# Maîtrise de Mathématiques Fondamentales et Applications
# Grands Systèmes Linéaires
# feuille de TP 6

15 Octobre 2009

Ce TP a pour but de construire les méthodes itératives (du gradient) pour déterminer la solution $x$ du système linéaire $Ax = b$, où $A$ est une matrice carrée d'ordre $n$, *symétrique définie positive* et $b$ un vecteur de taille $n$. Dans tout ce qui suit, on se trouve dans le cas réel. On désigne par $(\cdot,\cdot)$ le produit scalaire euclidien ( i.e $(x,y) = \sum_{i=1}^{n} x_i y_i$) et $\|\cdot\|$ la norme associée.

---

**Exercice -** 1 *Méthode du gradient à pas fixe (ou de Richardson)*

---

On considère la fonction $f$ définie de $\mathbb{R}^n$ dans $\mathbb{R}$ par $f(x) = \frac{1}{2}(Ax,x) - (b,x)$.
**1-** Montrer que $x_0$ est solution de $Ax = b$ si et seulement si $x_0$ minimise la fonction $f$.

**2-** On considère la méthode itérative :
$\|x_0$ étant un vecteur initial donné, $x_{k+1} = x_k - \alpha\nabla f(x_k)$.
**a-** Donner la matrice d'itération $B$ de cette méthode et conclure que cette méthode converge si et seulement si $0 < \alpha < \frac{2}{\lambda_n}$ où $0 < \lambda_1 \leq \ldots \leq \lambda_n$ sont les valeurs propres de la matrice symétrique définie positive $A$.

**b-** Montrer que le meilleur choix de $\alpha$ est : $\alpha_{opt} = \frac{2}{\lambda_n+\lambda_1}$ et qu'alors $\varrho(B) = \frac{\lambda_n-\lambda_1}{\lambda_n+\lambda_1}$.

**c-** Écrire en pseudo-code l'algorithme de gradient à pas constant, en faisant intervenir le résidu dans les itérations (*on rappelle que le résidu à l'itération k est défini par* $r_k = b - Ax_k$).

**d-** Programmer cet algorithme sous MATLAB à travers une fonction de prototype
`function [x,res]=Gradient(A, b, x0, tol,iterMax, alpha)`, où

- `A` est la matrice du système et `b` le vecteur second membre,
- `x0` est le vecteur initial, `tol` la valeur $\varepsilon$ du test d'arrêt et `iterMax` le nombre maximal d'itérations, `alpha` désigne le paramètre $\alpha$,
- `x` est la solution approchée,
- `res` un vecteur stockant la valeur de la norme du résidu à chaque itération.

**e-** On considère la matrice et le second membre du laplacien `A,b` générés par la commande `[A,b] = laplace2Df(10,10)` (*voir TP 5*).
On fixe `iterMax = 10000, tol = 1E-4`. Faites varier $\alpha$ entre `1E-3` et `3 E-3` par pas de `1E-4`. Représenter le logarithme du nombre d'itérations en fonction du paramètre $\alpha$. Déterminer numériquement la valeur $\alpha$ conduisant à un nombre d'itérations minimal. Comparer avec la valeur donnée par la théorie.

---

**Exercice -** 2 *Méthode du gradient à pas variable*

---

**1-** On suppose construite une suite $(p_0, p_1, \dots p_k, \dots)$ de vecteurs linéairement indépendants. Et on considère la méthode itérative suivante :

$x_0$ vecteur initial donné,
$x_{k+1} = x_k + \alpha_k p_k$.
Où $\alpha_k$ est choisi tel qu'il réalise le minimum de $f(x_k + \alpha p_k)$.

**a-** Montrer que $\alpha_k = \frac{(r_k, p_k)}{(Ap_k, p_k)}$, où $r_k = b - Ax_k$, et conclure que $(r_{k+1}, p_k) = 0, \forall k \geq 0$.

**b-** On pose $E(x_k) = (A(x_k - x), (x_k - x))$ où x est la solution de $Ax = b$.
Montrer que pour la valeur de $\alpha_k$ ci-dessus, on a $E(x_{k+1}) = E(x_k) - \frac{(r_k, p_k)^2}{(Ap_k, p_k)}$.

**c-** En remarquant que $E(x_k) = (A^{-1}r_k, r_k)$, montrer que $E(x_{k+1}) = E(x_k)\left(1 - \frac{(r_k, p_k)^2}{(A^{-1}r_k, r_k)(Ap_k, p_k)}\right)$.

**d-** Déduire de la question précédente que $E(x_{k+1}) \leq E(x_k)\left[1 - \frac{1}{cond(A)}\left(\frac{r_k}{\|r_k\|}, \frac{p_k}{\|p_k\|}\right)^2\right]$.
Où $cond(A) = \frac{\lambda_n}{\lambda_1}$ désigne le conditionnement de la matrice $A$.
*on utilisera le fait que* $(Ay, y) \leq \lambda_n(y, y), \quad (A^{-1}y, y) \leq \frac{1}{\lambda_1}(y, y) \forall y$.
Conclure qu'une condition suffisante de convergence est de choisir les $p_k$ tels que
$\forall k \geq 0 \quad \left(\frac{r_k}{\|r_k\|}, \frac{p_k}{\|p_k\|}\right) \geq \mu > 0$, où $\mu$ est une constante indépendante de $k$.

**e-** Déduire de la question précédente que $p_k = r_k, \forall k \geq 0$ est un choix possible assurant la convergence.

**2- On prend dans cette question** $p_k = r_k \quad \forall k \geq 0$.

**a-** Écrire l'algorithme ainsi obtenu.

**b-** Que devient $E(x_{k+1})$ de la question 1-c ci-dessus ?

**c-** *On admet l'inégalité de Kantorovich suivante :* $\frac{(Ay, y)(A^{-1}y, y)}{(y, y)^2} \leq \frac{(\lambda_n + \lambda_1)^2}{4\lambda_n \lambda_1} \quad \forall y \neq 0.$.
Montrer qu'on a alors $E(x_{k+1}) = E(x_k)\left(\frac{cond(A)-1}{cond(A)+1}\right)^2$.

**d-** Conclure que $\|x_k - x\| \leq \sqrt{\frac{E(x_0)}{\lambda_1}}\left(\frac{cond(A)-1}{cond(A)+1}\right)^k$.

**e-** Programmer cet algorithme sous MATLAB à travers une fonction de prototype
`function [x,res]=GradientV (A, b, x0, tol, iterMax)`.
(*Les arguments sont définis comme à la question 2-d de l'exercice 1.*)

---

**Exercice - 3** *Méthode du gradient conjugué*

---

Une amélioration de la méthode de gradient à pas variable consiste à choisir les directions $p_k$ telles que $x_{k+1}$ réalise le minimum de $f$ sur $x_0 + [p_0, p_1, \dots p_k]$ , en souhaitant que ce problème de minimisation globale soit égale au problème de minimisation locale : $\min_{x = x_0 + \bar{x} + y, y \in [p_k]} f(x)$, dans lequel $\bar{x} \in [p_0, p_1, \dots p_{k-1}]$ est connu et issu d'une précédente minimisation. Les deux premières questions ci-dessous justifient pourquoi on choisit les $p_k$ A-conjugués.

**1-** Monter que $f(x_0 + \bar{x} + \alpha p_k) = f(x_0 + \bar{x}) + \alpha(p_k, A\bar{x}) + \frac{\alpha^2}{2}(p_k, Ap_k) - \alpha(p_k, r_0)$.

**2-** Conclure qu'un moyen de découpler le problème de minimisation globale en une succession de problèmes de minimisation locale est de prendre $(p_k, A\bar{x}) = 0$, ce qui est équivalent à $(p_k, Ap_i) = 0, \ \forall 0 \leq i \leq k-1$. **On dit dans ce cas que les vecteurs** $p_i, i = 1 \dots k$ **sont A-conjugués**.

La méthode du gradient conjugué consiste alors en deux points :

- choisir les directions $p_k$ A-conjuguées c'est-à-dire $(Ap_k, p_j) = 0 \ \forall 0 \leq j \leq k-1$,
- $p_0 = r_0$, et prendre $p_{k+1}$ dans le plan contenant $r_{k+1}$ et $p_k$ c'est-à-dire $p_{k+1} = r_{k+1} + \beta_{k+1} p_k$.

**3-** Montrer que les vecteurs $p_{k+1}$ et $p_k$ sont A-conjugués si et seulement si $\beta_{k+1} = -\frac{(r_{k+1}, Ap_k)}{(p_k, Ap_k)}$.

**4-** En remarquant que $(r_k, p_{k-1}) = 0\ \forall k$, montrer que $(r_k, p_k) = (r_k, r_k)\ \forall k$.
Simplifier alors l'expression de $\alpha_k$.

**5-** En écrivant $r_k = p_k - \beta_k p_{k-1}$, montrer que $(r_{k+1}, r_k) = 0$.(*on utilisera l'expression de* $\beta_k$).

**6-** En écrivant $Ap_{k-1} = \frac{1}{\alpha_{k-1}}(r_{k-1} - r_k)$, montrer que
$(Ap_{k-1}, r_k) = -\frac{1}{\alpha_{k-1}}(r_k, r_k)$ et $(Ap_{k-1}, p_{k-1}) = \frac{1}{\alpha_{k-1}}(r_{k-1}, r_{k-1})$.
Simplifier alors l'expression de $\beta_{k+1}$.

**7-** Montrer que $(r_k, r_j) = 0\ \forall\ 0 \leq j \leq k-1$. En déduire qu'en arithmétique exacte, l'algorithme du gradient conjugué converge en au plus $n$ itérations, où $n$ est la taille du système. (*On remarquera que si* $r_k \neq 0, \forall\ 0 \leq k \leq n-1$, *alors* $[r_0, \ldots, r_{n-1}]$ *est une base de* $\mathbb{R}^n$).

**8-** Écrire l'algorithme du gradient conjugué.

**9-** Programmer cet algorithme sous MATLAB à travers une fonction de prototype
`function [x,res]=GradientC (A, b, x0, tol, iterMax)`.
(*Les arguments sont définis comme à la question 2-d de l'exercice 1.*)

---

**Exercice -** 4 *Comparaison numérique des méthodes*

---

On considère $A$ et $b$ la matrice et le second membre obtenus par discrétisation par différences finies du laplacien sur $]0,1[\times]0,1[$ avec pour second membre $f = 1$ et des conditions aux limites de Dirichlet homogènes. La grille utilisée est définie par $h_x = \frac{1}{n+1}, h_y = \frac{1}{m+1}$.
**1-** Exécuter le script suivant, en vous assurant que les fonctions appelées sont bien accessibles. (*Se référer aux précédents TPs*).

Comparaison des méthodes

```
clear all;
A = laplace2D(10,10);
f = inline('x-x+1','x','y');
g = inline('x-x+0','x','y');
b = Smlaplace2D(10,10,f,g);
x0=b;x0(:,1) = 0;
tol = 1e-9;
iterMax = 100000;
disp(' calcul ');
[x,resJ]=Jacobi(A,b,x0, tol ,iterMax);
[x,resG]=GaussSeidel(A,b,x0, tol ,iterMax);
[x,resS]=sor(A,b,x0, tol ,iterMax, 1.6);
[x,resGradV]=gradientV(A,b,x0, tol ,iterMax);
[x,resGrad]=gradient(A,b,x0, tol ,iterMax,1.);
[x,resGradC]=gradientC(A,b,x0, tol ,iterMax);
disp(' affichage ');
semilogy(1:length(resJ),resJ,'-.r',
         1:length(resG),resG,'--ms',
         1:length(resS),resS,'--s',
         1:length(resGradV),resGradV,':s',
         1:length(resGrad),resGrad,'-b',
         1:length(resGradC),resGradC,'-*r'
        );
legend('Jacobi','Gauss-Seidel','SOR w=1.6','gradient pas variable',
    'gradient pas fixe optimal','gradient conjuguÃ©');
```

**2-** Interpréter les résultats observés.