LINMA1170 - Analyse numérique
Méthodes itératives

François Henrotte

November 24, 2020

ÉCOLE
POLYTECHNIQUE
DE LOUVAIN

1 Introduction

Dans ce document, on s'intéresse à la résolution d'un système linéaire d'équations

$$Ax = b \quad , \quad A \in \mathbb{C}^{m\times m} \quad , \quad b \in \mathbb{C}^m \tag{1}$$

où la matrice $A \in \mathbb{C}^{m\times m}$ est carrée et régulière (invertible). Dans cette étude, on fera fréquemment référence à la solutions exacte du système d'équations $x^\star = A^{-1}b$ et au résidu

$$r(x) = b - Ax = A(x^\star - x).$$

Les figures de ce document sont extraites du livre de référence "Numerical Linear Algebra" de Lloyd N. Trefethen et David Bau.

2 Méthodes directes vs méthodes itératives

Lecture 32

Méthodes directes:

- On peut résoudre (1) en utilisant une factorisation de la matrice A: $A = LU$ ou $A = QR$. On a alors une méthode qui demande de l'ordre de $\mathcal{O}(m^3)$ opérations.
- Pour une problème de taille modérée, i.e., $m = 10^4$, la résolution par une méthode directe demande déjà $\mathcal{O}(10^{12}) = \mathcal{O}(\text{Teraflops})$ opérations.

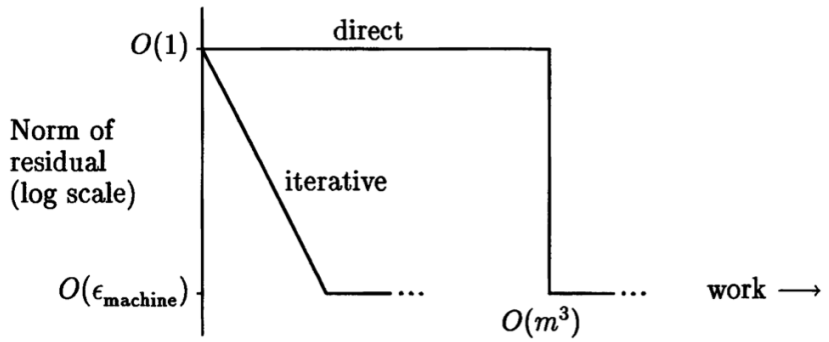


Figure 1: Comparaison entre les méthodes directes et itératives pour la solution de (1).

- Tant que l'on n'a pas été au bout de ces $\mathcal{O}(m^3)$ opérations on n'a aucune approximation de la solution disponible. C'est tout ou rien.

Méthodes itératives:

- Solution intiale arbitraire x_0, puis amélioration progressive de celle-ci, $x_1, x_2, x_3, \ldots, x_m = x^\star$ au cours d'itérations successives.

- La solution exacte $x^\star$ est également obtenue en $\mathcal{O}(m^3)$ en arithmétique exacte avec les méthodes itératives...

- ...mais on peut espérer arriver à une solution suffisamment précise, càd inférieure à une tolérance ϵ que l'on se donne,

$$\frac{||r_n||}{||r_0||} \leq \epsilon \quad \text{ou} \quad ||e_n|| = ||x^\star - x_n|| \leq \epsilon$$

  beaucoup plus rapidement, $<< \mathcal{O}(m^3)$. Ces inégalités sont appelées "critère d'arrêt de la méthode itérative". Notez que le second critère d'arrêt nécessite la connaissance de la solution exacte et est donc rarement utilisable en pratique.

- Si le processus itératif est interrompu à l'itération p, on dispose avec x_p d'une meilleure approximation que x_0 de la solutions exacte $x^\star$. Parfois, très peu d'itérations suffisent (quelques dizaines) et c'est cela qui donne toute leur valeur aux méthodes itératives.

- Si A est symétrique définie positive (SDP) on peut utiliser l'algorithme des gradients conjugués (Hestenes et Stiefel, 1952).

- Si A, n'est pas SDP, on peut utiliser l'algorithme GMRES (General Minimum residual) (Youssef Saad et Martin Schultz, 1986!).

3 Itération d'Arnoldi

Lecture 33

Comment rendre possible les méthodes itératives ?
Remplacer la factorisation (A=QR, LU, SVD) par une transformation

$$A = QHQ^*$$

qui peut facilement devenir (itération d'Arnoldi) :

$$AQ_n = Q_{n+1}\tilde{H}_n. \tag{2}$$

Equation (33.3)

L'idée de l'itération d'Arnoldi est de construire colonne par colonne la matrice unitaire Q qui transforme A en sa forme de Hessenberg.

Une matrice de Hessenberg $H \in \mathbb{C}^{m\times n}$ est une matrice éventuellement rectangulaire ($m \geq n$), qui est triangulaire supérieure avec en plus une seule rangée d'éléments non-nuls juste en dessous de la diagonale. Dans le cas de l'itération de Arnoldi, la matrice de Hessenberg utilisée a la forme

$$H = \begin{pmatrix} * & * & \dots & * & * \\ * & * & \dots & * & * \\ 0 & * & \dots & * & * \\ 0 & 0 & \ddots & * & * \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & * \end{pmatrix} \in \mathbb{C}^{n+1,n}$$

L'itération n d'Arnoldi est représentée par l'équation matricielle

$$\overset{(m\times m)}{\begin{pmatrix} \\ A \\ \\ \end{pmatrix}} \overset{(m\times n)}{(q_1,\dots,q_n)} = \overset{(m\times n+1)}{(q_1,\dots,q_n,q_{n+1})} \overset{(n+1\times n)}{\begin{pmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ h_{21} & h_{22} & \dots & h_{2n} \\ 0 & h_{32} & \dots & h_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & h_{n+1,n} \end{pmatrix}} \tag{3}$$

qui n'est qu'une réécriture de (2), et se déroule donc comme suit.

On part avec un vecteur b arbitraire (pour l'algorithme de GMRES, ce vecteur sera le second membre du système $Ax = b$ à résoudre) et on définit la première colonne de Q comme étant $q_1 = b/||b||$. Pour $n = 1$, (3) s'écrit

$$Aq_1 = (q_1, q_2) \begin{pmatrix} h_{11} \\ h_{21} \end{pmatrix} = h_{11}q_1 + h_{21}q_2$$

d'où il faut extraire h_{11}, h_{21} et q_2, càd une colonne de $\tilde{H}$ et une colonne de Q. Pour cela il faut procéder par ordre. On pose d'abord $v = Aq_1$ puis on identifie $h_{11} = q_1^* v (= q_1^* A q_1)$ en multipliant l'équation par q_1 et en utilisant le fait que, même si l'on ne connait pas encore q_2, on sait déjà que $q_1^* q_2 = 0$ car on veut construire une matrice Q unitaire. On soustrait ensuite $v = v - h_{11} q_1$ pour obtenir le vecteur $h_{21} q_2$ dont la norme vaut $h_{21} = ||v||$ et le vecteur normé est $q_2 = v/h_{21}$. On remarque que q_2 est combinaison linéaire de q_1 et de Aq_1, càd de b et Ab, ce que l'on écrit

$$q_2 \in \langle b, Ab \rangle.$$

L'algorithme se poursuit alors comme suit

Algorithm 33.1. Arnoldi Iteration

$b =$ arbitrary, $q_1 = b/\|b\|$
for $n = 1, 2, 3, \ldots$
  $v = Aq_n$
  for $j = 1$ **to** n
    $h_{jn} = q_j^* v$
    $v = v - h_{jn} q_j$
  $h_{n+1,n} = \|v\|$ [see Exercise 33.2 concerning $h_{n+1,n} = 0$]
  $q_{n+1} = v/h_{n+1,n}$

page 252

4 Sous-espaces de Krylov K_n

page 253

Si on en écrit la boucle sur j de l'itération d'Arnoldi algébriquement, on a

$$q_{n+1} = \frac{1}{h_{n+1,n}} \left(Aq_n - \sum_{k=1}^{n} h_{kn} q_k \right) \tag{4}$$

qui montre que q_{n+1} est combinaison linéaire des $q_k, k = 1, \ldots, n$, et de Aq_n. Autrement dit, si on se souvient que $q_1 = b$,

$$\begin{aligned}
&q_1 \in \langle b \rangle \\
&q_2 \in \langle b, Ab \rangle \\
&q_3 \in \langle b, Ab, A^2 b \rangle \\
&\ldots \\
&q_n \in \langle b, Ab, A^2 b, \ldots, A^{n-1} \rangle
\end{aligned}$$

et donc

$$\begin{aligned}
\langle q_1, q_2 \rangle &= \langle b, Ab \rangle = K_2 \\
\langle q_1, q_2, q_3 \rangle &= \langle b, Ab, A^2 b \rangle = K_3 \\
&\dots \\
\langle q_1, \dots, q_n \rangle &= \langle b, Ab, A^2 b, \dots, A^{n-1} b \rangle = K_n
\end{aligned} \tag{5}$$

où les ensembles $K_n \subseteq \mathbb{C}^m, n = 1, \dots$ sont les sous-espaces de Krylov du système $Ax = b$. Bien voir que, pour K_2 par exemple, Ab n'est pas orthogonal à b, mais q_1 est orthogonal à q_2.

Quelques remarques importantes sur ces notations.

- La notation $\langle v_1, \dots, v_n \rangle$ représente l'espace vectoriel des combinaisons linéaires des vecteurs $\{v_1, \dots, v_n\}$.
- On a vu que la produit matrice-vecteur Ac pouvait être vu également comme l'ensemble des combinaisons linéaires des colonnes $\{a_1, \dots, a_n\}$ de la matrice $A \in \mathbb{C}^{m \times n}$. Donc

  $$v \in \langle a_1, \dots, a_n \rangle \Leftrightarrow v = (a_1, \dots, a_n)c = Ac = \sum_{k=1}^{n} c_k a_k$$

  où $c \in \mathbb{C}^n$ est le vecteur des coefficients de la combinaison linéaire qui représente le vecteur v dans la base $\{a_1, \dots, a_n\}$.
- Distinguez donc bien
  - l'espace vectoriel $\langle a_1, \dots, a_n \rangle$,
  - la base $\{a_1, \dots, a_n\}$
  - et la matrice $(a_1, \dots, a_n)$,

  même si toutes ces notions sont bien sûr liées.

On a maintenant compris ce que fait l'itération d'Arnoldi. Elle construit de façon itérative une base orthonormale pour les sous-espaces de Krylov successifs K_n. Les sous-espaces de Krylov sont le concept (géométrique) central permettant de comprendre les méthodes itératives dites "de Krylov" qui sont reprises dans le tableau suivant :

	$Ax = b$	$Ax = \lambda x$
$A = A^*$	CG	Lanczos
$A \neq A^*$	GMRES CGN BCG et al.	Arnoldi

5 Polynomes

page 259 et 268

Les polynômes jouent un rôle fondamental dans l'étude théorique des méthodes de Krylov. En effet,

$$\begin{aligned} x \in K_n \Rightarrow \quad & x = K_n c \\ \Rightarrow \quad & x = c_0 b + c_1 Ab + c_2 A^2 b + \cdots + c_{n-1} A^{n-1} b \\ \Rightarrow \quad & x = q(A)b \end{aligned}$$

où

$$q(z) = c_0 + c_1 z + c_2 z^2 + \cdots + c_{n-1} z^{n-1}$$

est un polynôme d'ordre $n-1$. Notez que dans la première implication ci-dessus, K_n représente d'abord un espace vectoriel puis une matrice (même notation pour deux concepts proches, mais différents).

Le résidu s'écrit alors, pour $x_n \in K_n$,

$$r_n = b - Ax_n = b - A\, q(A)\, b = p_n(A)b$$

où

$$p_n \in P_n = \left\{\text{polynomes } p \text{ de degré } \leq n, p(0) = 1\right\} \tag{6}$$

puisque le coefficient devant le vecteur de base b doit obligatoirement être 1.

6 L'algorithme GMRES

Lecture 35

Le principe de l'algorithme GMRES (General Minimum Residual) tient en une seule ligne :

À l'itération n de l'algorithme de GMRES, on approxime la solutions exacte $x^\star$ par le vecteur $x_n \in K_n$ qui minimise

$$||r_n(x_n)|| = ||b - Ax_n||.$$

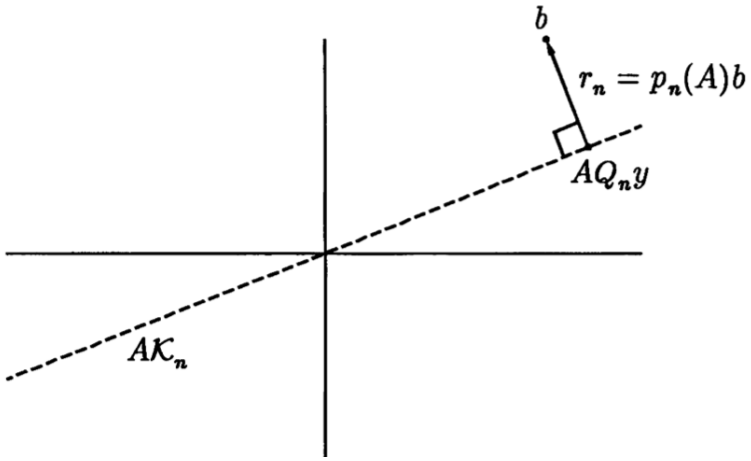
Dans la démonstration de l'algorithme, on va montrer que toutes les expressions suivantes de la norme du résidu à minimiser sont équivalentes:

$$||r_n(x_n)|| = ||b - Ax_n|| = ||AK_nc - b|| = ||p_n(A)|| = ||AQ_ny - b||.$$

Pour cela, on a utilisé le fait que le vecteur $x_n \in K_n$ peut s'écrire $x_n = K_nc$, avec c est le vecteur des coefficients de x_n dans la base $\{b, Ab, \ldots, A^{n-1}b\}$ et où K_n représente ici la matrice $(b, Ab, \ldots, A^{n-1}b)$ associée au sous-espace de Krylov K_n. Ensuite, en utilisant l'identité des espaces (5)

$$\langle q_1, \ldots, q_n\rangle = \langle b, Ab, A^2b, \ldots, A^{n-1}b\rangle = K_n$$

tels qu'ils sont construits par l'itération d'Arnoldi, on voit que le même vecteur $x_n \in K_n$ peut également s'écrire $x_n = Q_ny$ avec y le vecteur des coefficients de x_n dans la base $\{q_1, \ldots, q_n\}$ et où Q_n représente donc ici la matrice $(q_1, \ldots, q_n)$. Graphiquement l'itération d'Arnoldi ressemble donc à ceci:



Sur base du principe général de l'algorithme GMRES, on peut dire les choses suivantes.
D'abord, la convergence est monotone :

$$||r_{n+1}|| \leq ||r_n|| \quad \Leftarrow \quad K_{n+1} \supseteq K_n.$$

Le résidu est en effet plus petit si on le minimise dans un espace vectoriel plus grand. De même, pour dire la même chose autrement :

$$||p_{n+1}(A)b|| \leq ||p_n(A)b|| \quad \Leftarrow \quad P_{n+1} \supseteq P_n.$$

L'erreur est plus petite lorsque l'on interpole dans un espace polynomial plus grand.

Enfin, $||r_m|| = 0$ car $K_m = \mathbb{C}^m$ si $A \in \mathbb{C}^{m \times m}$ est invertible. La convergence est donc assurée en m itérations (où m est la dimension de la matrice du système et si on travaille en arithmétique exacte), mais on veut une convergence en $p << m$ itérations.

7 L'algorithme des gradients conjugués (CG)

Lecture 38

- C'est la plus ancienne/connue/utilisée des méthodes de Krylov.
- Elle n'est valable que pour les matrices SDP.
- Pilier du développement de la simulation numérique (e.g., par éléments finis) pour les problèmes en variable réelle, monophysiques, sans terme convectif, etc...
- On va supposer $A \in \mathbb{R}^{m \times m} \quad , \quad A = A^T$
- Une matrice de Hessenberg qui est en plus symétrique est une matrice tridiagonale. CG utilise cette propriété pour accélérer l'algorithme.
- A est définie positive implique que

  $$x^T A x > 0 \quad \forall x \in \mathbb{R}^m, ||x|| \neq 0$$

  et que toutes ses valeurs propres sont positives.
- Dans ces conditions,

  $$||x||_A = \sqrt{x^T A x}$$

  est une norme vectorielle pour $\mathbb{R}^m$ (ce n'est pas une des normes vectorielles que l'on a vues jusqu'ici).
- On définit l'erreur à l'itération n

  $$e_n = x_n - x^\star.$$

Le principe de l'algorithme des gradients conjugués (CG, conjugate gradients) tient alors également en une seule ligne (comparez avec GMRES) :

> À l'itération n de l'algorithme CG, on approxime la solutions exacte $x^\star$ par le vecteur $x_n \in K_n$ qui minimise
>
> $$||e_n||_A = \sqrt{(x_n - x^\star)^T A (x_n - x^\star)}.$$

On peut d'abord noter que la fonctionnelle à minimiser s'écrit

$$||e_n||_A = x_n^T A x_n - 2x_n^T b + (x^\star)^T b.$$

Le dernier terme est inconnu, mais puisqu'il est constant (indépendant de x_n), il n'intervient pas dans la minimisation. Les deux premiers termes forment une fonctionnelle quadratique de x_n, ce qui lui confère de nombreuses propriétés utiles

- convexité
- minimum unique
- convergence rapide avec les méthodes de gradient
- etc...

L'algorithme CG s'écrit :

Algorithm 38.1. Conjugate Gradient (CG) Iteration

$x_0 = 0,\ r_0 = b,\ p_0 = r_0$
for $n = 1, 2, 3, \ldots$

$\alpha_n = (r_{n-1}^T r_{n-1})/(p_{n-1}^T A p_{n-1})$	step length
$x_n = x_{n-1} + \alpha_n p_{n-1}$	approximate solution
$r_n = r_{n-1} - \alpha_n A p_{n-1}$	residual
$\beta_n = (r_n^T r_n)/(r_{n-1}^T r_{n-1})$	improvement this step
$p_n = r_n + \beta_n p_{n-1}$	search direction

Analyse de l'algorithme :

- Il y a un seul produit matrice-vecteur, Ap_{n-1}, par itération.
- Il représente $\mathcal{O}(m^2)$ opérations si A est dense, mais seulement $\mathcal{O}(m)$ opérations si A est creuse (largeur de bande $<< m$).
- Les théorèmes sont démontrés en arithmétque exacte mais les performances de CG restent excellentes en arithmétique en virgiule flottante.
- Les résidus sont orthogonaux (cf Théorème 38.1)

$$r_n^T r_j = 0 \quad , \quad j < n$$

- Les directions de recherche sont conjuguées

$$p_n^T A p_j = 0 \quad , \quad j < n$$

- Directions de recherches:

$$p_0 = r_0 \quad , \quad p_n = r_n + \beta_n p_{n-1} \quad \Rightarrow \quad \langle p_0, \ldots, p_{n-1} \rangle = \langle r_0, \ldots, r_{n-1} \rangle$$

- Résidus:

$$r_0 = b \quad , \quad r_n = r_{n-1} - \alpha_n A p_{n-1} \quad \Rightarrow \quad \langle r_0, \ldots, r_{n-1} \rangle = \langle b, Ab, \ldots, A^{n-1}b \rangle = K_n$$

  qui montre que CG est bien une méthode de Krylov.

- Solutions à chaque itération

$$x_0 = 0 \quad , \quad x_n = x_{n-1} + \alpha_n p_{n-1} \quad \Rightarrow \quad \langle x_1, \ldots, x_n \rangle = \langle p_0, \ldots, p_{n-1} \rangle$$

- Convergence monotone:

$$||e_{n+1}||_A \leq ||e_n||_A$$

  pour la même raison qu'expliqué plus haut pour GMRES.

On a donc pour chaque itération de l'algorithme CG les identités suivantes entre les espaces vectoriels utilisés :

$$\langle x_1, \ldots, x_n \rangle = \langle p_0, \ldots, p_{n-1} \rangle = \langle r_0, \ldots, r_{n-1} \rangle = \langle b, Ab, \ldots, A^{n-1}b \rangle = K_n. \tag{7}$$

qui confirment le principe de l'algorithme CG donné en une seule phrase plus haut.

8 Préconditionnement

8.1 Idée de base du préconditionnement

Les méthodes itératives convergent rapidement si A est proche de l'identité (nombre de conditionnement $\kappa(A) \approx 1$) mais convergent très lentement (i.e., sont inutilisables) si le problème est mal conditionné ($\kappa(A) >>$). En général les problèmes ne sont pas bien conditionnés et il faut nécessairement faire quelque chose pour pouvoir utiliser les méthodes itératives de façon efficace.

L'idée de base du préconditionnement est de trouver une matrice M, appelée préconditionneur, telle que

- $||\mathbb{I} - M^{-1}A|| << 1$
- M est proche de A
- $M^{-1}A$ est proche de $\mathbb{I}$

- Les valeurs propres de $M^{-1}A$ sont rassemblées dans une région aussi petite que possible autour de 1 dans le plan complexe. Donc préconditionner revient à rassembler les valeurs propres autour de 1.
- $\kappa(M^{-1}A) << \kappa(A)$. Donc préconditionner revient à améliorer le nombre de conditionnement de la matrice du système.
- Préconditionnement à gauche: on résout $M^{-1}Ax = M^{-1}b$ au lieu de $Ax = b$. Les deux systèmes ont clairement la même solution.
- Préconditionnement à droite: on résout $AM^{-1}\tilde{x} = b$ avec $x = M^{-1}\tilde{x}$.
- Préconditionnement pour systèmes SDP: Les préconditionnement à gauche et à droite peuvent faire perdre la symétrie du système, ce qui est dommage. Si A est SDP, on préfèrera donc résoudre $L^TAL\tilde{x} = L^Tb$ avec $\tilde{x} = L^{-1}x$ où L est invertible et $M^{-1} = LL^T$. La matrice $\tilde{A} = L^TAL$ reste alors SDP mais est mieux conditionnée que A.

Les idées de base listées ci-dessus sont plus ou moins valables pour tous les types de préconditionneurs. Le préconditionnement est une opération assez empirique, qui demande de l'expérience, une bonne connaissance du problème à résoudre, et de disposer d'un grand arsenal de préconditionneurs différents. L'implémentation du préconditionnmenent (à gauche, par exemple), est relativement trivial. Il suffit de remplacer A par $M^{-1}A$ et b par $M^{-1}b$ dans l'algorithme non-préconditionné.

Voir aussi le document `Notes préconditionnement.pdf` sur Moodle.

8.2 Préconditionnement de CG

Comme mentionné plus haut, un préconditionnement spécifique est conseillé pour les matrices SDP. Voyons en détail comment appliquer ce principe au cas de l'algorithme CG.

On choisit une matrice invertible L et on va utiliser comme précontionneur la matrice $M^{-1} = LL^T$. On a alors successivement

$$\begin{aligned} Ax &= b \\ M^{-1}Ax &= M^{-1}b \\ L^TAx &= L^Tb && \text{car } L \text{ est invertible} \\ (L^TAL)(L^{-1}x) &= L^Tb \\ \hat{A}\hat{x} &= \hat{b} \end{aligned}$$

Soit donc les définitions: $\hat{A} = (L^TAL)$, $\hat{x} = L^{-1}x$, $\hat{b} = L^Tb$. On met maintenant des chapeaux à toutes les variables de l'algorithme CG non-préconditionné pour avoir, avec les définitions ci-dessus,

l'algorithme préconditionné. On note ensuite qu'on a les relations suivantes

$$\begin{aligned}\hat{x}_n &= L^{-1}x_n\\ \hat{p}_n &= L^{-1}p_n\\ \hat{r}_n &= L^T r_n\end{aligned}$$

entre les vecteurs préconditionnés et les vecteurs non-préconditionnés. On le vérifie par simple substitution. Par exemple

$$\begin{aligned}\hat{r}_n &= \hat{b} - \hat{A}\hat{x}_n\\ &= L^T b - L^T A L L^{-1} x_n\\ &= L^T(b - Ax_n)\\ &= L^T r_n\end{aligned}$$

On définit

$$\tilde{r}_n = L\hat{r}_n = M^{-1}r_n.$$

Si on substitue maintenant dans l'algorithme toutes les grandeurs avec des chapeaux par leur expression, on observe que toutes les matrices $L^{\times}$ disparaissent comme par magie et que seuls subsistent à certains endroits stratégiques des $\tilde{r}_n$ qui viennent remplacer desr_n.

$$\begin{aligned}&\hat{r}_0 = \hat{b}_0 \quad \Rightarrow \quad L^T r_0 = L^T b_0 \quad \Rightarrow \quad r_0 = b_0\\ &\hat{p}_0 = \hat{r}_0 \quad \Rightarrow \quad L^{-1}p_0 = L^T b_0 \quad \Rightarrow \quad p_0 = \tilde{r}_0\\ &\hat{\alpha}_n = \frac{\hat{r}_{n-1}^T\hat{r}_{n-1}}{\hat{p}_{n-1}^T\hat{A}\hat{p}_{n-1}} = \frac{r_{n-1}^T L L^T r_{n-1}}{p_{n-1}^T L^{-T}L^T A L L^{-1} p_{n-1}} = \frac{r_{n-1}^T\tilde{r}_{n-1}}{p_{n-1}^T A p_{n-1}}\\ &\hat{x}_n = \hat{x}_{n-1} + \hat{\alpha}_n\hat{p}_{n-1} \quad \Rightarrow \quad L^{-1}x_n = L^{-1}x_{n-1} + \hat{\alpha}_n L^{-1}p_{n-1} \quad \Rightarrow \quad x_n = x_{n-1} + \hat{\alpha}_n p_{n-1}\\ &\hat{r}_n = \hat{r}_{n-1} - \hat{\alpha}_n\hat{A}\hat{p}_{n-1} \quad \Rightarrow \quad L^T r_n = L^T r_{n-1} - \hat{\alpha}_n L^T A L L^{-1} p_{n-1} \quad \Rightarrow \quad r_n = r_{n-1} - \hat{\alpha}_n A p_{n-1}\\ &\hat{\beta}_n = \frac{\hat{r}_n^T\hat{r}_n}{\hat{r}_{n-1}^T\hat{r}_{n-1}} = \frac{r_n^T L L^T r_n}{r_{n-1}^T L L^T r_{n-1}} = \frac{r_n^T\tilde{r}_n}{r_{n-1}^T\tilde{r}_{n-1}}\\ &\hat{p}_n = \hat{r}_n + \hat{\beta}_n\hat{p}_{n-1} \quad \Rightarrow \quad L^{-1}p_n = L^T r_n + \hat{\beta}_n L^{-1}p_{n-1} \quad \Rightarrow \quad p_n = \tilde{r}_n + \hat{\beta}_n p_{n-1}\end{aligned}$$

L'algorithme CG préconditionné s'écrit donc finalement

$x_0 = 0, r_0 = b$
Résoudre $M\tilde{r}_0 = r_0$ pour avoir $\tilde{r}_0$
$p_0 = \tilde{r}_0$
for $n = 1, 2, 3, \ldots$
$\alpha_n = \frac{r_{n-1}^T \tilde{r}_{n-1}}{p_{n-1}^T A p_{n-1}}$
$x_n = x_{n-1} + \alpha_n p_{n-1}$
$r_n = r_{n-1} - \alpha_n A p_{n-1}$
Résoudre $M\tilde{r}_n = r_n$ pour avoir $\tilde{r}_n$
$\beta_n = \frac{r_n^T \tilde{r}_n}{r_{n-1}^T \tilde{r}_{n-1}}$
$p_n = \tilde{r}_n + \beta_n p_{n-1}$

Toutes les matrices $L^\times$ ont donc disparu dans l'algorithme préconditionné. Noter aussi que l'on a rebaptisé $\hat{\alpha}_n \to \alpha_n$ et $\hat{\beta}_n \to \beta_n$, ce qui est sans conséquence. Les seules différences par rapport à l'algorithme non-préconditionné apparaissent en rouge ci-dessus. Il y a une seule étape supplémentaire qui consiste à résoudre le système $M\tilde{r}_n = r_n$ à chaque itération.

Notez que si l'on résout le système $M\tilde{r}_0 = r_0 = b$ exactement au moyen d'une factorisation LU complète de la Matrice A, i.e., M = A = LU, on trouve $\tilde{r}_0 = x^\star$. Cela conduit directement à $r_1 = 0$ et l'algorithme converge en une seule itération. Cependant le coût de cette LU est prohibitif. Toute l'idée du préconditiopnnementest de choisir une matrice M pour laquelle cette résolution est au moins un ordre de grandeur plus rapide qu'une méthode directe (au prix qu'elle ne doit plus donner la solution exacte) par exemple, une factorisation LU incomplète (ILU0), ou une factorisationnde Choleski incomplète (IC).

9 Convergence de l'algorithme CG

La semaine prochaine...