Analyse numérique avancée
Corrigé du TD 1

M. Kern

18 février 2021

Merci à :
— Matthieu Grondin et Christophe Sive pour l'exercice 1 ;
— Imad Badda et Martin Duguey pour l'exercice 2 ;
— Alexandre Chautard et Sarobidy Randimbiarison pour l'exercice 4 ;
— Yphta Mbangue Lobe et Youssouf Traore pour l'exercice 5 ;
— Lionel Ouedraogo et Bahassane Zalhata pour l'exercice 6.

Exercice 1 : Convergence du gradient conjugué

1 La majoration du cours (inégalité (1.18)) conduit à

$$k \ln\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right) \leq \log\left(\frac{\varepsilon}{2}\right),$$

soit (attention aux signes, les deux quantités dans les logs sont plus petites que 1)

$$k \geq \frac{-1}{\ln\left(\dfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)} \log\left(\frac{2}{\varepsilon}\right)$$

Il est utile d'effectuer un développement limité de l'argument du logarithme dans la limite $\kappa \gg 1$. En écrivant

$$\frac{-1}{\log\left(\dfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)} = \frac{-1}{\log\left(1-\dfrac{2}{\sqrt{\kappa}+1}\right)} \approx \frac{\sqrt{\kappa}}{2},$$

on obtient le résultat plus simple à interpréter :

$$k \gtrapprox \frac{\sqrt{\kappa}}{2} \ln\frac{2}{\varepsilon}.$$

On remarque que la majoration analogue pour la méthode du gradient à pas optimal fait intervenir la quantité κ, alors qu'ici on a $\sqrt{\kappa}$. Si κ est grand, la différence est importante.

2 D'après le résultat rappelé dans le cours (formule en bas de la page 18, voire aussi les notes sur le Laplacien), le conditionnement de la matrice du Laplacien sur une grille avec N points par direction (donc avec un pas $h = 1/(N+1)$, et le nombre d'inconnues est $n = N^2$), vérifie pour h petit

$$\kappa(A) = \frac{\cos^2\left(\dfrac{N\pi}{2(N+1)}\right)}{\sin^2\left(\dfrac{N\pi}{2(N+1)}\right)} \approx \frac{4}{\pi^2 h^2}.$$

D'après la question précédente, le nombre d'itérations nécessaires pour réduire l'erreur d'un facteur ε est donc, au premier ordre

$$k \gtrapprox \frac{h}{4\pi} \log \frac{2}{\varepsilon}.$$

Pour atteindre un niveau de réduction de l'erreur fixé, le nombre d'itérations sera donc proportionnel au pas du maillage.

3 Il est naturel de choisir $\alpha = 1/\lambda_n$, puisque dans ce cas $p(\lambda_n) = 0$ et qu'on a alors l'inégalité

$$|p(\lambda)| \leq |q(\lambda)|, \quad \forall \lambda \in [0, \lambda_n].$$

La majoration du Corollaire 1.1 du cours permet donc d'écrire

$$\|x_k - x^*\|_A \leq \max_{\lambda \in [c,d]} |q(\lambda)| \, \|x_0 - x^*\|_A .$$

En choisissant q de degré $n-1$ qui minimise la quantité correspondante au max, on utilise le résultat rappelé dans l'énoncé, mais avec k remplacé par $k-1$ (une itération pour « se débarrasser » de la plus grande valeur propre), et κ remplacé par $\tilde{\kappa} = d/c$, qui peut être beaucoup plus petit.

En reprenant l'estimation simplifiée, on a donc

$$k \gtrapprox 1 + \frac{\sqrt{\tilde{\kappa}}}{2} \ln \frac{2}{\varepsilon}.$$

Exercice 2 : Coût du gradient conjugué

Remarque préliminaire : les estimations de coût de calculs doivent toujours être interprétées avec précaution. Tout d'abord, on ne retient que le terme dominant. Ensuite, le modèle de coût élémentaire utilisé n'est pas précisé : par exemple, les additions et les multiplications sont-elle comptées ensemble ou séparément ? Cela ne change pas le terme dominant, mais introduit évidemment un facteur 2. Enfin, sur les ordinateurs actuels, le coût principal d'un calcul est plus dû au *mouvement* des données qu'au calcul.

1 Dans l'algorithme du gradient conjugué (Algorithme 1.2), les trois opérations contenant des combinaisons linéaires ont un coût proportionnel à n (la taille du système). Il en est de même pour les deux produits scalaires.

Il reste donc le calcul de Ap (il est bien entendu inutile d'effectuer ce calcul deux fois, on sauve le résultat dans un vecteur intermédiaire). Pour une matrice pleine générale, cette opération a un coût proportionnel à n^2.

Le coût total est évidemment le coût d'une itération multiplié par le nombre d'itérations. Ici, nous utilisons le gradient conjugué comme une méthode directe, et donc nous comptons les n itérations pour obtenir la solution exacte. Dans ce cas, le coût total de la méthode est proportionnel à n^3.

C'est le même ordre de grandeur que pour la méthode Cholesky ou de Gauss, mais une estimation plus précise montre que le coût du gradient conjugué est plus important que pour ces deux méthodes.

2 Si maintenant A est la matrice du Laplacien sur une grille avec N points par coté, le coût de l'opération « produit matrice vecteur » devient proportionnel à N. En effet, la matrice en question a 5 diagonales non-nulles, donc le calcul de chaque élément du produit ne demande au plus que 5 opérations (addition et multiplication). Le coût d'une itération devient donc proportionnel à n (n est le nombre d'inconnues, avec $n = N^2$).

Et dans ce cas, il est naturel d'employer la méthode du gradient conjugué comme une méthode itérative, et donc d'estimer le nombre d'itérations pour réduire l'erreur d'un facteur fixé (la valeur du facteur ne change pas la complexité). D'après l'exercice précédent (question 2), ce nombre d'itération est proportionnel à $N = \sqrt{n}$, ce qui montre que le coût total est proportionnel à $n^{3/2}$. Une comparaison juste avec les méthodes directes considère les méthodes spécialisées pour les matrices creuses, et conduisent à une complexité du même ordre. Pour conclure, notons qu'il existe des méthodes « optimales », c'est-à-dire avec un coût proportionnel à n, pour résoudre ce problème : il s'agit des méthodes multigrilles. On ne peut pas faire mieux, puisqu'il faut évidemment que chaque inconnue intervienne dans le calcul de la solution, d'où un coût minimal proportionnel à n.

Exercice 3 : Récurrences à 3 terme et algorithme de Lanczos

1 On prend les notations du Chapitre 1 du cours.

On part de la relation (1.8) qui définit p_k, que l'on rend explicite,

$$p_k = \frac{1}{\alpha_k}(x_{k+1} - x_k)$$

et de la même relation écrite au rang $k-1$, et l'on reporte ces égalités dans la récurrence de p_k (1.10), on obtient

$$\frac{1}{\alpha_k}(x_{k+1} - x_k) = r_k + \beta_{k-1}\frac{1}{\alpha_{k-1}}(x_k - x_{k-1})$$

ce qui donne

$$x_{k+1} = \alpha_k r_k + \left(1 + \frac{\alpha_k}{\alpha_{k-1}}\beta_{k-1}\right) x_k - \frac{\alpha_k}{\alpha_{k-1}}\beta_{k-1} x_{k-1}.$$

On multiplie cette égalité par A, on change de signe, et on ajoute b de chaque côté, en remarquant que la somme des coefficients de x_k et x_{k-1} vaut 1 :

$$(*) \qquad r_{k+1} = -\alpha_k A r_k + \left(1 + \frac{\alpha_k}{\alpha_{k-1}}\beta_{k-1}\right) r_k - \frac{\alpha_k}{\alpha_{k-1}}\beta_{k-1} r_{k-1}.$$

2 Puisque le vecteurs q_k est, à un facteur multiplicatif près, égal à r_{k-1}, on sait que $(q_1, \ldots, q_k)$ forme une base orthonormée de $K_k(r_0)$ (le vecteur q_k est identique à celui noté v_k au Chapitre 2).

On peut donc poser $x_k = x_0 + Q_k y_k$, avec $y_k \in \mathbf{R}^k$,et $r_k = r_0 - AQ_k y_k$. Comme on sait que x_k est *caractérisé* dans $x_0 + K_k(r_0)$ par la condition $r_k \perp K_k(r_0)$, y_k vérifie

$$Q_k^T \left(r_0 - AQ_k y_k\right), \quad \text{soit } Q_k^T A Q_k y_k = Q_k^T r_0.$$

On note que, puisque A est définie positive et que Q_k est de rang maximum, la matrice $Q_K^T A Q_k$ est également définie positive, et $y)_k$ est ainsi bien défini.

3 On part de la relation (*) dans laquelle on remplace r_j par $\|r_j\|\, q_{j+1}$ pour $j = k-1, k, k+1$, et on utilise la définition de $\beta_k = \frac{\|r_{k+1}\|^2}{\|r_k\|^2}$ pour obtenir, après un court calcul

$$Aq_k = -\frac{\sqrt{\beta_{k-1}}}{\alpha_{k-1}} q_{k+1} + \left(\frac{1}{\alpha_{k-1}} + \frac{\beta_{k-2}}{\alpha_{k-2}}\right) q_k - \frac{\sqrt{\beta_{k-2}}}{\alpha_{k-2}} q_{k-1}.$$

Si l'on désigne par T_k la matrice tridiagonale :

$$T_k = \begin{pmatrix} \frac{1}{\alpha_0} & -\frac{\sqrt{\beta_1}}{\alpha_1} & & & & \\ -\frac{\sqrt{\beta_1}}{\alpha_1} & 1 + \frac{\beta_0}{\alpha_0} & -\frac{\sqrt{\beta_2}}{\alpha_2} & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\frac{\sqrt{\beta_{k-2}}}{\alpha_{k-2}} & \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-2}}{\alpha_{k-2}} & -\frac{\sqrt{\beta_{k-1}}}{\alpha_{k-1}} & \\ & & & -\frac{\sqrt{\beta_{k-1}}}{\alpha_{k-}} & \frac{1}{\alpha_k} + \frac{\beta_{k-1}}{\alpha_{k-1}} \end{pmatrix}$$

on peut réécrire la relation précédente sous la forme

$$AQ_k = T_k Q_k - \frac{\sqrt{\beta_k}}{\alpha_k} q_{k+1} e_{k+1}^T, \quad \text{avec } e_{k+1}^T = (0, \ldots, 1) \in \mathbf{R}^k.$$

On a obtenu ainsi une description matricielle de *l'algorithme de Lanczos* (1952). Il est utilisé pour calculer les valeurs propres de grandes matrices symétriques creuses, et il est équivalent à l'algorithme du gradient conjugué.

Exercice 4 : Convergence de GMRES

1 Pour $n = 4$, la matrice A a la forme suivante :

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Par inspection, la solution est $x = e_4$.

2 On a d'abord $r_0 = b = e_1$, donc $\beta = \|r_0\| = 1$. Au cours de l'algorithme, on notera ξ_1 le premier vecteur de la base canonique de $\mathbf{R}^k$, pour $k = 1, \dots n$.

Première itération On a naturellement $v_1 = r_0 = e_1$, puis $w_2 = Av_1 = e_2$, qui est déjà de norme 1. On en déduit $H_1 \in \mathbf{R}^{2\times 1}$ par

$$(H_1)_{11} = (e_1, e_2) = 0, \quad (H-1)_{2,1} = (e_2, e_2) = 1$$

soit $H_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

On calcule ensuite $y_1 \in \mathbf{R}$ qui minimise $\|H_1 y_1 - \xi_1\|_2^2 = 1 + y_1^2$, on a donc $y_1 = 0$, et $x_1 = V_1 y_1 = 0$. L'algorithme ne fait aucun progrès à la première itération.

Deuxième itération À la deuxième itération, avec $v_2 = w_2$, on calcule $Av_2 = e_3$, qui est directement orthogonal à v_1 et v_2, et donc $w_3 = e_3$, de sorte que $h_{12} = h_{22} = 0$, alors que $h_{32} = 1$. Donc

$$H_2 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Cette fois on cherche $y_2 \in \mathbf{R}^2$, qui minimise

$$\|H_2 y_2 - \xi_1\|_2^2 = 1 + \|y_2\|_2^2,$$

et encore une fois $y_2 = 0$, $x_2 = V_2 y_2 = 0$, et l'algorithme ne fait toujours pas de progrès.

Troisième itération Pour $k = 3$, avec $v_3 = e_3$, on trouve encore $Av_3 = e_4$, donc $w_4 = e_4$, et on voit que

$$H_3 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

puis que $x_3 = 0$.

Les matrices V_k sont donc formées par les colonnes successives de la matrice identité.

Dernière itération La dernière étape est différente, on a $v_4 = e_4$, d'où $Av_4 = e_1$, et on obtient $w_5 = 0$ (on a déjà 4 vecteurs orthogonaux dans $\mathbf{R}^4$) , et $h_{14} = 1$ alors que $h_{24} = h_{34} = h_{44} = h_{54} = 0$. On a donc

$$H_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

On cherche alors $y \in \mathbf{R}^4$ qui minimise

$$\|H_4 y - \xi_1\|_2^2 = (y_4 - 1)^2 + y_1^2 + y_2^2 + y_3^2,$$

ce qui donne $y_4 = (0, 0, 0, 1)^T$, puis (comme prévu) $x_4 = e_4$, la solution exacte du problème.

Pour une dimension n quelconque, on voit de la même façon que les résidus successifs sont donc tous égaux à r_0, sauf à la dernière itération, où l'on trouve la solution. L'algorithme ne fait aucun progrès jusqu'à la dernière itération.

Il s'agit bien entendu du pire cas possible pour GMRES. Il montre qu'aucun résultat général de convergence ne peut être obtenu, sans hypothèse supplémentaire sur la matrice.

Exercice 5 : Factorisation incomplète et matrices de différences finies

1 Pour une matrice A_p avec une seule diagonale non-nulle en position p ($p = 0$ correspond à la diagonale principale, $p < 0$ aux éléments dans le triangle inférieur strict, $p > 0$ au triangle supérieur struct), les indices (i, j) des éléments non-nuls sont tels que $j = i + p$.

Étant donné (i, j), dans la somme qui définit $(A_p A_q)_{ij}$ tous les termes sont nuls sauf, peut-être, celui correspondant à $(A_p)_{i,i+p}(A_q)_{i+p,j}$. Mais celui-ci ne peut être non-nul que si $j = i + p + q$, ce qui correspond à la matrice A_{p+q}, avec

$$(A_p A_q)_{i,i+p+q} = (A_p)_{i,i+p}(A_q)_{i+p,i+p+q}.$$

2 La construction d'une factorisation incomplète demande que les seuls éléments non-nuls dans les facteurs de M soient aux mêmes positions que dans la matrice originale.

Comme A a 5 diagonales non nulles, on peut écrire

$$L_A = (L_A)_{-1} + (L_A)_{-n} \quad \text{et } U_A = (U_A)_1 + (U_A)_n.$$

Par conséquent, L_M aura également 2 diagonales non-nulles, aux positions -1 et $-n$. et U_M aura deux diagonales non-nulles aux positions 1 et n.

3 On développe alors le produit qui définit M :

$$M = (L_M\Delta^{-1} + I)(U_M + \Delta) = L_M + \Delta + U_M + L_M\Delta^{-1}U_M.$$

Rappelons que la deuxième contrainte de construction de M est que $M_{ij} = A_{ij}$ si $A_{ij} \neq 0$. On ne doit donc retenir dans le produit que les termes qui correspondent à des positions existantes dans A, et éliminer les autres.

En développant le produit $L_M\Delta^{-1}U_M = ((L_M\Delta^{-1})_{-1} + (L_M\Delta^{-1})_{-n})\,((U_M)_1 + (U_M)_n)$, on obtient deux sortes de terme (la diagonale Δ^{-1} ne modifie pas les positions)

— des termes en position $1-n$ et $n-1$, qui n'existent pas dans A ;

— des termes en position 0, qui contribueront à la diagonale.

On voit alors que les seuls termes du produit définissant M qui sont en position ± 1 ou $\pm n$ sont ceux qui viennent de L_M et U_M. On en déduit que $L_M = L_A$ et $U_M = U_A$.

Il reste enfin à identifier les termes de la diagonale. On calcule donc

$$\begin{aligned} M_{kk} &= \Delta_{kk} + (L_A\Delta^{-1}U_A)_{kk} \\ &= \Delta_{kk} + (L_A)_{k,k-1}\Delta^{-1}_{k-1,k-1}(U_A)_{k-1,k} + (L_A)_{k,k-n}\Delta^{-1}_{k-n,k-n}(U_A)_{k-n,k}. \end{aligned}$$

Et comme ont doit avoir, pour ces éléments, $M_{kk} = A_{kk}$, on en déduit que les éléments de la diagonale Δ vérifient la récurrence :

$$\Delta_{kk} = A_{kk} - \frac{(L_A)_{k,k-1}(U_A)_{k-1,k}}{\Delta_{k-1,k-1}} - \frac{(L_A)_{k,k-n}(U_A)_{k-n,k}}{\Delta_{k-n,k-n}},$$

et on ne prend pas en compte les termes qui font intervenir des indices négatifs.

Pour interpréter cette récurrence, il est utile de revenir à la grille sous-jacente. Avec une numérotation naturelle des inconnues, si l'indice k dans le vecteur inconnu (donc l'élément diagonal Δ_{kk}) correspond à des indices i et j dans la grille, alors $k-1$ correspond à $i-1$ et j (point à gauche) et $k-n$ correspond à i et $j-1$ (point en dessous), avec les exceptions pour $i = 1$ et $j = 1$ (les bords de la grille), qui correspondent aux cas exceptionnels $k = 1$ et $k = n$.

Exercice 6 : Opérations avec la matrice transposée en format CSR

Dans ce corrigé, l'hypothèse de l'énoncé « les éléments diagonaux de L valent 1 » est abandonnée.

1 On doit résoudre le système triangulaire supérieur

$$L_{ii}x_i + \sum_{j=i+1}^{n}(L^T)_{ij} = L_{ii}x_i + \sum_{j=i}^{n} L_{ji}x_j = b_i, \quad i = 1, \ldots, n,$$

pour lequel l'algorithme naturel (une « descente ») parcourt les éléments de L^T par ligne, donc ceux de L par colonne.

Pour trouver un algorithme par ligne, remarquons qu'une fois qu'on a résolu l'équation pour x_n, l'équation pour x_{n-1} est

$$L_{ii}x_i + \sum_{j=i+1}^{n-1} L_{ji}x_j = b_i - L_{ni}x_n, \quad i = 1, \ldots, n-1.$$

On voit qu'à chaque étape, on doit retirer la contribution de l'élément qui vient d'être calculé aux éléments du second membre. On doit ensuite résoudre un problème similaire de taille $n-1$. Cela conduit à un algorithme qui parcourt les éléments de L par ligne, dans lequel l'ordre des boucles sur i et j est inversé

Algorithme 1 Résolution d'un système triangulaire $L^T x = b$, parcours de L par lignes

```
x = b
for j = n, ..., 1 do
  x_i = x_i / L_ii
  for i = j − 1, ..., 1 do
    x_i = x_i − L_ji x_j
  end for
end for
```

2 L'algorithme précédent admet une traduction immédiate si l'on suppose maintenant que la matrice L est stockée sans un format CSR.

Algorithme 2 Solution du système triangulaire $L^T x = b$, L en stockage CSR

```
x = b
for j = n, ..., 1 do
  k1 = IL(j); k2 = IL(j + 1) − 1
  for k = k1, ..., k2 do
    i = JL(k);   x(i) = x(i) − AL(k)x(j)      // AL(k) contient L_ji
  end for
end for
```