université PARIS DIDEROT PARIS 7

L3 MIASHS - OPTIMISATION - 2018-2019

TP 2 ET 3 : MÉTHODES DE GRADIENT

L'objectif de ces séances est de programmer et de comparer différentes méthodes de minimisation de fonctionnelles quadratiques, appelées méthodes de gradient.

1. Méthode de gradient sans contraintes

Étant donné un entier $N \geqslant 1$, on considère A une matrice symétrique de $\mathbb{R}^{N\times N}$ et b un vecteur de $\mathbb{R}^N$. On cherche à obtenir le vecteur de $\mathbb{R}^N$ minimisant la fonctionnelle

$$J(x) = \frac{1}{2}(Ax, x) - (b, x).$$

On désire programmer et comparer les algorithmes de Gradient à pas fixe (GPF), de Gradient à pas optimal (GPO) et de Gradient conjugué (GC).

Rappel des algorithmes.

Algorithme du Gradient à Pas Fixe :

(GPF) | On choisit x_0 un vecteur de $\mathbb{R}^N$ et $\rho > 0$ un pas fixe.
Itérer pour $n \geqslant 0$: $x_{n+1} = x_n - \rho\nabla J(x_n)$.

Algorithme du Gradient à Pas Optimal :

(GPO) | On se donne x_0 un vecteur de $\mathbb{R}^N$.
Itérer pour $n \geqslant 0$: $x_{n+1} = x_n - \rho_n\nabla J(x_n)$, avec $\rho_n := \frac{(r_n,r_n)}{(Ar_n,r_n)}$ et $r_n = Ax_n - b$.

Algorithme du Gradient Conjugué :

(GC) | On se donne x_0 un vecteur de $\mathbb{R}^N$, et on pose $r_0 = Ax_0 - b$, $d_0 = -r_0$.
Itérer pour $n \geqslant 0$:
$\rho_n := \frac{(r_n,r_n)}{(Ad_n,d_n)}$, $\quad x_{n+1} = x_n + \rho_n d_n$,
$\alpha_n := \frac{||Ax_{n+1}-b||^2}{||r_n||^2}$,
$r_{n+1} = Ax_{n+1} - b$, $\quad d_{n+1} = -r_{n+1} + \alpha_n d_n$.

Exercice 1. (Programmation de la methode GPF.)

1) Dans le fichier `tp2.sce`, définir les données suivantes (de taille $N = 5$)

$$(1) \qquad A = \begin{bmatrix} 3 & -1 & 0 & 0 & 0 \\ -1 & 12 & -1 & 0 & 0 \\ 0 & -1 & 27 & -1 & 0 \\ 0 & 0 & -1 & 48 & -1 \\ 0 & 0 & 0 & -1 & 75 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Exécuter le fichier sous Scilab. Vérifier que `A` est symétrique (ou pourra faire le test en Scilab : `A==A'`). Vérifier vos données en tapant par exemple `A`, `b` dans la fenêtre Scilab, ou à l'aide de la commande `disp`. Vous pouvez également choisir d'évaluer certaines lignes seulement de `tp2.sce`, en les sélectionnant puis Ctrl+E.

2) Programmer une fonction `GPF` sous la forme suivante :

```
function [x,n,resi]=GPF(A,b,x0,eta,Imax,rho)
```

Les paramètres d'entrée sont : la matrice `A`, les vecteurs `b` et `x0`, le paramètre `eta` (η) pour le critère d'arrêt, le nombre maximal d'iterations utilisées `Imax`, et le pas de la méthode `rho` (ρ). Les paramètres de sortie sont : le vecteur solution `x`, le nombre d'itérations `n` effectuées, et la liste des normes des résidus `resi`$= [\|r_1\|, \ldots, \|r_n\|]$, où $r_n = Ax_n - b$.

À chaque itération dans l'algorithme, on utilisera un test d'arrêt du type

$$\big[n > I_{max} \quad \text{ou} \quad \|r_n\| \leqslant \eta\big].$$

En particulier, on continue à itérer tant que $\big[n \leqslant I_{max}$ et $\|r_n\| > \eta\big]$ [1].

3) Test de la méthode.

On pourra initialiser à

$$x_0 := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

Pour commencer on pourra prendre par exemple $I_{max} = 100$, $\eta = 10^{-2}$ et le paramètre $\rho = 10^{-2}$. Que ce passe-t-il si on prend $I_{max} = 1000$? Puis on comparera avec $\rho = \frac{2}{\lambda_1+\lambda_N}$, où λ_1 et λ_N sont resp. les plus petite et plus grande valeurs propres de A. (En SCILAB, utiliser `S=spec(A)` pour avoir les valeurs propres de A, puis `min(S)` et `max(S)` pour avoir les plus petites et plus grandes valeurs propres de A.)

Pour chaque $\eta \in \{10^{-2}, 10^{-4}, 10^{-6}\}$, tester la méthode et indiquer le nombre d'itérations nécessaires pour atteindre le seuil de précision fixé.

Exercice 2. Programmer de même la méthode GPO. Comparer la méthode avec GPF sur les données (1). Même question pour la méthode GC.

Représenter sur un même graphique et pour les différentes méthodes les résidus $(||r_n||)$ en fonction de n. On pourra utiliser une échelle logarithmique pour les ordonnées [2].

1. Dans `Scilab` le ET logique s'écrit `&&`.
2. On rappelle la commande `plot2d( X, Y, style=color("red") );` pour tracer (en rouge) la courbe de `Y` en fonction de `X`. Ici `Y` et `X` sont des vecteurs colonnes. On peut rajouter l'option `logflag="nl"` pour faire un tracé avec une échelle `"n"`ormale en abscisse et `"l"`ogarithmique en ordonnée. Deux telles commandes exécutées à la suite permettent de tracer deux courbes.

Exercice 3. (Préconditionnement.)
On considère toujours les données (1).

1) Noter le nombre d'itérations nécessaires pour avoir $\|r_n\| \leqslant \eta = 10^{-10}$ avec les methodes GPF, GPO. (On se limitera à $I_{max} = 2000$ itérations au plus.)

2) On considère C la matrice diagonale $C = diag(c_1, \ldots, c_N)$ avec $c_i = 1/i$ et $N = 5$, et la nouvelle matrice $A_3 = CAC$. Vérifier que le problème de minimisation de J sur $\mathbb{R}^N$ équivaut au problème de minimisation de la fonctionnelle $J_3(x) = \frac{1}{2}(x, A_3 x) - (b_3, x)$ sur $\mathbb{R}^N$ pour un vecteur b_3 bien choisi.

3) Estimer avec SCILAB le conditionnement des matrices A et A_3.

4) Estimer à nouveau le nombre d'itérations nécessaires pour avoir $\|r_n\| \leqslant \eta = 10^{-10}$ avec les methodes GPF, GPO et appliquées à J_3. (On se limitera à $I_{max} = 2000$ itérations au plus.) Que peut-on en conclure ?

Exercice 4. On se donne maintenant la matrice

$$A_2 = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix} \in \mathbb{R}^{N \times N}. \tag{3}$$

et un vecteur $b_2 \in \mathbb{R}^N$, défini par

$$b_2 = \begin{bmatrix} -20 \\ -20 \\ \vdots \\ -20 \\ -20 \end{bmatrix} \in \mathbb{R}^N \tag{4}$$

avec $N \geqslant 1$ et $h = \frac{1}{N+1}$. On fixe $N = 29$.

1) Construire b_2 et la matrice A_2 (on pourra utiliser une double boucle for).

2) Comparer le nombre d'itérations nécessaires pour les méthode GPO et GC pour A_2 et b_2 à la place de A et b, avec `eta` $= 10^{-6}$ et `Imax` $= 10^4$.

2. MÉTHODES DE GRADIENT PROJETÉ

Dans toute cette partie on considère la matrice A_2 donnée par (3) et le vecteur b_2 donné par (4). On rappelle que $N = 29$.

On cherche à obtenir le vecteur de $\mathbb{R}^N$ minimisant la fonctionnelle

$$J(x) = \frac{1}{2}(A_2 x, x) - (b_2, x)$$

sur un sous-ensemble K de contraintes. On s'intéressera à des contraintes d'obstacle, du type $x_i \geqslant g_i$ où $g = (g_i)$ est donnée par

$$g_i = -1 + \max(0,\ 0.565 - 10(t_i - 0.4)^2), \quad i = 1, \ldots, N$$

avec $t_i = ih$ (pour $i = 1, \ldots, N$). L'ensemble de contrainte K correspondant peut donc s'ecrire sous la forme

$$K := \left\{ x \in \mathbb{R}^N, \quad x \geqslant g \right\}.$$

On désire programmer l'algorithme de gradient projeté.

Rappel : Algorithme du Gradient Projeté

(GP) | On choisit x_0 un vecteur de $\mathbb{R}^N$ et $\rho > 0$ un pas fixe.
Itérer pour $n \geqslant 0$: $x_{n+1} = P_K(x_n - \rho \nabla J(x_n))$.

On rappelle que pour la projection sur $K = \prod_i [a_i, +\infty[$, on a la formule $P_K(x) = (\max(a_i, x_i))_i$.

Exercice 5. Programmation de la méthode de gradient projeté.
Dans cet exercice, on considère la contrainte K.

1) Definir le vecteur (colonne) de contraintes $g = (g_i)$.

2) Programmer une fonction correspondant à la méthode de gradient projeté.
La fonction `GP` aura la forme suivante :

```
function [x,n,resi]=GP(A,b,x0,g,eta,Imax,rho)
```

Les paramètres d'entrée sont : les matrices et vecteurs `A` et `b`, le vecteur obstacle `g`, le paramètre `eta` (η) pour le critère d'arrêt, le nombre maximal d'iterations utilisées `Imax` et le pas de la méthode `rho` (ρ), un paramètre vecteur initial `x0`. Les paramètres de sortie sont : le vecteur solution `x`, le nombre d'itérations `n` effectuées et une liste d'estimateurs d'erreurs `resi`$= [e_1, \ldots, e_n]$ où ici $e_n = \|x_n - x_{n-1}\|$.
À chaque itération dans l'algorithme, on utilisera un test d'arrêt du type

$$\text{si } \left[\, n > I_{max} \text{ ou } e_n \leqslant \eta \,\right], \quad \text{alors arrêter}$$

(en particulier, on continue à itérer tant que $\left[n \leqslant I_{max} \text{ et } e_n > \eta\right]$).
De plus, à chaque itération, on vérifiera que l'algorithme ne diverge pas ; dans le cas où l'algorithme diverge, on affichera un message d'erreur. On pourra effectuer un test du type

$$\text{si } \|x\| > 10^{10} \text{ alors } \texttt{error("L'algorithme a divergé")}$$

3) Pour $I_{max} = 2000$, $\eta = 10^{-4}$, et $x_0 = 0$, indiquer le nombre d'itérations nécessaires pour atteindre le seuil de précision fixé lorsque le pas de l'algorithme vaut $\rho \in \{\rho_{opt}, 10^{-4}, 10^{-3}, 10^{-2}\}$, où ici encore $\rho_{opt} = \frac{2}{\lambda_1 + \lambda_N}$. Que constatez-vous ? A votre avis, quelle condition sur ρ permet de garantir la convergence de l'algorithme ?

4) Reprendre la question précédente, en prenant cette fois-ci comme point de départ la solution du problème *sans contrainte*.

5) Ici on a pris comme estimateur d'erreur la quantité $e_n = \|x_n - x_{n-1}\|$. Vérifier numériquement que pour l'algorithme du gradient projeté on ne peut pas prendre $e_n = \|\nabla J(x_n)\|$ car elle ne converge pas forcément vers 0.