A(:,L+1)); if (bestval>oldbestval)&(gen>1) then
    u=int(Npop*rand()+1); A(u,:)=bestelem;
end
// selection
// 1st step: sort elements
[s,p]=sort(A(:,L+1)); A=A(p,:); val=[val,A(Npop,L+1)];
bestelem=A(Npop,:); oldbestval=A(Npop,L+1);
// 2st step: choose Npop elements with probability pi
//
Asel=[]; p=(1:Npop)/sum(1:Npop);ps=cumsum(p); for i=1:Npop
  u=rand();isel=1;
  while (u>ps(isel))
     isel=isel+1;
  end
Asel=[Asel;A(isel,:)]; end
// barycentric crossover
Acrois=[];
   for k=1:Npop/2
   u1=int(Npop*rand()+1);
   u2=int(Npop*rand()+1);
    if (rand()<pc) then
      alpha=rand();
      uenf1=alpha*Asel(u1,:)+(1-alpha)*Asel(u2,:);
      uenf2=(1-alpha)*Asel(u1,:)+alpha*Asel(u2,:);
    else
      uenf1=Asel(u1,1:L+1);uenf2=Asel(u2,1:L+1);
    end
    Acrois=[Acrois;uenf1;uenf2];
   end
// normal mutation
Amut=Acrois;
  for k=1:Npop
      if (rand()<pm) then
         epsilon=sigma*rand(1,L+1,'normal');Amut(k,:)=Amut(k,:)+epsilon;
      end
  end
A=Amut; end
 for i=1:Npop
  A(i,L+1)=J(A(i,1:L));
end;
 [s,p]=sort(A(:,L+1)); A=A(p,:); disp(A(Npop,1:L));plot2d(val)
```

1. Implement a new selection process, called 2-tournament. It consists in chosing randomly two elements of the population and to keep the best one (in terms of the cost function J) and to repeat this operation `Npop` times.

2. Implement a new mutation process, called non uniform mutation. It consists in modifying each coordinate x_i of a given element of the population by the operation :

$$x_i \to x_i + ((xmax - x_i) * u_i)^b$$

with probability 0.5 or

$$x_i \to x_i - ((x_i - xmin) * u_i)^b$$

with probability 0.5. In this expression, u_i is a random number between 0 and 1 and b is a fixed positive value.

3. Apply this new algorithm to approximate the global minima of the Rastrigin function in 2D :

$$J(x, y) = (x^2 - \cos(2\pi x)) + (y^2 - \cos(2\pi y)) + 2$$

and compare with the original one.