PROJET L2 INFO3A de 2021–2022
D. Michelucci

Dans ALGO/SOLVING/TEXTES, le fichier methodes_directes.pdf présente la méthode de Torczon. Vous la programmerez en python. Vous la testerez sur des exemples simples, puis sur des problèmes géométriques tels que ceux dans le fichier ALGO/SOLVING/newton.py Vous pouvez récupérez tout ce que vous voulez dans newton.py et dans tool.py (une petite librairie graphique). Ma solution fait moins de 125 lignes (sans compter les procédures graphiques prises dans newton.py).

Remarque : newton.py s'attend à ce qu'il existe un répertoire IMG, dans lequel il stocke les images.

Remarque : la méthode de Newton résout F(X)=0 en partant d'un point X0 donné. Or la méthode de Torczon est une méthode de minimisation, donc vous minimiserez la norme de F(X), en partant de X0.

Remarque : si votre méthode de Torczon se bloque, vous pouvez pondérer les carrés des (F[i])(X) par des coefficients aléatoires positifs (disons dans [1, 2]), ou tenter de modifier les équations, ou minimiser la somme des valeurs absolues des (F[i])(X).

Deux variantes sont la méthode de Hooke-Jeeves, et celle de Nelder-Mead. Vous pouvez les programmer si vous le souhaitez.

Vous généraliserez l'exemple des 3 cercles tangents entre eux et tangents aux cotés d'un triangle à un nombre entier n quelconque de rangées cercles. Ce problème peut être résolu facilement quand le triangle est équilatéral et de sommets X=(1,0,0), Y=(0,1,0), Z=(0,0,1). Déduisez-en une solution *approximative* pour un triangle générique ABC (non équilatéral), selon le principe : si un cercle dans le triangle équilatéral a pour centre : u X + v Y + w Z (avec 1==u+v+w), alors le cercle correspondant dans le triangle ABC a pour centre u A + v B + w C. Toute méthode partant de cette solution approximative devrait converger. Autres avantages de cet exemple : il est facile d'observer le comportement du solveur ; le système d'équations est bien contraint (autant d'inconnues que d'équations) ; il est facile de générer des exemples complexes, en augmentant le nombre de rangées.

Avant de tester votre solveur sur des exemples compliqués, testez le sur des exemples simples. Testez d'abord ces exemples avec le solveur de Newton.

Vous dessinerez des courbes log-log des temps de calcul par la méthode de Newton et celle de Torczon. Le premier axe, horizontal, est le log du nombre de cercles (ou d'inconnues), le deuxième axe, vertical, est le log du temps de calcul (désactivez l'affichage et la sauvegarde sur fichier). Voir des exemples de courbes log-log dans ALGO/TRI_TABLEAU/qksort.py .

N'utilisez pas la fonction mainloop() de tkinter, et ne perdez pas votre temps à faire de la programmation évènementielle ou IHM : vous ferez ce genre de choses plus tard.

Si jamais nous testons une autre méthode de résolution (gradient conjugué, BFGS, ...), l'algorithme de Torczon en 1D nous donnera une méthode de recherche le long d'une demi-droite (linesearch).

N'attendez pas le dernier jour pour commencer votre projet.