Course on Optimization — University of Mauritius, January 2018
Laurent DUMAS, Versailles University

TD3: direct and trust region methods

Exercice 1– *On the Pattern Search Algorithm*

1. Implement with Scilab/ Matlab the following Pattern Search Algorithm (issued from the PhD thesis of B. Pauwels) :

Algorithme 1.

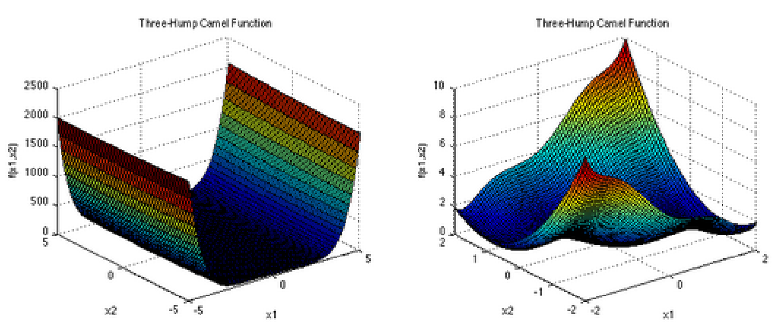
1. Sélectionner θ dans $(0,1)$, γ dans $[1,\infty)$, et $\mathcal{D} \subset \mathbb{R}^n$.
2. Choisir le point de départ x_0 dans $\mathbb{R}^n$ et le pas initial α_0 dans $(0,\infty)$.
3. Pour $k = 0, 1, 2, \ldots$
   - s'il existe une direction d dans $\mathcal{D}$ telle que $f(x_k + \alpha_k d) < f(x_k) - c\alpha_k^2/2$

     définir $\begin{cases} x_{k+1} = x_k + \alpha_k d \\ \alpha_{k+1} = \gamma \alpha_k \end{cases}$; l'itération k est qualifiée de *succès* ;
   - sinon

     définir $\begin{cases} x_{k+1} = x_k \\ \alpha_{k+1} = \theta \alpha_k \end{cases}$; l'itération k est qualifiée d'*échec*.

for a given positive span set $\mathcal{D}$.

2. Apply the previous algorithm to find the minimum of the three-hump camel back function :



$$f(\mathbf{x}) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$$

Description:

Exercice 2– *On the Nelder Mead algorithm*

1. Recall briefly the main principles of the Nelder Mead algorithm. A 2D illustration of the possible steps can be used.

2. Prove that no shrinkage steps are performed when the Nelder Mead algorithm is applied to a strictly convex function. We recall that $f : \mathbb{R}^n \to \mathbb{R}$ is strictly convex if and only if :

$$\forall (x, y) \in \mathbb{R}^n \times \mathbb{R}^n,\ \forall \lambda \in]0, 1[,\ f(\lambda x + (1-\lambda) y) < \lambda f(x) + (1-\lambda) f(y) \text{ if } x \neq y$$

Exercice 3– *On the Lagrange interpolation*

Consider a set $\mathcal{Y} = \{X_1, ..., X_p\}$ of p points in $\mathbb{R}^n$ where p is the cardinality of the polynomial space $\mathbb{R}_d[x_1, ..., x_n]$ ($d \geq 1$). Assume that the set is poised. Denote $\mathcal{B} = \{\Phi_1, ..., \Phi_p\}$ the monomial basis of $\mathbb{R}_d[x_1, ..., x_n]$.

The following algorithm is proposed to define a new polynomial basis :

Initialisation : set $l_j = \Phi_j$ for all $j = 1, ..., p$.

For $i = 1, 2, ..., p$:

- *Point selection* : find $j_0 = argmax_{i \leq j \leq p} |l_i(X_j)|$. If $l_i(X_{j_0}) = 0$ then stop (the set is not poised). Otherwise, swap points X_i and X_{j_0} in $\mathcal{Y}$.
- *Normalisation* : change $l_i(x) \leftarrow \dfrac{l_i(x)}{l_i(X_i)}$
- *Orthogonalization* : for $j = 1, ..., p$, $j \neq i$, change $l_j(x) \leftarrow l_j(x) - l_j(X_i) l_i(x)$

1. If $d \in \{1, 2\}$, what is the value of p for a given n ?
2. Give a condition on a matrix, built with $\mathcal{B}$ and $\mathcal{Y}$, so that the set is poised.
3. Prove that the previous algorithm transforms the basis $\mathcal{B}$ into the Lagrange basis (which definition will be recalled).

Exercice 4– *On a first order DFO trust region method*

The following algorithm in Matlab gives an example of a first order DFO trust region method. The objective is here to use a classical trust region method in dimension n, based on a linear interpolation of the function to minimize f made with a Lagrange interpolation from a set of p points :

```
n=3;  % dimension
p=n+1;
gamma=1.1;
theta=0.9;
eta=0.01;
Nstep=100;
X=rand(n,1); delta=0.1;            % initialization
Xla=[X,X*ones(1,p-1)+delta*(ones(n,p-1)-2*rand(n,p-1))];
Xlatot=Xla;  % total set of possible interpolation points
Xtot=[X];
for i=1:Nstep
```

```
    k=size(Xlatot,2);
    u=zeros(k,1);
    for j=1:k
        u(j)=norm(Xlatot(:,j)-X);
    end
    [a,b]=sort(u);
    Xla=Xlatot(:,b(1:p));      % choice of the nearest p points from X
    w=linlagrange(X,Xla);g=w(2:p);A=zeros(n,n);b=zeros(n,1);
    hplus=linprog(g,A,b,A,b,-delta*ones(n,1),delta*ones(n,1));
    Xplus=X+hplus;
    Xlatot=[Xlatot,Xplus];
    rhok=(f(X)-f(Xplus))/(f(X)-linmodel(g,f(X),hplus)+1E-16);
    if (rhok>eta)
       X=Xplus;delta=gamma*delta;
    else
       delta=theta*delta;
    end
    Xtot=[Xtot,X];
end
disp('best value:');disp(X)
```

In particular, the Matlab instruction `linprog` is used to minimize the function $m(x) = g' * x$ for $-\delta \leq x_i \leq \delta$. $(1 \leq i \leq n)$. The functions `linlagrange`, `linmodel` and `f` need to be defined to complete the code.

1. Give a global description of the script above.
2. Write a possible function `linmodel.m`
3. Write a possible function `linlagrange.m`, either in the particular case where $n = 2$ or in the general case.