1 Analyse de convergence

Grâce à l'algorithme 1 décrit en Annexe A, il est possible de résoudre, de manière itérative, des systèmes linéaires et notamment ceux du modèle `ndt.py`. L'intérêt de cette méthode, comparée aux méthodes directes est le choix d'un critère d'arrêt lié (indirectement) à la précision de la solution comme discuté en section 3. En effet la fonction prend en argument un paramètre `rtol` qui va servir à comparer les résidus à chaque itération et a arrêter l'algorithme si celui-ci atteint une valeur inférieure à `rtol`.

Les procédés iteratifs ont de nombreux avantages mais pour en vérifier la qualité il faut tout d'abord s'assurer de sa convergence. Les graphes en figures 1 et 2 montrent le taux de convergence de cet algorithme pour les 4 régimes avec et sans préconditionnement pour un grand maillage petit maillage (393 noeuds) et un grand maillage (2008 noeuds) respectivement. Ces graphes ont été générés avec une valeur de `rtol = 1e-11` et une limite du nombre d'itérations maximales à 300[1].

Comme le montrent ces graphes, l'algorithme converge bien. En effet, à chaque itération, la norme du résidu relatif est inférieur au précédent. Et ce, quel que soit le régime et la taille de la matrice. Ceci correspond au résultat attendu puisqu'à chaque itération de GMRES, la norme du résidu est minimisée sur un espace de plus en plus grand (l'espace de Krylov gagne une dimension à chaque itération). Dès lors, la minimisation d'une même quantité sur un espace plus grand ne peut être inférieure à la minimisation sur un sous-espace.

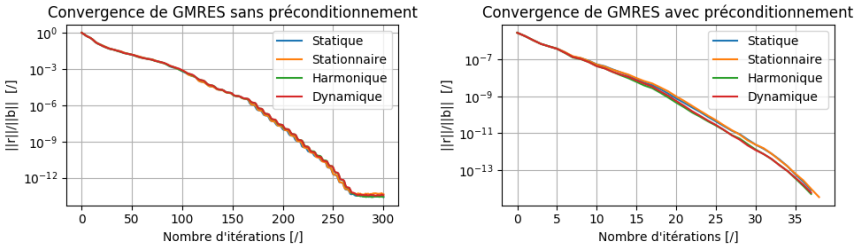

Figure 1: Convergence de GMRES pour un petit maillage

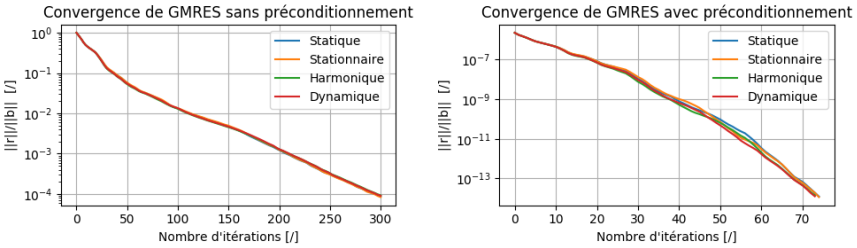

Figure 2: Convergence de GMRES pour un grand maillage

Pour les grand maillages la norme du résidu commence par baisser mais après un certain nombre d'itérations, ne s'améliore plus (ou alors le nombre d'itérations dépasse 300 et l'algorithme s'arrête). Ceci est dû aux erreurs numériques et à un certain moment les propriétés de l'espace de Krylov se détériorent et la solution ne s'améliore plus. C'est pourquoi en pratique la technique de restart est implémentée.

Ces graphes présentent de grandes différences entre l'algorithme qui utilise le préconditionnement et celui qui ne le fait pas, ceci est discuté en section 2.

2 Effet du préconditionnement

Les graphes en figures 1 et 2 montrent que l'algorithme converge beaucoup plus vite lorsque la technique de préconditionnement est appliquée. En effet, pour les deux maillages, le critère de convergence est obtenu en quelques dizaines d'itérations avec préconditionnement, tandis que sans, le critère de convergence n'est toujours pas atteint après plusieurs centaines d'itérations.

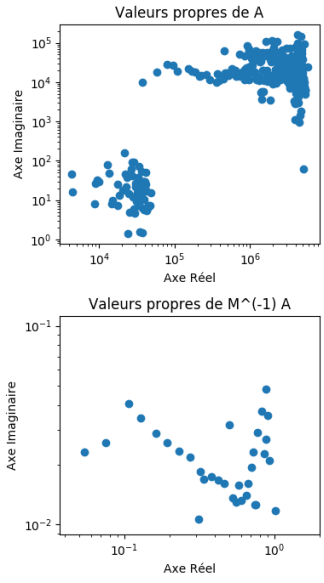

Figure 3: Valeurs propres

Ceci montre clairement que l'algorithme GMRES préconditionné offre un taux de convergence bien meilleur que GMRES sans préconditionnement. Ceci s'explique par le fait que lorsque le système est précondtionné, c'est $M^{-1}Ax = M^{-1}b$ qui est résolu plutôt que $Ax = b$. Or, comme vu au devoir 2, le conditionnement de $M^{-1}A$ est plus faible de plusieurs ordres

[1]En pratique, `rtol` n'est pas si bas (et varie plutôt entre 10^{-6} et 10^{-10}) et les itérations sont arrêtées avant d'arriver à une solution qui ne s'améliore plus, auquel cas la technique de restart est lancée.

de grandeur que le conditionnement de A. Et comme le taux de convergence de l'algorithme GMRES est régi par le conditionnement du problème et la répartition des valeurs propres. Au plus faible est le conditionnement et au plus les valeurs propres sont rassemblées, au plus l'algorithme convergera vite.

La figure 3 montre cette répartition des valeurs propres pour le régime dynamique ($f = 50Hz, vel = 1m/s$) et un maillage de 393 noeuds. Il est clair que les valeurs propres de $M^{-1}A$ sont rassemblées et très proche les unes des autres par rapport à celles de la matrice A. Ceci entrainera une meilleure convergence.

3 Qualité de la solution

Les deux sections précédentes montrent que l'algorithme converge et qu'il converge plus rapidement si la matrice est préconditionnée. Cependant, il reste à montrer que la solution vers laquelle l'algorithme converge est bien celle attendue. Une première approche est de regarder la valeur de la précision relative : $\frac{||Ax-b||_2}{||b||_2}$. Le tableau 1 reprend les précisions relatives et le nombre d'itérations pour les 4 régimes quand le système est préconditionné (1) ou pas (2) lorsqu'on applique l'algorithme avec un `rtol = 1e-7` avec un maillage de 393 noeuds.

		Statique `f=0, vel=0`	**Stationnaire** `f=0, vel=1`	**Harmonique** `f=50, vel=0`	**Dynamique** `f=50, vel=1`
(1)	**Précision** [/]	$8.39 \cdot 10^{-5}$	$6.37 \cdot 10^{-5}$	$8.45 \cdot 10^{-5}$	$5.68 \cdot 10^{-5}$
	Nombre d'itérations [/]	25	26	24	25
(2)	**Précision** [/]	$7.17 \cdot 10^{-11}$	$7.40 \cdot 10^{-11}$	$7.07 \cdot 10^{-11}$	$8.35 \cdot 10^{-11}$
	Nombre d'itérations [/]	232	233	235	234

Table 1: Précisions pour les 4 régimes

La solution converge donc vers une solution et la précision relative est excellente dans le cas du système non-préconditionné. Dans le cas du préconditionné, elle est bonne mais pas aussi bonne. Ceci vient du fait que pour le modèle préconditionné, le résidu est calculé avec $||M^{-1}(Ax-b)||_2$. et donc le résidu est une combinaison linéaire de $Ax-b$ mais comme les valeurs singulières de M^{-1} sont inférieures à l'unité[2], $||M^{-1}(Ax-b)||_2 < ||Ax-b||_2$. La norme du résidu du système préconditionné sera toujours inférieure à la précision du problème et donc la précision relative sera moins bonne pour un système préconditionné que pour un système non-préconditionné avec le même `rtol`. Il convient donc de se demander si une meilleure précision peut être obtenue avec un `rtol` plus faible et un plus grand nombre d'itérations (mais qui resterait inférieur à celui du système non-préconditionné). Le graphe 4 montre la précision relative obtenue en fonction du nombre d'itérations réalisées par GMRES (avec 393 inconnues et en régime dynamique). Celui-ci montre clairement qu'une bonne précision peut être obtenue en baissant le paramètre `rtol` et ce en un nombre considérablement plus petit d'itérations que s'il n'y a pas préconditionnement.

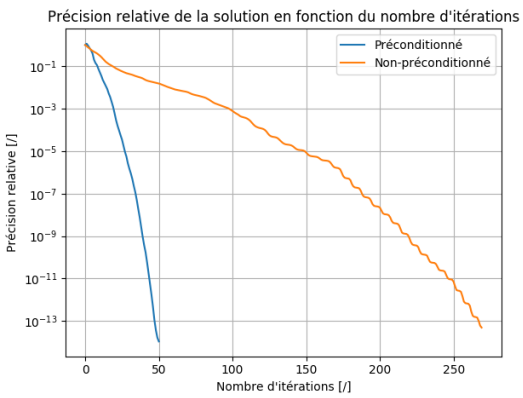

Figure 4: Précision relative selon les itérations

La précision relative donne une bonne idée de la qualité de la solution si le problème est bien posé. Il est également intéressant de voir si la solution s'approche de la solution exacte x^*. En prenant la solution donnée par le solveur de numpy comme solution exacte x^*, il est possible d'analyser $\frac{||x-x^*||_2}{||x^*||_2}$ qui donne une idée de la proximité de la solution obtenue par rapport à la solution exacte. Cette analyse montre que pour tous les régimes et toutes les tailles de noeuds, au plus `rtol` est faible au plus on se rapproche de la solution exacte (jusqu'au moment où la solution ne converge plus, cfr. section 1)

[2]Dans le modèle `ndt`, la matrice A le module des valeurs singulières de l'ordre de 10^3 à 10^6, et comme M est la décomposition ILU(0) de A, $M \approx A$ et donc les valeurs singulières de M^{-1} sont approximativement égales à l'inverse de valeurs singulières de A et donc inférieures (en module) à 1.

A Annexe : Algorithme GMRES avec préconditionnement

L'algorithme GMRES avec préconditionnement peut s'écrire comme suit. Considérons la résolution du système linéaire $Ax = b$. ILU(0) est utilisé comme préconditionneur et M est la décomposition ILU(0) de A : $M = \hat{L}\hat{U}$. L'itéré initial est $u_0 = 0$, sauf si le restart est implémenté, auquel cas il peut être non-nul.

Algorithm 1 Algorithme GMRES préconditionné

$r_0 = M^{-1}(b - Au_0)$; $\beta = ||r_0||_2$; $v_1 = r_0/\beta$ ▷ M est la décomposition ILU(0) de A : $M = \hat{L}\hat{U}$
for m = 1, 2, ..., max_iter **do**
  $w = M^{-1}Av_m$ ▷ Multiplication par M^{-1} se fait en résolvant le système $\hat{L}\hat{U}w = Av_m$
  for i = 1, .., j **do**
    $h_{i,m} = v_i^* w$
    $w = w - h_{i,m}v_i$
  end for
  $h_{m+1,m} = ||w||_2$
  $v_{m+1} = w/h_{m+1,m}$

  $V_m = [v_1, v_2, ..., v_m]$; $\bar{H}_m = [h_{i,j}]$ ▷ $\bar{H}_m$ est $[h_{i,j}]$ pour $i = 1, 2, ..., m+1$ et $j = 1, 2, ..., m$
  $\min_{y \in \mathbb{R}^m} ||\beta e_1 - \bar{H}_m y||_2$ ($= ||r_m||_2$) ▷ Fait avec la décomposition QR

  if $||r_m||_2 < rtol$ **then**
    break
  end if
end for
if $||r_m||_2 > rtol$ **then**
  $u_0 = u_0 + V_m y$
  restart
end if
$u = u_0 + V_m y$

A l'itération n de cet algorithme, les vecteurs et matrices suivants sont maintenus en mémoire. Notons N le nombre d'inconnues du système (A est de dimension $N \times N$).

- Tout au long de l'algorithme, u_0, b et r_0 ne changent pas et sont des vecteurs de taille N. A est une matrice de taille $N \times N$ (sauvegardée sous format CSR), β est un scalaire. M est une matrice sous format CSR qui représente une matrice de taille $N \times N$ et qui est représentative des matrices $\hat{L}$ et $\hat{U}$, le résultat de la décomposition ILU(0) de A.
- Ensuite, certaines variables qui dépendent de l'itération. $||r_n||_2$ est un scalaire représentant la norme du résidu de la solution par rapport au système à l'itération n. w est un vecteur de taille N.
- Il y a également la matrice $\bar{H}_n$ qui est de dimension $(n+1) \times n$ (et qui est donc plus grande à chaque itération).
- La matrice V_n de dimension $N \times n$ (n vecteurs de dimension N).
- e_1 est un vecteur de dimension $n+1$ est tel que $e_1 = [1, 0, 0, ..., 0]^T$ et donc y est un vecteur de dimension n et est la solution du problème de minimisation par les moindres carrés.
- Si n était la dernière itération, on va appliquer le produit $V_n y$ qui donnera un vecteur de taille N (qu'on additionne à u_0) pour donner le vecteur solution u de taille N également