Chp. 7. Algorithmes de gradient conjugué

7.1 Première idée fondamentale

Appliquons l'algorithme de gradient à pas optimal à la minimisation d'une forme quadratique elliptique : $f = (1/2)\, x^T Q\, x$, où : Q est une 2×2 matrice symétrique définie positive :

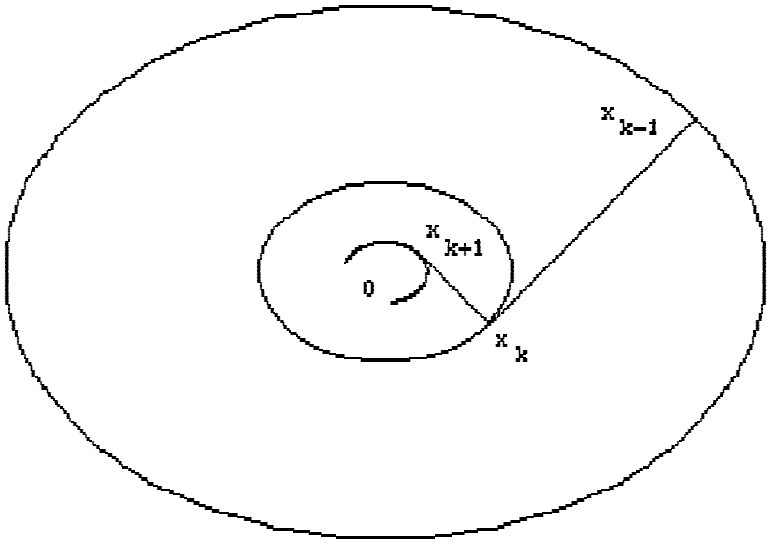

FIG. 7.1-1: `GradOpt` : $f = (1/2)\, x^T Q\, x$. Deux directions de recherche consécutives : $u_k = x_{k+1} - x_k$, et : $u_{k-1} = x_k - x_{k-1}$ sont orthogonales : $u_k^T u_{k-1} = 0$, mais x_k et u_{k-1} sont « conjuguées » : $x_k^T Q\, u_{k-1} = 0$.

A l'étape k , l'algorithme effectue un pas t_{k-1} dans la direction u_{k-1} et actualise le point courant x_{k-1} en : $x_k = x_{k-1} + t_{k-1}\, u_{k-1}$, où t_{k-1} minimise : $\varphi(t) = f(x_{k-1} + t\, u_{k-1})$, donc :

$$\varphi'(t_{k-1}) = \nabla f(x_{k-1} + t_{k-1}\, u_{k-1})^T u_{k-1} = \nabla f(x_k)^T u_{k-1} = x_k^T\, Q\, u_{k-1} = 0$$

Ainsi, la direction $-x_k$ qui aurait permis de trouver à l'étape $k+1$ l'unique minimum $x^\star = 0$ de f sur $\mathbb{R}^2$ est orthogonale à u_{k-1} pour le produit scalaire : $< x, y >= x^T Q\, y$.

Définition 15 *On dit que deux vecteurs u et v de $\mathbb{R}^n$ sont conjugués par rapport à une forme bilinéaire de matrice symétrique définie positive Q si : $u^T Q\, v = 0$*

La première idée fondamentale de l'algorithme du gradient conjugué consiste à choisir chaque direction de recherche de sorte qu'elle soit conjuguée à la direction de recherche précédente par rapport à la Hessienne du critère.

7.2 Seconde idée fondamentale

En dimension deux, il n'existe qu'une seule direction de recherche conjuguée à une direction donnée. En dimension $n > 2$, il en existe une infinité (un sous-espace de dimension $n-1$). Pour trouver, à l'étape k, une direction u_k conjuguée à u_{k-1}, la seconde idée fondamentale de l'algorithme du gradient conjugué consiste à chercher u_k sous forme d'une combinaison linéaire de u_{k-1} et de la direction de Cauchy au point courant :

$$u_k = -\nabla f(x_k) + s_k\, u_{k-1}$$

en choisissant le coefficient réel s_k de telle sorte que :

$$u_{k-1}^T Q\, u_k = s_k\, u_{k-1}^T Q\, u_{k-1} - u_{k-1}^T Q\, \nabla f(x_k) = 0$$

ce qui est toujours possible si u_{k-1} est non nul, puisque : $Q > 0$ implique alors : $u_{k-1}^T Q\, u_{k-1} > 0$.

Connaissant u_{k-1} non nul, on calcule x_k puis s_k et u_k. Si x_k n'est pas un point critique de f, $\nabla f(x_k)$ est, par définition du pas optimal, un vecteur non nul orthogonal à u_{k-1}, et u_k est nécessairement non nul, ce qui permet de calculer x_{k+1}, puis s_{k+1} et u_{k+1}, et ainsi de suite.

Lorsque le critère est une *fonction quadratique elliptique*, on peut démontrer qu'une telle stratégie conduit à définir une suite de directions de recherche *deux à deux* conjuguées, et que les gradients du critère aux points de la suite x_k ainsi obtenue forment un système de vecteurs de $\mathbb{R}^n$ *deux à deux* orthogonaux au sens de la métrique usuelle.

7.3 L'algorithme GradConj

On montre en outre que le réel s_k se calcule par la formule : $s_k = \| \nabla f(x_k) \|^2 \, \| \nabla f(x_{k-1}) \|^{-2}$, d'où la récurrence fondamentale définissant la suite des directions de recherche successives :

$$u_k = -\nabla f(x_k) + \frac{\| \nabla f(x_k) \|^2}{\| \nabla f(x_{k-1}) \|^2}\, u_{k-1}$$

Le calcul du pas optimal : $t_k = -\dfrac{\nabla f(x_k)^T u_k}{u_k^T \nabla^2 f(x_k)\, u_k}$ dans la direction u_k s'obtient alors directement par dérivation de : $\varphi(t) = f(x_k + t\, u_k) = f(x_k) + t\, \nabla f(x_k)^T u_k + \dfrac{t^2}{2}\, u_k^T \nabla^2 f(x_k)\, u_k$, d'où l'algorithme :

```
GradConj(f, x_0, tolerance)
(* Minimise une fonction quadratique elliptique f *)
```

$x \leftarrow x_0$, $\quad u \leftarrow -\nabla f(x_0)$

Tant que : $\| u \| \geq$ tolerance

$$t \leftarrow -\frac{\nabla f(x)^T u}{u^T \nabla^2 f(x)\, u}$$

$s \leftarrow \| \nabla f(x) \|^2$, $\quad x \leftarrow x + t\,u$, $\quad s \leftarrow s^{-1} \star \| \nabla f(x) \|^2$, $\quad u \leftarrow -\nabla f(x) + s\,u$

Retourner x

7.4 Convergence de l'algorithme GradConj

Supposons que f est une fonction quadratique elliptique sur $\mathbb{R}^n$, alors :

Théorème 7.1 *Pour toute initialisation x_0, l'algorithme* `GradConj` *converge en au plus n itérations.*

Preuve : L'orthogonalité deux à deux des gradients aux points calculés par l'algorithme s'obtient par une récurrence technique. On admettra ce résultat, qui est la clé de la démonstration. Si l'algorithme n'a pas trouvé l'unique minimum de f sur $\mathbb{R}^n$ après $n-1$ itérations en effet, les gradients du critère au point initial x_0 et aux $n-1$ premiers points calculés par l'algorithme forment une base orthogonale de $\mathbb{R}^n$. Le gradient du critère au point x_n calculé à la $n^{ème}$ itération doit être orthogonal à tous les vecteurs de cette base : il est donc nul, et x_n minimise f sur $\mathbb{R}^n$.

□

7.5 Un problème test

On cherche à résoudre le système d'équations linéaires : $A\,x = b$, où :

- A est la matrice de Hilbert d'ordre n, de terme général :
$$A_i^j = \frac{1}{i+j-1} \quad (i,j = 1 \ldots n)$$
- $b = (1, 2, \ldots, n)$

en minimisant la fonction quadratique elliptique : $f = \frac{1}{2} \| A\,x - b \|^2$.

L'intérêt de ce test est que la matrice de Hilbert est toujours symétrique et définie positive, ce qui assure l'ellipticité du critère et l'unicité de la solution, mais très mal conditionnée dès les petites valeurs de n (cf. tab. 7.1).

0n peut sur ce test comparer les résultats obtenus, pour différentes valeurs de n, avec les algorithmes `GradOpt` et `GradConj` initialisés, par exemple, avec : $x_0 = (0, 0, \ldots, 0)$. Le pas optimal est calculé dans les deux cas au moyen de la procédure `GoldenSearch`. La valeur du paramètre `tolerance` passé à cette procédure est 10^{-8}.

- Pour $n = 3$, l'algorithme `GradOpt` est déja excessivement lent : il faut en effet plus de 30.000 itérations pour qu'en arrondissant chacune des coordonnées du point finalement calculé à l'entier le

n	3	5	7
χ	274 636	$2.27.10^{11}$	$2.26.10^{17}$
$\frac{\chi-1}{\chi+1}$	0.999993	$\geq 1-10^{-11}$	$\geq 1-10^{-17}$

TAB. 7.1: *Conditionnement χ de la matrice $Q=A^2$, et valeur du taux théorique de convergence de l'algorithme* `GradOpt` *en fonction de n*

plus proche, on obtienne la solution entière exacte $(27,-192,210)$ du système d'équations linéaires à résoudre. L'algorithme `GradConj` calcule les coordonnées de cette solution à 10^{-4} près en seulement trois itérations.

• Pour $n=5$, l'algorithme `GradOpt` ne retourne même plus, après 100.000 itérations, l'ordre de grandeur des coordonnées de l'unique solution entière : $(125,-2880,14.490,-24.640,13.230)$. L'algorithme `GradConj` calcule encore cependant une valeur approchée à 10^{-1} près de chacune des coordonnées de la solution en une dizaine d'itérations.

• Dès que $n \geq 7$, les erreurs d'arrondi suffisent en pratique à rendre les deux algorithmes divergents. Pour n « grand » l'algorithme `GradConj` se compare encore favorablement cependant avec une procédure directe d'inversion de matrice. Pour $n=100$, par exemple, le mauvais conditionnement de la matrice A et les erreurs d'arrondi piègent même la procédure interne optimisée : `Inverse[A].b` de `Mathematica`™, qui retourne une estimation erronée de la solution $A^{-1}b$, pour laquelle la valeur correspondante de la norme de $Ax-b$, supposée être nulle, est en fait supérieure à 10^4 !. Si l'on force le calcul en nombres entiers −tous les coefficients de la matrice A sont rationnels et toutes les coordonnées de b entières− le résultat est exact, mais le temps de calcul approche quatre minutes. L'algorithme `GradConj` trouve, après cent itérations, un point x pour lequel la norme de $Ax-b$ est de l'ordre de 0.8 , avec un temps de calcul de l'ordre d'une vingtaine de secondes.

7.6 L'algorithme de Fletcher-Reeves

L'algorithme du gradient conjugué repose sur une analyse spécifique du problème de la minimisation d'un critère quadratique elliptique. C'est dans ce cadre seulement que l'on peut prouver la convergence en un nombre fini d'itérations.

La formule de récurrence : $u_k = -\nabla f(x_k) + \dfrac{\| \nabla f(x_k) \|^2}{\| \nabla f(x_{k-1}) \|^2}\, u_{k-1}$, qui sert de base à l'algorithme n'utilise cependant pas explicitement le caractère quadratique du critère. Rien n'interdit donc d'appliquer encore « brutalement » cette formule, combinée, à chaque étape, avec une procédure unidimensionnelle de calcul du pas optimal, pour minimiser un critère non quadratique. C'est le principe de l'algorithme de Fletcher-Reeves :

```
FletcherReeves( f, x_0, tolerance)
x ← x_0,    g ← ∇f(x_0)    u ← −g
Tant que : ‖ u ‖ ≥ tolerance
  h ← g,    s ← ‖ h ‖^2
  Calculer le pas optimal t dans la direction u
  x ← x + t u,    g ← ∇f(x)    s ← s^{-1} ‖ g ‖^2,    u ← −g + s u
Retourner x
```

Comme pour l'algorithme `GradOpt`, le calcul du pas s'effectue en pratique au moyen d'une procédure comme `GoldenSearch`, ou `QuadSearch`.

On peut raisonnablement espérer que l'algorithme se comportera exactement comme dans le cas quadratique, et, en particulier, convergera en un petit nombre d'itérations lorsque le critère est une fonction elliptique qui diffère peut, dans un voisinage de l'optimum, de son approximation quadratique.

En outre, l'algorithme ainsi obtenu est *robuste* puisque, par définition du pas optimal :

$$\nabla f(x_k)^T u_k = -\| \nabla f(x_k \|^2 + s_k \nabla f(x_k)^T u_{k-1} = -\| \nabla f(x_k \|^2 < 0$$

et donc les directions de recherche calculées par l'algorithme sont toujours des directions de descente.

7.7 Un test numérique

On a utilisé l'algorithme de Fletcher-Reeves pour minimiser la fonction : $f = x^4 + 4\,y^4 + 4\,x\,y$, qui atteint son minimum sur $I\!R^2$ en deux points : $x_1^\star = (-\sqrt[4]{8}/2, \sqrt[4]{2}/2)$,, et : $x_2^\star = (\sqrt[4]{8}/2, -\sqrt[4]{2}/2)$ et possède un point selle en $(0, 0)$.

L'allure générale des ensembles de niveau correspondants apparaît sur la figure 7.7-2. Les résultats obtenus pour différentes initialisations sont consignés dans la troisième colonne du tableau 7.2, page 42.

Bien que théoriquement robuste, l'algorithme de Fletcher-Reeves apparait numériquement sensible au défaut d'ellipticité du critère. Il semble par exemple hésiter entre les deux minima locaux de f lorsque l'initialisation $(0.5, 0.5)$ est proche du point selle (figure 7.7-2), et l'analyse de la suite des valeurs du critère aux points calculés par l'algorithme montre alors que les directions de recherche successives utilisées ne sont pas toutes en fait des directions de descente.

7.8 Relance de l'algorithme

Le point calculé par l'algorithme après 111 itérations à partir de l'initialisation $(0.5, 0.5)$ est, par exemple, proche du minimum local $x_1^\star$ de f. En relançant l'algorithme à partir de ce point, on imposera à nouveau la direction de Cauchy comme première direction de recherche, permettant d'un coup de se rapprocher suffisamment de $x_1^\star$ pour assurer une convergence de l'algorithme en seulement quelques itérations. En laissant « tourner l'algorithme » au contraire, on obtient encore,

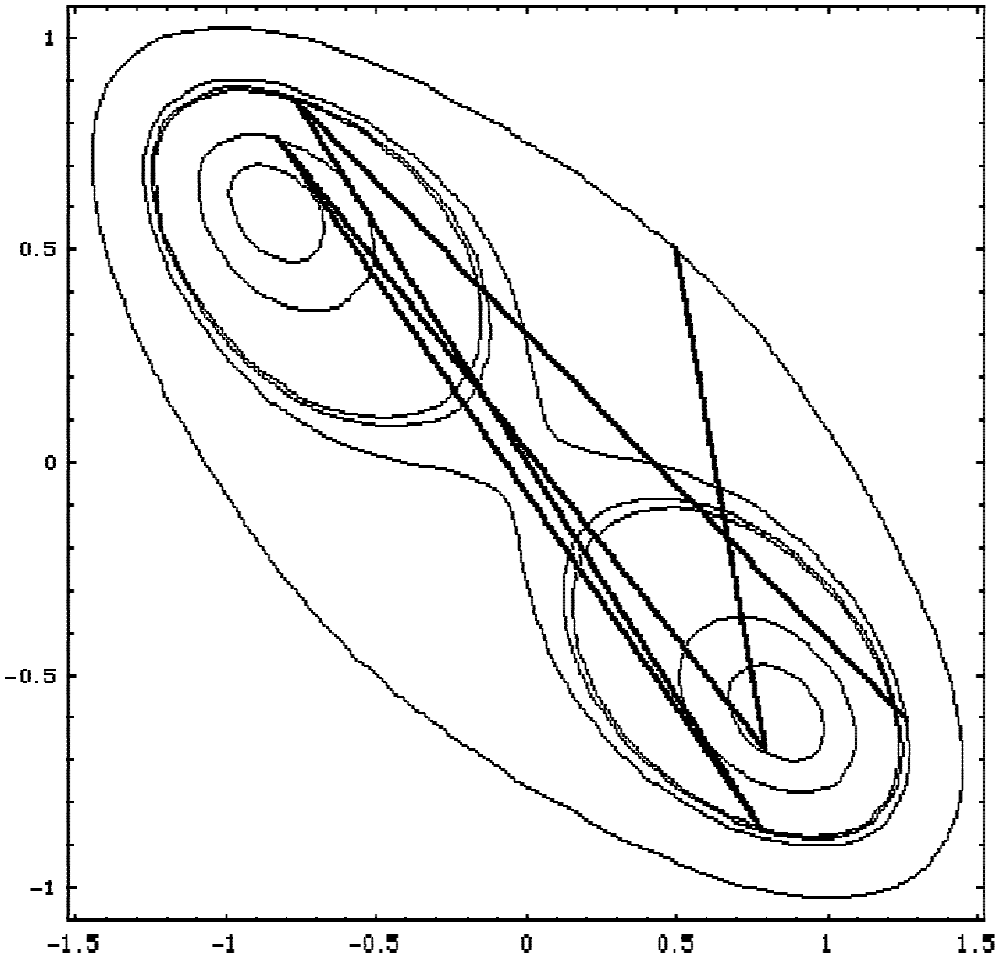

FIG. 7.7-2: *FletcherReeves : minimisation de la fonction* $x^4 + y^4 + 4\,x\,y$, *initialisation :* $(0.5, 0.5)$ *, itérations : 4, 16, 49, 111, et 599*

après 599 itérations, le point : $(1.261, -0.792)$, plus éloigné du plus proche minimum local $x_2^\star$ de f que ne l'était de $x_1^\star$ le point $(-0.744, 0.877)$ obtenu après 111 itérations.

Il est ainsi souhaitable de *relancer* périodiquement l'algorithme du gradient conjugué. Dans le test précédent, le meilleur résultat est obtenu empiriquement en relançant l'algorithme toutes les *deux* itérations !

7.9 Comparaison des algorithmes GradOpt et FletcherReeves

Le tableau 7.2 permet de comparer les performances des algorithmes de gradient à pas optimal et de Fletcher-Reeves, avec ou sans relance périodique, pour différents choix de l'initialisation. Dans les trois cas, la procédure de recherche linéaire utilisée est la procédure `GoldenSearch`. La valeur du paramètre `tolerance` passé à cette procédure est fixée à : 10^{-8}.

	`GradOpt`		`Fletcher-Reeves`			
Initialisation			$(\star)$		$(\star\star)$	
$(0.1, 0.1)$	14	$x_2^\star$	`divergence`		6	$x_2^\star$
$(0.5, 0.5)$	8	$x_2^\star$	`divergence`		6	$x_2^\star$
$(1, 1)$	4	$x_2^\star$	6	$x_2^\star$	5	$x_2^\star$
$(1, -1)$	9	$x_2^\star$	7	$x_2^\star$	5	$x_2^\star$
$(10, 10)$	10	$x_1^\star$	9	$x_1^\star$	7	$x_1^\star$
$(10, -10)$	12	$x_1^\star$	9	$x_1^\star$	8	$x_1^\star$
$(100, 100)$	12	$x_1^\star$	10	$x_2^\star$	9	$x_2^\star$
$(100, -100)$	18	$x_1^\star$	10	$x_2^\star$	10	$x_2^\star$
$(1000, 1000)$	16	$x_1^\star$	13	$x_1^\star$	11	$x_1^\star$
$(1000, -1000)$	16	$x_1^\star$	11	$x_1^\star$	12	$x_1^\star$

TAB. 7.2: *Minimisation de : $f = x^4 + 4\,y^4 + 4\,x\,y$. Comparaison des algorithmes* `GradOpt` *et* `Fletcher Reeves` *($\star$) sans relance, ($\star\star$) avec relance toutes les deux itérations)*

Il donne le nombre d'itérations nécessaires pour obtenir une approximation à 10^{-4} près de chacune des coordonnées de l'un ou l'autre des deux minima de f sur $\mathbb{R}^2$, ainsi que celui, $x_1^\star = (-\sqrt[4]{8}/2, \sqrt[4]{2}/2)$ ou : $x_2^\star = (\sqrt[4]{8}/2, -\sqrt[4]{2}/2)$, vers lequel l'algorithme converge :

On constate que la version avec relance toutes les deux itérations cumule la robustesse de l'algorithme du gradient à pas optimal et le gain en rapidité du à une meilleure prise en compte de la géomètrie des ensembles de niveau apportée par l'algorithme de Fletcher-Reeves.

7.10 La variante Polak-Ribière

Cette variante de l'algorithme de Fletcher-Reeves fut proposée en 1969 par E. Polak et G. Ribière. Elle reprend le principe de l'algorithme de Fletcher-Reeves en remplaçant simplement la formule de récurrence fondamentale définissant les directions de recherche successives par :

$$u_k = -\nabla f(x_k) + \frac{(\nabla f(x_k) - \nabla f(x_{k-1}))^T \nabla f(x_k)}{\| \nabla f(x_{k-1}) \|^2} u_{k-1}$$

Dans le cas d'un critère quadratique, $\nabla f(x_{k-1})$ et $\nabla f(x_k)$ sont orthogonaux, et on retrouve la formule utilisée par l'algorithme de Fletcher-Reeves. Dans le cas général, les deux formules différent, et la version Polak-Ribière est réputée plus performante dans certaines applications.

```
PolakRibiere( f, x_0, tolerance)
x ← x_0,    g ← ∇f(x_0)    u ← −g
Tant que : ‖u‖ ≥ tolerance
  h ← g,    s ← ‖h‖^2
  Calculer le pas optimal t dans la direction u
  x ← x + t u,    g ← ∇f(x)    s ← s^{-1} ⋆ (g − h)^T g,    u ← −g + s u
Retourner x
```