LEPL1110 - Éléments finis - Projet

Vincent Degrooff - NOMA : 09341800

13 Mai 2021

1 Précision du programme

Pour chacun des 5 maillages, notons l'approximation du potentiel vecteur $\hat{a}_i(x, y)$ pour $i = 1 \dots 5$, i croissant avec le nombre de noeuds.

Pour mesurer l'ordre de convergence de mon programme d'éléments finis, je dois mesurer l'erreur commise par la solution discrète. Malheureusement, je ne possède pas la solution exacte. Sur base des solutions $\hat{a}_i(x, y)$ j'ai donc mesuré différentes valeurs pour valeurs pour valider mon programme :

1. $\|\hat{a}_i(x, y)\|$: la norme du résultat
2. $\|\hat{a}_{i+1}(x, y) - \hat{a}_i(x, y)\|$: erreurs consécutives
3. $\|\hat{a}_5(x, y) - \hat{a}_i(x, y)\|$: erreur par rapport à une référence (le plus grand maillage)

Les normes considérées sont les normes L^2 et H^1. La longueur caractéristique h des maillages est définie comme la moyenne des longueurs caractéristiques des éléments qui le compose, au nombre de n_e. Cette longueur a simplement été choisie comme la racine carré de l'aire de l'élément, notée A :

$$h = \frac{1}{n_e} \sum_{j=1}^{n_e} \sqrt{A_j}$$

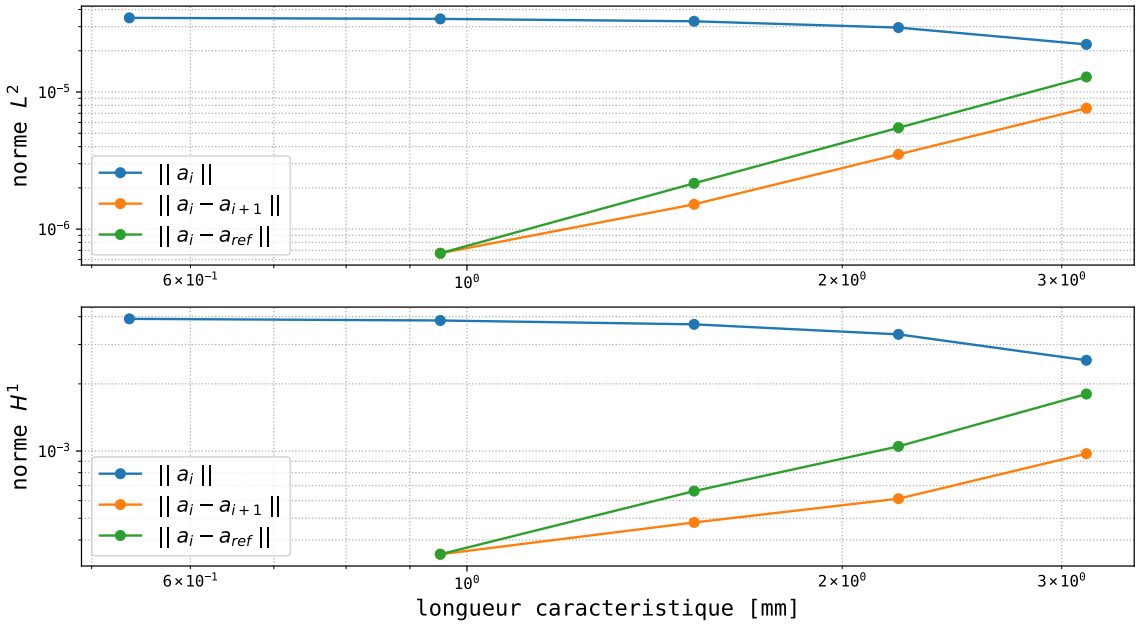

FIGURE 1 – Mesure des erreurs en fonction de la taille des éléments

Tout d'abord, on constate bien qu'en diminuant la taille des éléments, la norme de la solution, en bleu, se "stabilise". C'est bon signe, mais ce n'est pas suffisant pour conclure. On doit encore mesurer la pente des courbes verte et orange qui correspondent aux différences entre les solutions.

Pour calculer $\|\hat{a}_{i+1}(x,y) - \hat{a}_i(x,y)\|$, j'ai d'abord cherché à obtenir la solution du "petit maillage" sur chacun des noeuds du plus grand maillage. Ensuite, j'ai simplement intégré le carré de la différence entre les deux fonctions.

Enfin, j'ai fait une approximation linéaire dans un graphique log-log. Si l'on note l'erreur $e = C\ h^p$, je peux finalement approximer l'ordre de précision de mon programme par rapport à la taille des éléments du maillage.

type de norme	$\|\hat{a}_{i+1}(x,y) - \hat{a}_i(x,y)\|$	$\|\hat{a}_5(x,y) - \hat{a}_i(x,y)\|$
L^2	2.051	2.481
H^1	0.840	1.368

TABLE 1 – Ordre de précision p de la méthode d'éléments finis

Pour la première colonne, les erreurs consécutives, on constate que l'ordre est à peu près en accord avec le résultat théorique qui annonce un ordre de 2 pour la norme L^2 et de 1 pour la norme H^1. Pour la deuxième méthode, j'utilise $\hat{a}_5(x,y)$ comme référence, comme s'il s'agissait de la solution exacte du problème. La convergence est donc logiquement plus rapide puisque qu'il est plus simple de s'en approcher, en augmentant quelque peu le nombre de noeuds.

2 Fonctionnement de l'algorithme

Voici le détail des étapes clés dans le calcul du potentiel vecteur :

— Re-numérotation des noeuds pour améliorer le conditionnement de la matrice

— Assemblage de la matrice en format COO

— Tri des éléments de la matrice creuse + conversion en CSR

— Factorisation ILU(0)

— Gradient conjugué pré-conditionné

Afin d'obtenir de bonnes performances, j'utilise le gradient conjugué préconditionné (PCG) pour résoudre le système matriciel $Ax = b$. Il me semblait également important d'éviter d'assembler la matrice à chaque étape du PCG. C'est pourquoi j'ai construit deux structures de matrice creuse. La première, une structure COO (row, col, value) me permet d'assembler la matrice A facilement.

Lorsque cette opération est terminée, je trie les éléments, notés *ijk*, de la matrice creuse COO et j'additionne les éventuels doublons. Je la transforme ensuite en CSR (Compressed sparse row), plus efficace lors des calculs vectoriels. Ce format est similaire au COO sauf qu'il compresse les lignes.

Lorsque la matrice est sous forme CSR, j'effectue le pré-conditionnement. Par simplicité, j'ai décidé d'implémenter une factorisation ILU(0). Finalement, j'effectue classiquement la procédure du gradient conjugué.

Un exemple concret de matrice creuse est fourni à la figure 2. On y remarque qu'une matrice de taille $n \times n$ avec NNZ éléments non nuls peut être stockée au format :

— COO : en 3 vecteurs de taille NNZ

— CSR : en 2 vecteurs de taille NNZ et un taille $n + 1$

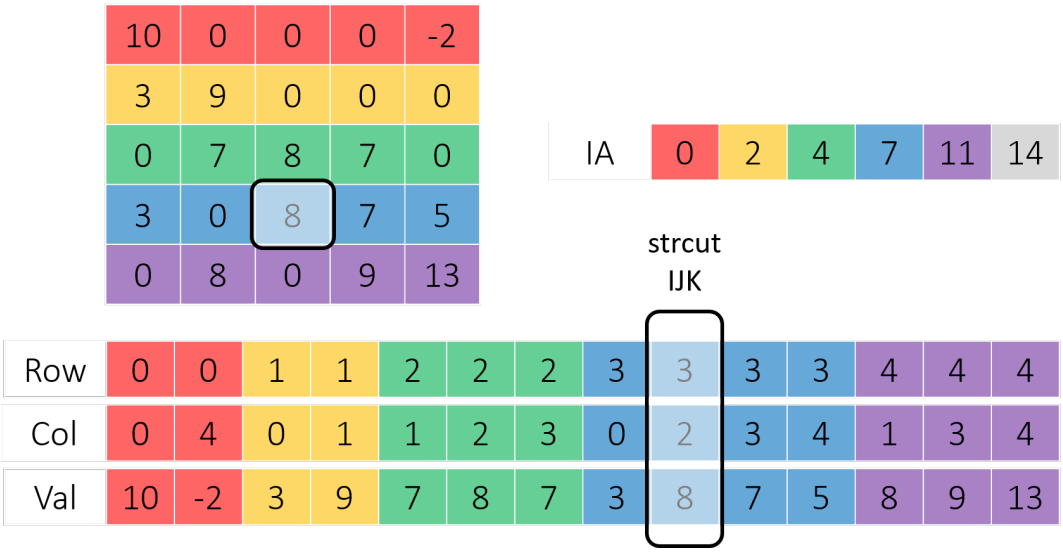

FIGURE 2 – Exemple de matrice creuse

3 Performances du code

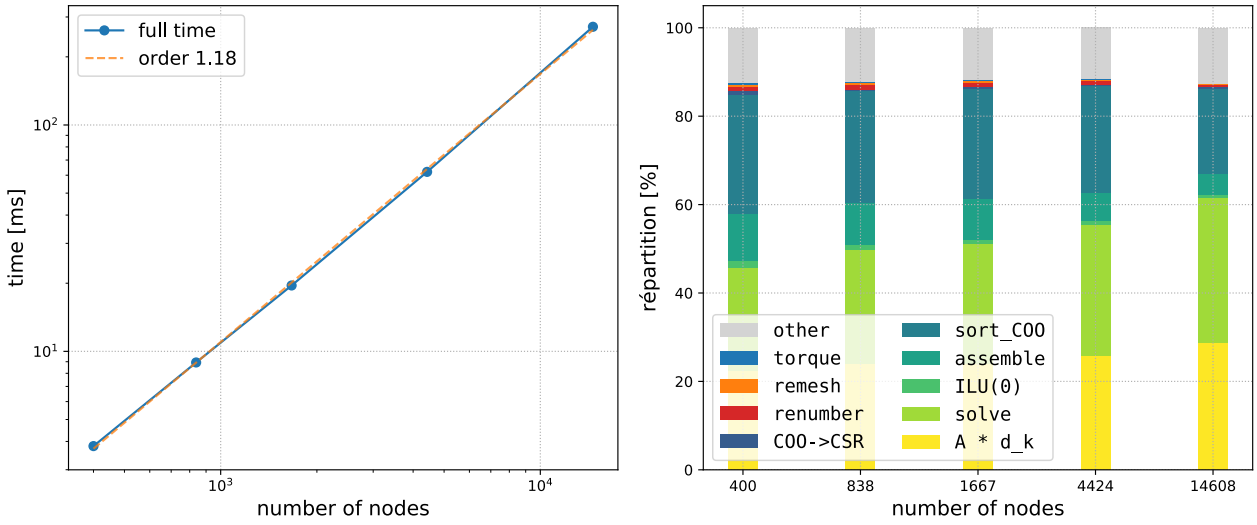

FIGURE 3 – Temps de calcul du programme en fonction du maillage et sa répartition dans différents postes

Pour mesurer le temps de calcul du code, j'ai exécuté les quatre fonctions lors de 50 rotations successives. Comme la première exécution est légèrement plus lente, je ne la prend pas en compte dans mon analyse. Cette démarche me semble appropriée puisque cette première itération ne représente qu'une fraction du temps de calcul sur une simulation complète.

Le graphique de gauche à la figure 3 représente le temps de calcul complet pour une itération temporelle, noté T, en fonction du nombre de noeuds n. En effectuant une régression linéaire sur le logarithme de T et le logarithme de n, on obtient un ordre de convergence légèrement supérieur à l'unité :

$$\log T \approx 1.18 \ \log n + C_1 \iff T \approx C_2 \, n^{1.18}$$

Sur le graphique de droite, on peut voir le temps pris par les différentes parties du programme. On constate tout d'abord que les opérations de remaillage et de calcul du couple sont négligeables face

aux différentes parties du calcul du potentiel magnétique. La numérotation des noeuds est également très rapide, mais j'y reviens plus en détails dans la section suivante.

On peut donc affirmer que la rapidité du programme est surtout dictée par deux opérations :

— l'assemblage de la matrice COO et son tri pour en retirer les doublons

— le PCG avec les produits matrice-vecteur et la résolution des deux systèmes triangulaires (back subsitution et forward substitution)

La rapidité du préconditionneur ILU(0) le rend d'autant plus intéressant puisqu'il permet de réduire considérablement le nombre d'itérations des gradients conjugués.

4 Impact de la numérotation

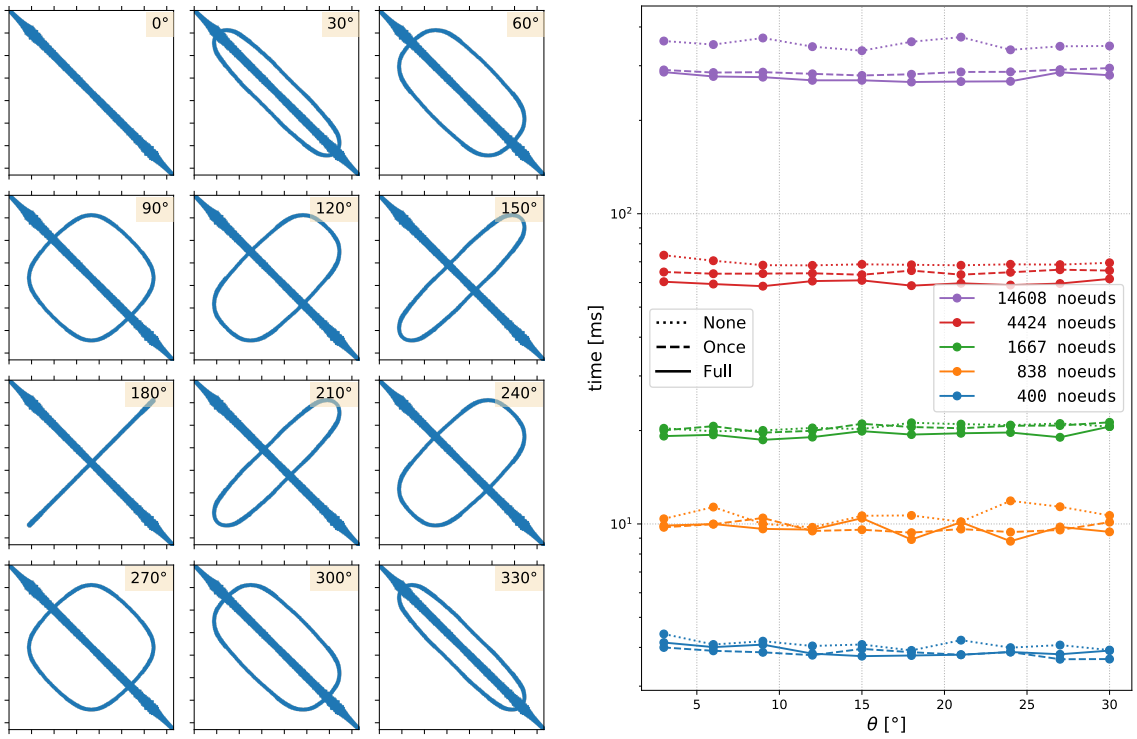

FIGURE 4 – Impact de la numérotation sur la matrice (size=14608) et sur le temps de calcul

Si l'on se contente de re-numéroter les noeuds uniquement à l'itération 0, la forme de la matrice évoluera au cours des itérations suivantes puisque, suite à la rotation du rotor, les triangles de la bande GAP changent de noeuds. Cette évolution est fournie à la figure 4. Au delà d'être esthétiquement plaisant, ce graphique montre que la largeur de bande de la matrice augmente et est maximale lorsque la rotation est de 180 degrés, comme on pouvait s'y attendre.

J'ai donc décidé de re-numéroter les noeuds (dans la direction x) à chaque itération temporelle puisque, comme on l'a vu à la figure 3, cette re-numérotation prend un temps plus que négligeable : inférieur à 0.5%.

Sur la partie droite de la figure 4, les temps de calcul pour trois méthodes sont représentés : sans aucune re-numérotation, avec une seule à la première itération et avec des re-numérotations à chaque étape. On remarque tout d'abord que pour les plus petits maillages, cette opération n'apporte

presque aucune plus value. Ensuite, on remarque logiquement que la numérotation à chaque étape permet d'augmenter légèrement les performances même pour des rotations faibles de 5 degrés.

5 Relation Couple - position angulaire

Dans la position de référence, avec les bobines A activées, le couple produit sur le rotor est négatif. Ce denier accélère et commence donc à tourner dans le sens négatif, c'est à dire de manière horlogique. Pour simplifier la lecture du prochain graphique, j'ai décidé de changer les signes des rotations et des couples afin d'avoir des valeurs positives.

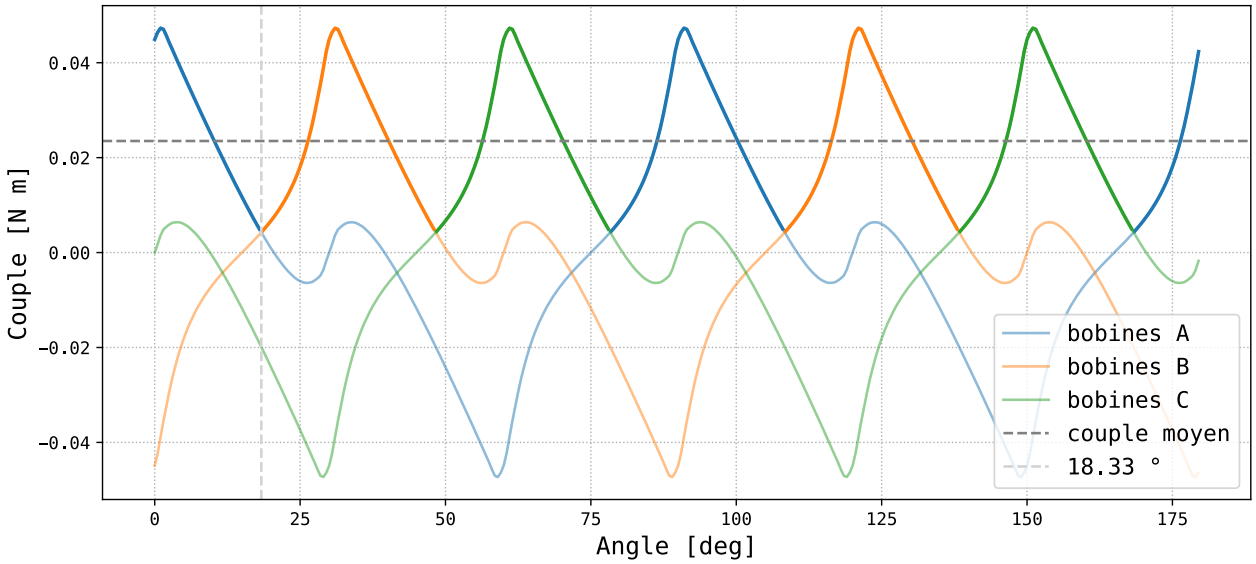

FIGURE 5 – Couple exercé sur le rotor au cours de sa rotation

Pour obtenir le plus grand couple moyen, il faut simplement toujours choisir la bobine qui fournit le plus grand couple. Dans notre cas, il faut changer de bobine tous les 30° avec un décalage initial d'à peu près 18.33°.

Références

[1] L.N. TREFETHEN et D. BAU. *Numerical Linear Algebra*. Siam, 1997.

[2] P. AMESTOY et A. BUTTARI. *Sparse Linear Algebra : Direct Methods, advanced features*. 2013. URL : `http://buttari.perso.enseeiht.fr/docs/ALC_Toulouse.pdf` (visité le 10/04/2020).

[3] Xiaoye Sherry LI. *Factorization-based Sparse Solvers and Preconditioners*. 2013. URL : `https://archive.siam.org/students/g2s3/2013/lecturers/XSLi/Lecture-Notes/sherry.pdf` (visité le 11/04/2020).

[4] Michael T. HEATH et Edgar SOLOMONIK. *Parallel Numerical Algorithms*. 2015. URL : `https://solomonik.cs.illinois.edu/teaching/cs554_fall2017/slides/slides_08.pdf` (visité le 10/04/2020).

[5] Yaoliang YU. *Graph Representation and Ordering*. 2018. URL : `https://cs.uwaterloo.ca/~y328yu/mycourses/475-2018/lectures/lec05.pdf` (visité le 25/04/2020).

Annexe A Magnétostatique non-linéaire

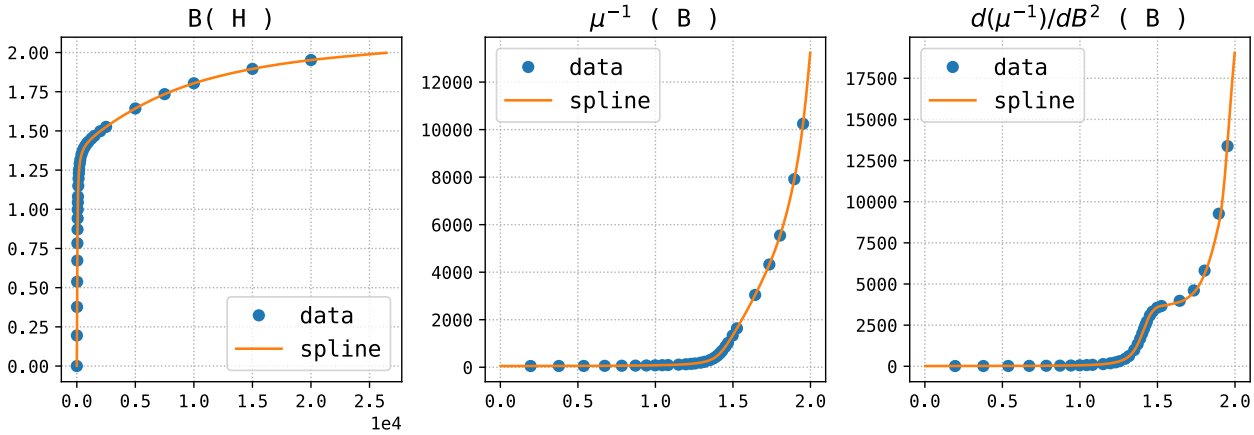

FIGURE 6 – Paramètres nécessaires pour la modélisation non-linéaire

Nous souhaitons résoudre l'équation non-linéaire suivante :

$$\boldsymbol{f}(\mathbf{x}) = A(\mathbf{x}) \cdot \mathbf{x} - b = 0$$

Pour y parvenir, nous allons utiliser l'algorithme de Newton-Rhapson pour des sytèmes non-linéaires. Le potentiel magnétique $a(x, y)$ qui est obtenu à l'itération n est notée $\mathbf{x}^n$:

$$\left.\frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}^n} \cdot \left(\mathbf{x}^{n+1} - \mathbf{x}^n\right) = -\boldsymbol{f}(\mathbf{x}^n)$$
$$\left(\left.\frac{\partial A}{\partial \mathbf{x}}\right|_{\mathbf{x}^n} \cdot \mathbf{x}^n + A(\mathbf{x}^n)\right) \cdot \Delta\mathbf{x} = b - A(\mathbf{x}^n) \cdot \mathbf{x}^n$$

Le terme $\frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}$ est fourni dans les slides du CM9. On en déduit une forme plus explicite, où $b^2 = \|B\|^2 = \left(\frac{\partial a}{\partial x}\right)^2 + \left(\frac{\partial a}{\partial y}\right)^2$:

$$A_{ij}(\mathbf{x}^n) = \int_\Omega \mu^{-1}(b^2) \left(\frac{\partial \phi_i}{\partial x}\frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y}\frac{\partial \phi_j}{\partial y}\right)$$
$$\left.\frac{\partial A_{ik}}{\partial \mathbf{x}_j}\right|_{\mathbf{x}^n} \cdot \mathbf{x}_k^n = \int_\Omega 2 \left.\frac{d\mu^{-1}}{db^2}\right|_{b^2} \left(\frac{\partial \phi_i}{\partial x}\frac{\partial \mathbf{x}^n}{\partial x} + \frac{\partial \phi_i}{\partial y}\frac{\partial \mathbf{x}^n}{\partial y}\right) \cdot \left(\frac{\partial \phi_j}{\partial x}\frac{\partial \mathbf{x}^n}{\partial x} + \frac{\partial \phi_j}{\partial y}\frac{\partial \mathbf{x}^n}{\partial y}\right)$$

Pour calculer cette intégrale, nous devons évaluer la perméabilité et sa dérivée par rapport à b^2. Nous avons à notre disposition une liste de couple (H, B) qui nous permet d'évaluer la perméabilité par la relation $B = \mu H$. Cependant, nous souhaitons calculer μ^{-1} et $\frac{d\mu^{-1}}{db^2}$ pour n'importe quelles valeurs de b^2. C'est la raison pour laquelle j'ai décidé d'utiliser une interpolation par splines cubiques de H en fonction de B. J'ai préféré cette option à une simple interpolation linéaire par morceaux puisqu'elle a l'avantage de préserver la continuité de la dérivée. Cette interpolation est représentée à la figure 6.

Avant d'effectuer l'itération de l'algorithme de Newton, j'ai besoin d'un itéré initial. Pour obtenir cette valeur initiale $\mathbf{x}^0$, soit je résous le système linéaire avec une perméabilité de l'acier constante,

soit je reprends la solution de l'itération précédente. En général, je reprends la solution précédente pour gagner du temps, sauf s'il y a un changement de bobine, ou si le $\Delta\theta$ est trop important.

Ensuite, grâce à l'interpolation par splines cubiques, je peux assembler la matrice jacobienne de l'algorithme de Newton-Rhapson sur base des deux intégrales définies plus haut. J'effectue finalement l'algorithme jusqu'à ce que le Δx soit suffisamment petit.

Les effets de la non linéarité ne sont ressentis que pour d'importantes valeurs de j_s. En effet, si l'on gardait la valeur de l'énoncé $j_s = 8.8464 \times 10^5$ [A s^{-2}], le champ $B \approx 0.04$ [T] resterait trop faible et l'on atteindrait pas la saturation dans le diagramme $B - H$.

J'ai donc mesuré le couple pour différentes valeurs de densité de courant : $j_s = k \cdot 8.8464 \times 10^5$ [A s^{-2}], avec k variable.

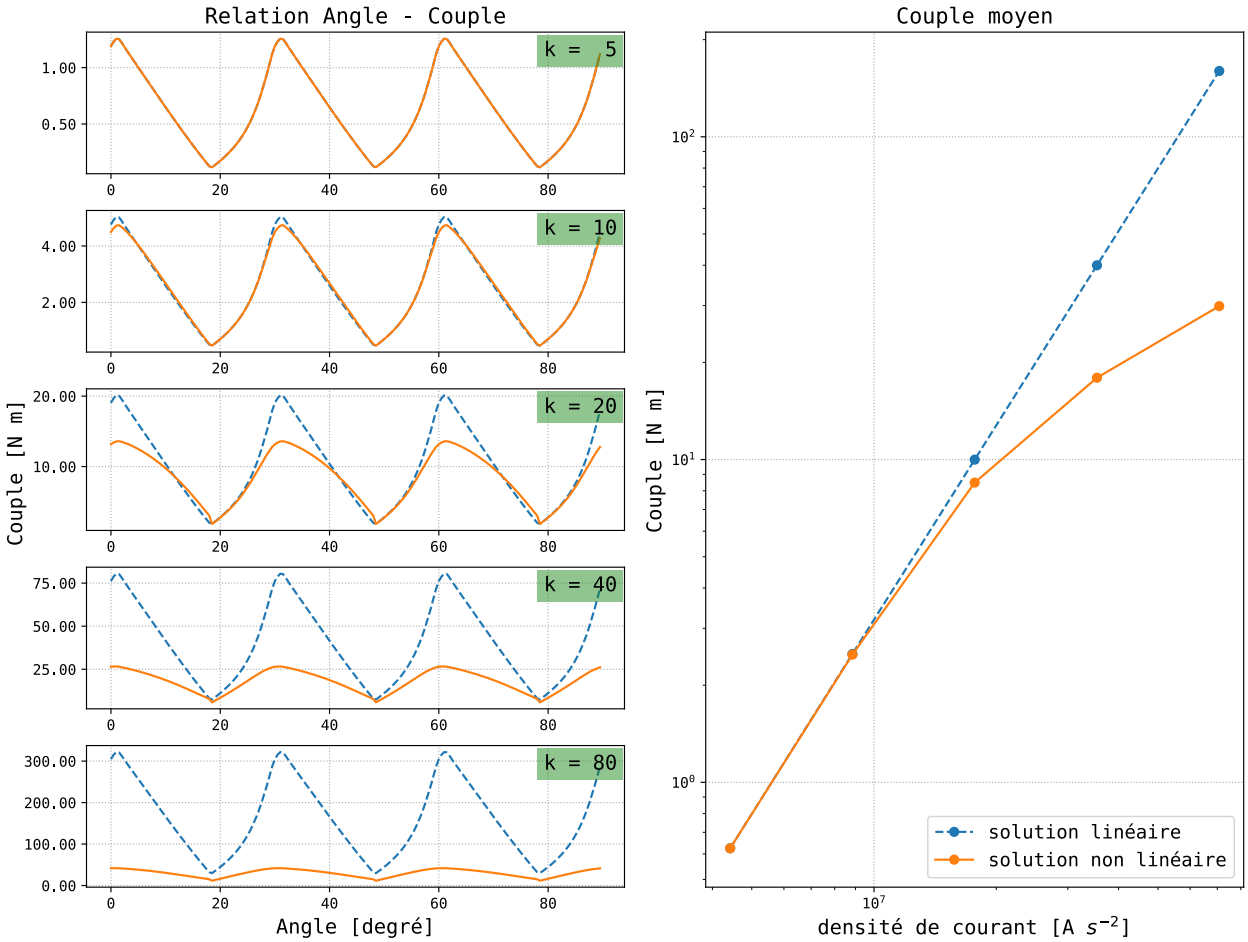

FIGURE 7 – Impact de la non-linéarité sur le couple

On observe bien, comme attendu que la solution linéaire fournit un couple qui évolue quadratiquement avec le courant (droite de pente 2 sur le graphe log-log). Au contraire, la solution non-linéaire démontre bien que l'acier sature pour de trop grandes valeurs de B.