Support de logiciel : Optimizer

Editeur : Edios, www.apprendre-a-programmer.com

8 février 2015

Table des matières

1 Présentation du produit

Optimizer un optimiseur sans contraintes implémenté sur MATLAB, dédié pour optimiser des fonctions convexes de dimension $N \geq 1$. Il rassemble toutefois plusieurs méthodes d'optimisation connues (Gradient, Gradient Conjugué, Newton, BFGS,...).

2 Détail

Optimizer donne l'avantage à son utilisateur de switcher d'une méthode d'optimisation à une autre aisément.

```
x_optimal = optimizer(@f, x0,@meth_opti,@regle_rech, tol)
```

les argument d'entrer de l'optimiseur	Significatif
f	La fonction coût
x0	Le point initial
meth_opti	Méthode d'optimisation
regle_rech	Régle de la recherche linéaire
tol	Test d'arrêt

Pour lancer Optimizer, l'utilisateur n'a besoin qu'au moins de la fonction objectif et le point initial, avec une telle configuration, Optimizer va considérer la méthode de gradient comme une méthode d'optimisation par défaut et la règle de wolf comme une méthode de recherche linéaire avec un critère d'arrêt pré-défini. Cependant, l'utilisateur a la possibilité de changer cette configuration de base comme il souhaite.

2.1 Méthodes d'optimisation prise en charge par Optimizer

Optimizer rassemble nombreuses méthodes d'optimisation sans contraintes, afin de donner à l'utilisation plus de choix et surtout la liberté de utiliser la méthode qu'il lui intéresse.

Grandient - Gradient conjugué - Newton - Quasi-Newton rang 1 - Quasi-Newton BFGS - Quasi-Newton DFP - Lenvenberg.

La méthode de recherche linéaire	la fonction matlab
Gradient	gradient1.m
Newton	newton.m
Gradient Conjugué	grad_conjugate.m
Quasi-Newton rang 1	newtion_r1.m
DFP	DFP.m
Quasi-Newton BFGS	BFGS.m
Lenvenberg	lenvenberg.m

Bien entendu, l'utilisation des ces méthodes présentés ci-dessus, demande une connaissance minimale de leurs avantages et inconvénients.

2.2 Méthodes de recherche linéaire

L'optimiseur prend en charge aussi 3 méthodes de recherche linéaire (Armijo, Wolf, Goldstein).

La méthode de recherche linéaire	la fonction matlab
Armijo	armijo.m
Wolf	wolf.m
Goldstein	goldstein.m

2.3 Scripts complémentaires

La fonction matlab	sa fonction
deriv_fonc.m	Dérivée de la fonction objectif
f_test.m	fonction teste
surfelem.m	fonction teste
surfelem2.m	fonction teste
test_area_opti.m	zone d'essai
loire.mat	la carte du département de la Loire

3 Application

3.1 Premier exemple : fonction simple

On commence par essayer la fonction suivante :

$$f(x_1, x_2) = x_1^2 + x_2^2 \tag{1}$$

Pour tester l'optimiseur sur cette fonction, on commence d'abord par écrire la fonction soit à l'aide la commande *inline*, alors :

```
f_test=inline('x(1)^2+x(2)^2','x');
```

Soit, par créer un m-file i.e. $f\_test.m$

```
[sol]=f_test(x)
sol=x(1)^2+x(2)^2;
end
```

Puis, on a besoin aussi de choisir le point initial x_0.

Ces deux arguments sont suffisants pour que Optimizer lancer le processus d'optimisation standard, ce qui donne la liberté à l'utilisateur, s'il veux d'autre spécification (la méthode d'optimisation, règle de recherche linéaire, test d'arrêt).

```
tol=1e-4;
sol=optimizer(@f_test,x0,@BFGS,@wolf,tol);
```

3.2 Second exemple : Application sur le problème de centre d'émission avec deux antennes

Bien entendu, après avoir testé l'optimiseur sur des fonctions simples, il est temps de l'utiliser pour résoudre le problème de centre d'émission, où on commence par le cas de deux antennes, alors on va utiliser la fonction suivante :

$$surfelem(x0, x1, R1, R2) \tag{2}$$

Où $R1 = 3$, R2 = 3, x1 = (-2, 6) et $x0 = (4, -4)$. Dans l'exemple on utilise la méthode DFP avec la règle de recherche linéaire wolf de la manière suivante :

```
x0=[4,-4,-2,6];
sol=optimizer(@surfelem2,x0,@DFP,@wolf);
Le point optimal

x0 =

    0.9487
   -3.2771
   -2.4191
    2.8610

Nombre d appel de la fonction coût
ctp =

   130

Nombre d itération
ite =

    12

Valeur de la fonction coût en x*
fx =

  -34.1544
```

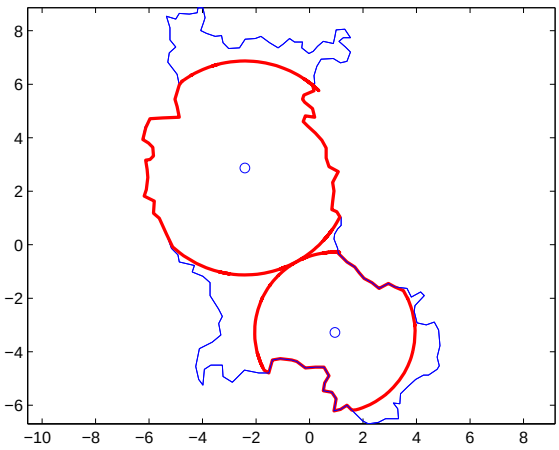

FIGURE 1 – Solution optimale

Remarque 1 *Pour que la méthode de BFGS converge, on propose pour ce cas une correction sur la direction de descente de la manière suivante :*

$$d_k = \begin{cases} -\nabla f(x_k) \ si \ mod(0,2) = 0 \\ -H_k \cdot \nabla f(x_k) \ sinon. \end{cases} \tag{3}$$