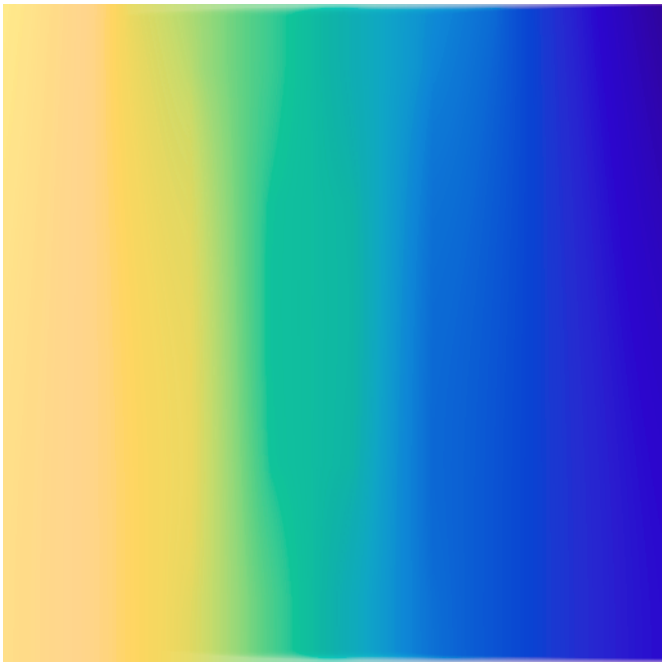


Rapport du projet de la Méthode de base réduite pour la résolution d'EDPs dépendantes de paramètre

Jincheng KE

Une présentation du projet sur les résultats obtenus.

Mars 2024

Table des matières

Chapitre 1

Mise en forme du problème paramétrique

1.1 Solution numérique des différents paramètres

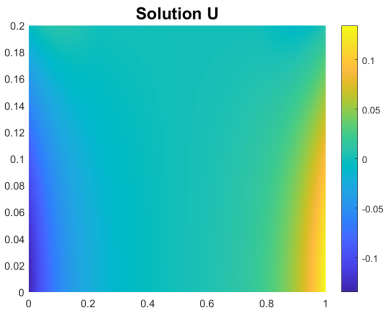


FIGURE 1.1 – $\kappa_1 = 1, \kappa_2 = 1$

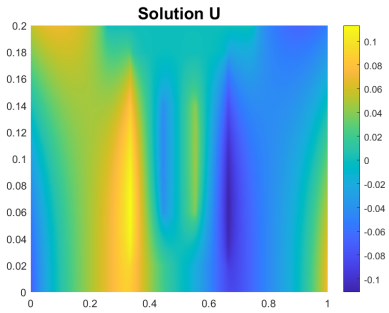


FIGURE 1.2 – $\kappa_1 = 0.1, \kappa_2 = 1$

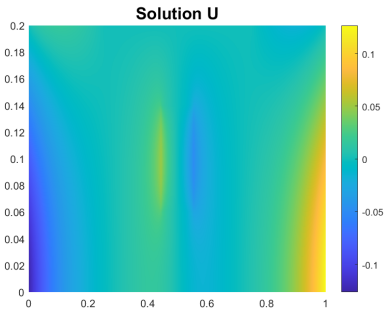


FIGURE 1.3 – $\kappa_1 = 1, \kappa_2 = 0.1$

1.2 La décompostition affince

On a la matrice $\mathbf{A}$ peut être décomposée par

$$\mathbf{A} = \mathbf{A}_0 + \kappa_1 \mathbf{A}_1 + \kappa_2 \mathbf{A}_2$$

avec

$$\mathbf{A}_q = \mathbf{K} \cdot \mathbb{1}_{\{\text{RefTriangles}=q\}}, \forall q = 0, 1, 2$$

Et comme on a

$$\mathbf{L} = \mathbf{M} \cdot \mathbf{F} + \mathbf{S}_1 \cdot \mathbf{G}_1 + \mathbf{S}_2 \cdot \mathbf{G}_2 + \mathbf{S}_4 \cdot \mathbf{G}_4 + \mathbf{S}_5 \cdot \mathbf{G}_5 - \mathbf{A} \cdot \mathbf{\Phi}_r$$

Il y a que la matrice $\mathbf{A}$ qui dépend du paramètre κ, on peut donc décomposer le vecteur $\mathbf{L}$ en décomposant la matrice $\mathbf{A}$, on obtient la formule suivante :

$$\mathbf{L} = \mathbf{L}_0 + \kappa_1 \mathbf{L}_1 + \kappa_2 \mathbf{L}_2$$

avec

$$\mathbf{L}_0 = \mathbf{M} \cdot \mathbf{F} + \mathbf{S}_1 \cdot \mathbf{G}_1 + \mathbf{S}_2 \cdot \mathbf{G}_2 + \mathbf{S}_4 \cdot \mathbf{G}_4 + \mathbf{S}_5 \cdot \mathbf{G}_5 - \mathbf{A}_0 \cdot \mathbf{\Phi}_r$$

et

$$\mathbf{L}_q = -\mathbf{A}_q \cdot \mathbf{\Phi}_r, \forall q = 1, 2$$

cf. code `FE_assemblages.m`

1.3 Validation numérique

Pour vérifier la décomposition affine pour la matrice $\mathbf{A}$ et pour le vecteur $\mathbf{L}$, on compare l'erreur absolue et l'erreur relative entre la solution de référence $\mathbf{U}_{\text{ref}}$ de l'équation $\mathbf{A}_{\text{ref}} \cdot \mathbf{U}_{\text{ref}} = \mathbf{L}_{\text{ref}}$ et la solution $\mathbf{U}$ de l'équation avec la composition affine $(\mathbf{A}_0 + \kappa_1 \mathbf{A}_1 + \kappa_2 \mathbf{A}_2) \cdot \mathbf{U} = \mathbf{L}_0 + \kappa_1 \mathbf{L}_1 + \kappa_2 \mathbf{L}_2$. Pour $\kappa = (0.1, 1)$, l'erreur absolue est environ $4.658e - 14$, l'erreur relative est environ $1.123e - 14$. Pour $\kappa = (1, 0.1)$, l'erreur absolue est environ $1.408e - 14$, l'erreur relative est environ $5.171e - 15$.

L'erreur est assez petite pour pouvoir valider cet algorithme de la décomposition affine.

Réduction par l'approche POD

Construction d'une base réduite par POD

L'implémentation cf. `MAIN_RBPOD.m`

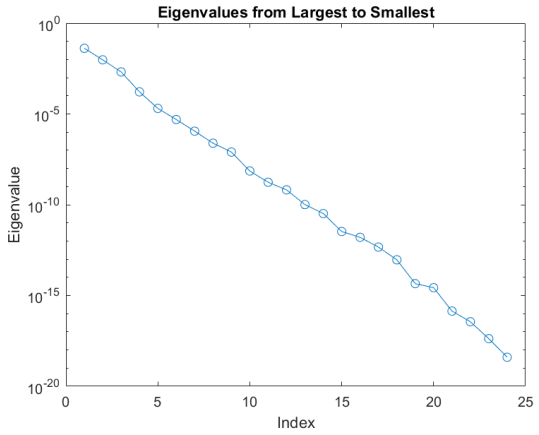


FIGURE 1.4 – type cartesien avec$n_1 = n_2 = 5$

Pour le plan d'expérience cartésien avec $n_1 = n_2 = 5$, on a observé que la vitesse de la décroissance des valeurs propres est uniforme et rapide, il y a toujours une tendance décroissante avec la même vitesse. De plus, la plus petite valeur propre est environ `1e-19`. Comme il y a une tendance décroissante avec la même vitesse, et la plus petite valeur propre ici est déjà assez petit, il semble que le problème est réductible. Mais si on veut une précision plus précise, il semble que c'est mieux de prendre plus d'échantillons.

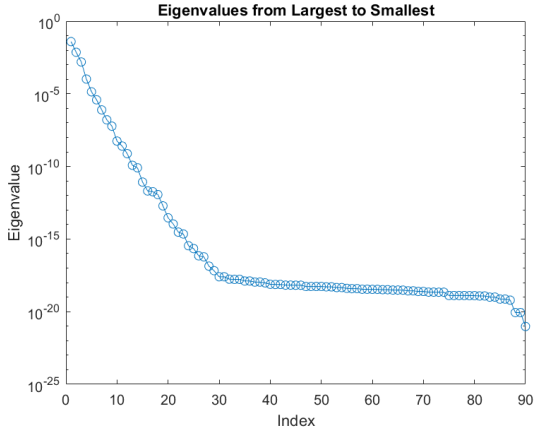


FIGURE 1.5 – type cartesien avec$n_1 = n_2 = 12$

Pour le plan d'expérience cartésien avec $n_1 = n_2 = 12$, on a observé que la tendance de la vitesse de la cécroissance uniforme et rapide se poursuit jusqu'à $n = 30$. Quand $n \geq 30$, la vitesse devient vraiment petite, les valeurs propres se stabilisent, et au final, la plus petite valeur propre est environ `1e-20`. Comme les valeurs propres se stabilisent quand $n \geq 30$, il semble que le nombre de RB de POD ne permettra plus d'améliorer la précision quand il est grand.

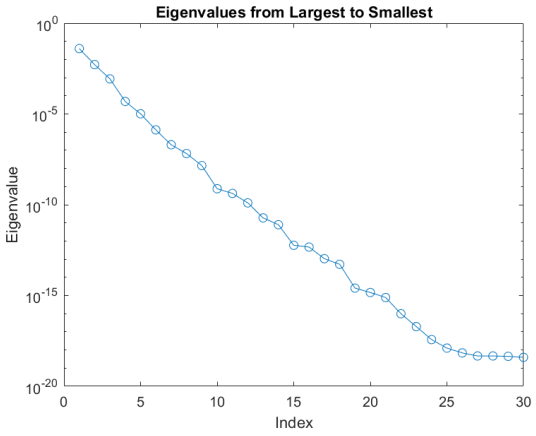


FIGURE 1.6 – type 'random' avec$n = 35$

Pour le plan d'expérience aléatoire avec $n = 35$, on a observé que globalement, il y a la même tendance de la décroissance rapide des valeurs propres avec le cas de l'expérience cartésien en prenant $n_1 = n_2 = 5$. La plus petit valeur propre est aussi environ `1e-19`. Mais localement, on peut voir que la vitesse de décroissance se varie. Cette variation de la vitesse locale est produite par la sensibilité à l'échantillonnage.

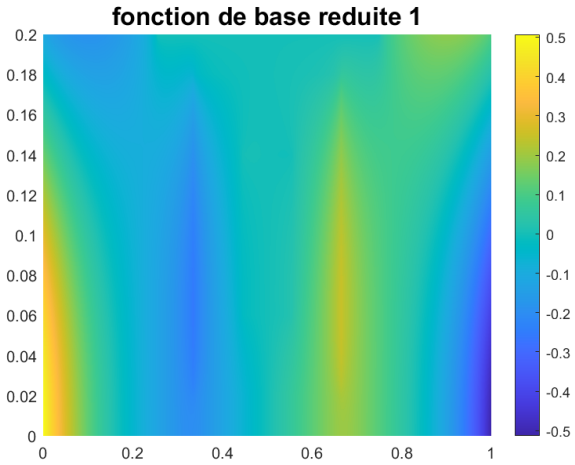


FIGURE 1.7 – $n_1 = n_2 = 12$ base 1

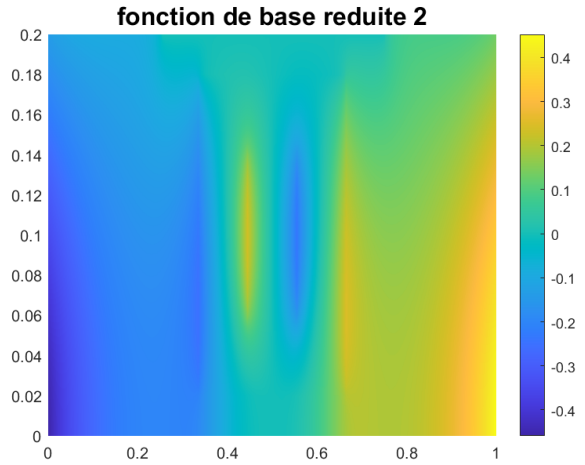


FIGURE 1.8 – $n_1 = n_2 = 12$ base 2

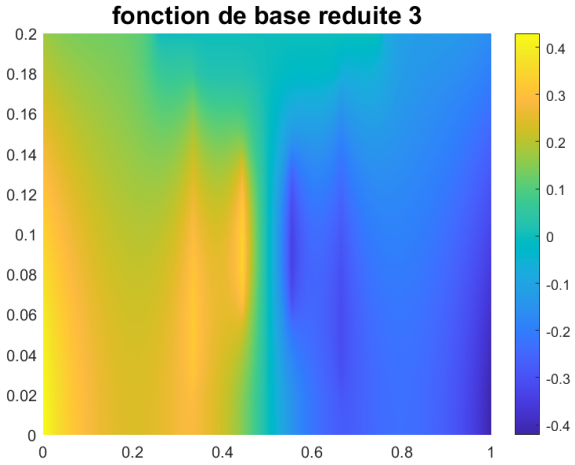


FIGURE 1.9 – $n_1 = n_2 = 12$ base 3

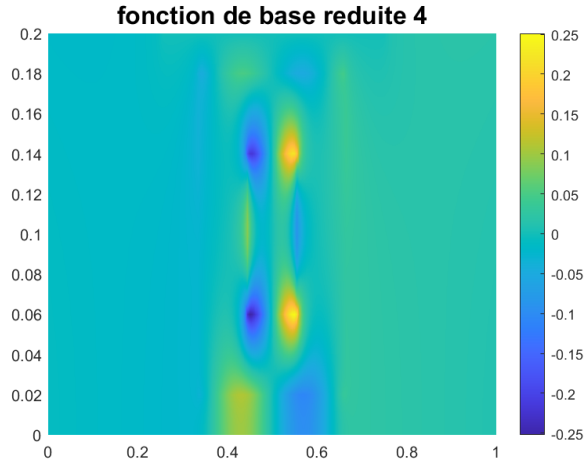


FIGURE 1.10 – $n_1 = n_2 = 12$ base 4

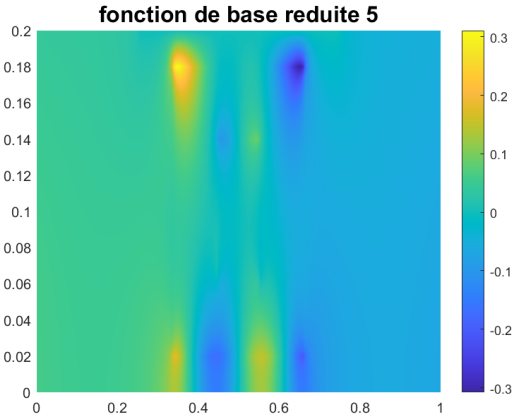


FIGURE 1.11 – $n_1 = n_2 = 12$ base 5

Les images FIGURE 1.7, 1.8, 1.9, 1.10et 1.11 sont les 5 prémières RB de POD pour le plan d'expérience cartésien avec $n_1 = n_2 = 12$.
L'implémentation cf. `MAIN_RBPOD.m`

1.4 Solveur base reduite

1.4.1 La décompostition affine

On a

$$\mathbf{A}^{\mathrm{rb}} = \mathbf{P}^{\mathrm{T}} \cdot \mathbf{A} \cdot \mathbf{P}$$

et

$$\mathbf{L}^{\mathrm{rb}} = \mathbf{P}^{\mathrm{T}} \cdot \mathbf{L}$$

En utilisant la décomposition affine de $\mathbf{A}$ et $\mathbf{L}$, on aura la décomposition suivante :

$$\mathbf{A}^{\mathrm{rb}} = \mathbf{A}_0^{\mathrm{rb}} + \kappa_1 \mathbf{A}_1^{\mathrm{rb}} + \kappa_2 \mathbf{A}_2^{\mathrm{rb}}$$

$$\mathbf{L}^{\mathrm{rb}} = \mathbf{L}_0^{\mathrm{rb}} + \kappa_1 \mathbf{L}_1^{\mathrm{rb}} + \kappa_2 \mathbf{L}_2^{\mathrm{rb}}$$

avec

$$\mathbf{A}_q^{\mathrm{rb}} = \mathbf{P}^{\mathrm{T}} \cdot \mathbf{A}_q \cdot \mathbf{P}$$

$$\mathbf{L}_q^{\mathrm{rb}} = \mathbf{P}^{\mathrm{T}} \cdot \mathbf{L}_q, \forall q = 0, 1, 2$$

L'implémentation cf. `RB_reduced_decomp.m`

Compléxité

Pour `RB_solve.m`, commme on a déja décomposé affinement, alors toutes les opérations sont sur la matrice de taille $N \times N$ ($N \approx 30$) et le vecteur de taille N, donc la complécité est $\mathcal{O}(N^3)$.

Validation numérique

L'implémentation cf. `MAIN_RBSOLVER.m`

On va tracer les erreurs d'approximation base réduite maximales (max error) et moyennes (mean error) en fonction de N avec $N = 4, 8, 12, 16, 20, 24, 28, 32$ et on va tracer également erreurs de l'approximation RB optimale (error2).

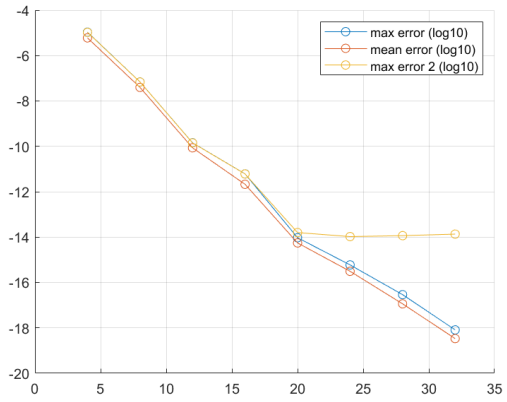


FIGURE 1.12 – type cartesien avec$n_1 = n_2 = 12$

Pour le plan d'expérience cartésien avec $n_1 = n_2 = 12$, on a observé que avec le nombre de RB qui devient de plus en plus grand, l'erreur maximale et l'erreur moyenne devient aussi de plus en plus petit, et comme la courbe est une ligne à l'échelle de semi-log, on a la vitesse de convergence est exponentielle. Pourtant, pour l'erreur de l'approximation RB optimale, elle décroît aussi vite que l'erreur maximale / moyenne au début, mais jusqu'à $n = 20$. Quand $n \geq 20$, elle se stabilise à valeur de environ `10e-14`. Cela implique on a atteint à la limite de cette échantillonnage. Cette valeur est la norme de projection de $\mathbf{U}$ dans l'espace $\mathbf{V}_{\mathrm{NTrain}}^{\perp}$ (NTrain = 144 ici).

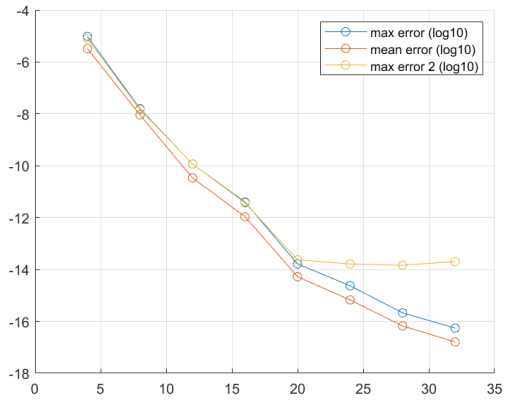


FIGURE 1.13 – type 'random' avec$n = 35$

Pour le plan d'expérience aléatoire avec $n = 35$, on a observé que avec le nombre de RB qui devient de plus en plus grand, l'erreur maximale et l'erreur moyenne devient aussi de plus en plus petit, et comme la courbe est une ligne à l'échelle de semi-log, on a la vitesse de convergence est exponentielle. Pourtant, pour l'erreur de l'approximation RB optimale, elle décroît aussi vite que l'erreur maximale / moyenne au début, mais jusqu'à $n = 20$. Quand $n \geq 20$, elle se stabilise à valeur de environ `10e-14`. On peut expliquer ce phénomène avec la même raison pour le plan d'expérience cartésien.

Chapitre 2

Réduction par l'approche Greedy

2.1 Calcul du résidu base réduite

Le coûte mémoire de `Respart_ll` est Q_l^2.
Le coûte mémoire de `Respart_la` est $Q_l \cdot Q_a \cdot N$.
Le coûte mémoire de `Respart_aa` est $Q_a^2 \cdot N^2$.
Pour `Respart_ll`, en utilisant la symétrie de la matrice B^-1, on peut optimiser le coûte mémoire à $0.5 * Q_l^2$.
De même, pour `Respart_aa`, en utilisant la symétrie de B^-1 et de P, on peut optimiser le coûte mémoire à $0.5 * Q_a^2 \cdot N^2$.

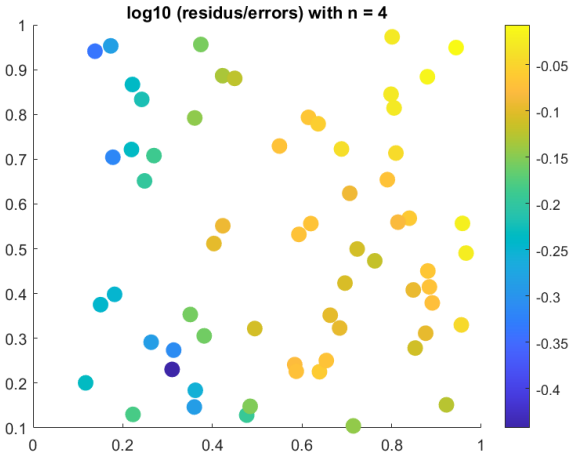


FIGURE 2.1 – $\log_{10} \frac{\texttt{résidus}}{\texttt{errors}}$ avec $n = 4$

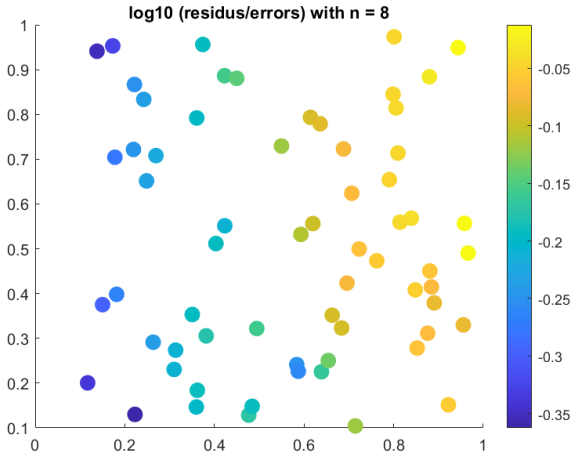


FIGURE 2.2 – $\log_{10} \frac{\texttt{résidus}}{\texttt{errors}}$ avec $n = 8$

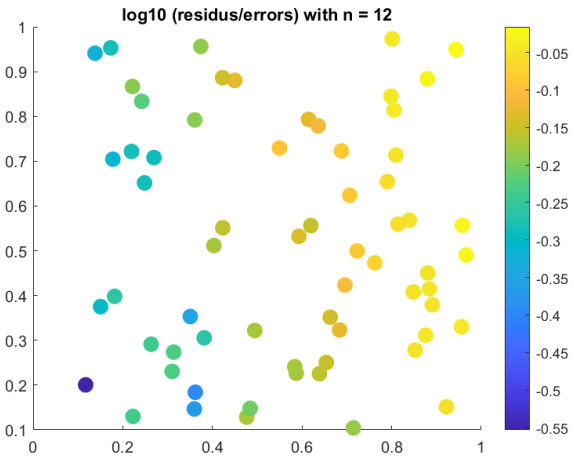


FIGURE 2.3 – $\log_{10} \frac{\texttt{résidus}}{\texttt{errors}}$ avec $n = 12$

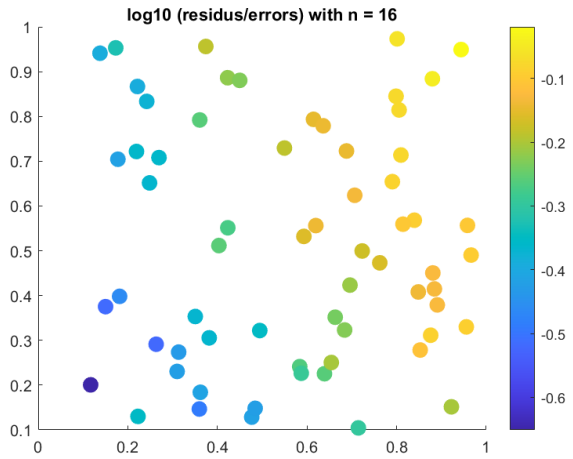
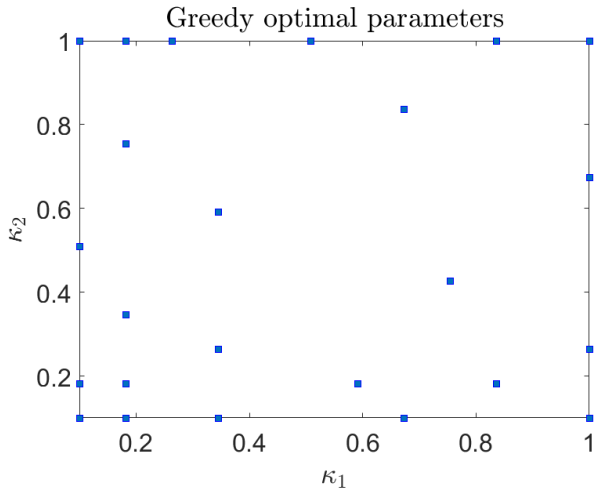


FIGURE 2.4 – $\log_{10} \frac{\texttt{résidus}}{\texttt{errors}}$ avec $n = 16$

Les images FIGURE 2.1, 2.2, 2.3et 2.4 sont le rapport `résidus/errors` lors de nombre de POD $N = 4, 8, 12, 16$.

On souhaite que ce rapport soit proche de 1 (sa valeur logarithmique proche de 0). On a observé que la plus part de points sont entre 0 et -0.15 pour $n = 4, 8$. Pour $n = 12$, environ la moitié de points sont entre 0 et -0.15. Pour $n = 16$, il reste environ moins d'un tiers de points qui sont entre 0 et -0.15. Cela implique que la qualité de l'algorithme de résidu devient de moins en moins bien. Ce problème est dû à la précision numérique de cet algorithme qui n'est pas suffisamment précise.

2.2 L'algorithme Greedy



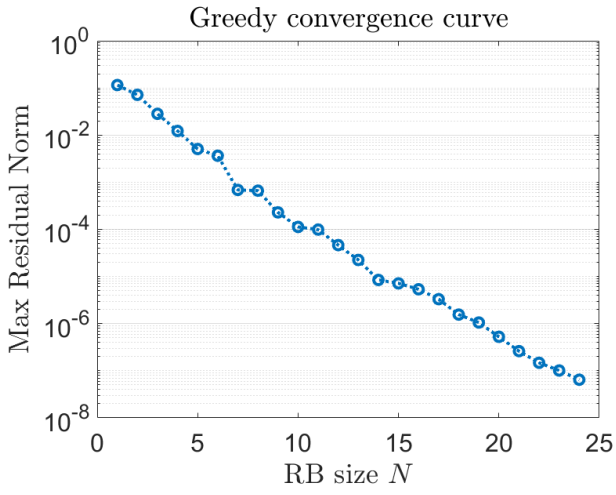
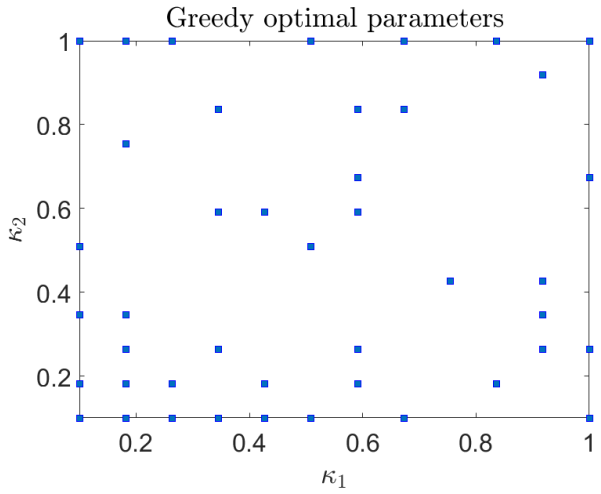


FIGURE 2.5 – algorithme classique avec `tol` $= 10e-7$ FIGURE 2.6 – algorithme classique avec `tol` $= 10e-7$



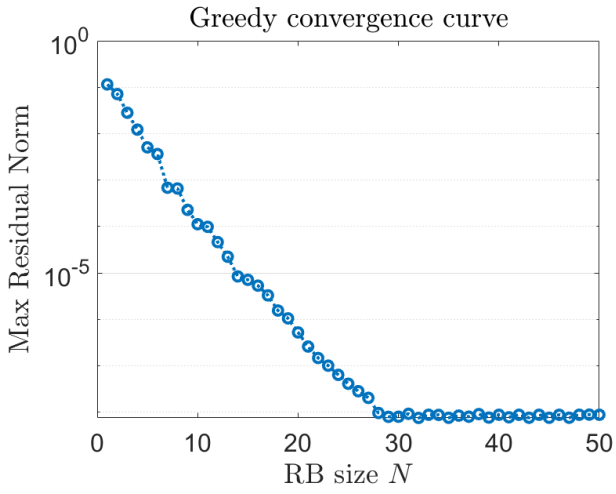
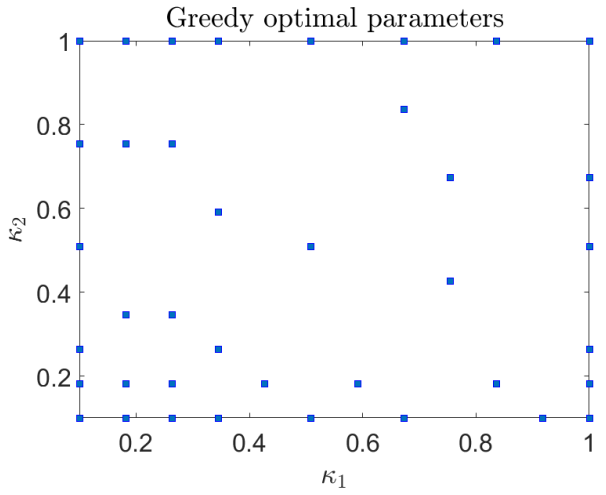


FIGURE 2.7 – algorithme classique avec `tol` $= 10e-10$ FIGURE 2.8 – algorithme classique avec `tol` $= 10e-10$

Les images FIGURE 2.5 et 2.6 sont le résultats pour la méthode classique avec une tolérance à $10e-7$. On peut observer que la méthode fonctionne bien, le résidu décroît très vite.

Les images FIGURE 2.7et 2.8 sont le résultats pour la méthode classique avec une tolérance à $10e-10$. On peut observer que la méthode ne fonctionne plus, le résidu décroît très vite jusqu'à $10e-8$ et qui se stabilise. Donc avec la méthode classique, on ne peut pas avoir une précision de niveau $10e-10$.

De plus, le nombre de résidu négatif peut également expliquer le problème de précision numérique. Pour tolérance à $10e-7$, tous les résidus sont positifs. Mais pour tolérance à $10e-10$, par exemple pour itération 50, il y a 20 résidus qui sont strictement négatifs. Or le résidu devrait être positif, on a trouvé alors le problème de précision numérique.



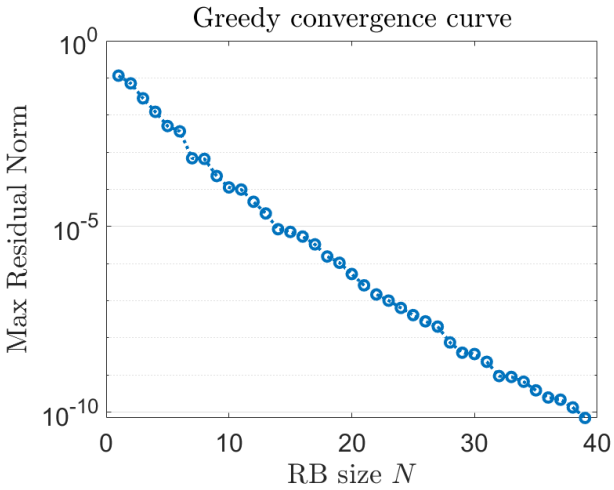
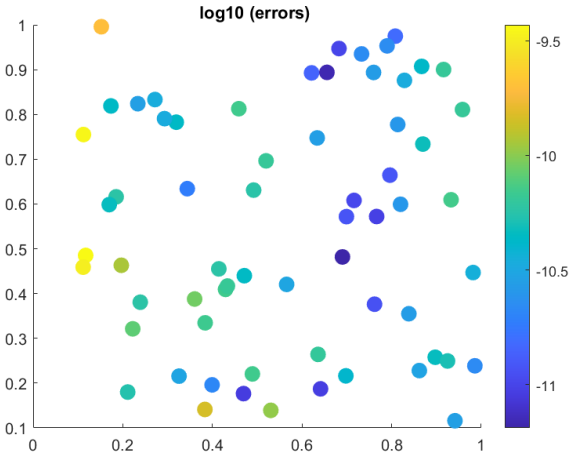


FIGURE 2.9 – algorithme stabilisé avec `tol` $= 10e - 10$ FIGURE 2.10 – algorithme stabilisé avec `tol` $= 10e - 10$
Les images FIGURE 2.9 et 2.10 sont le résultats pour la méthode stabilisé avec une tolérance à $10e - 10$. On peut observer que la méthode fonctionne bien, le résidu décroît très vite.

De plus, on a observé aussi que tous les résidus sont positifs. Cela implique que la méthode stabilisée améliore le problème de précision numérique.

2.3 Certification



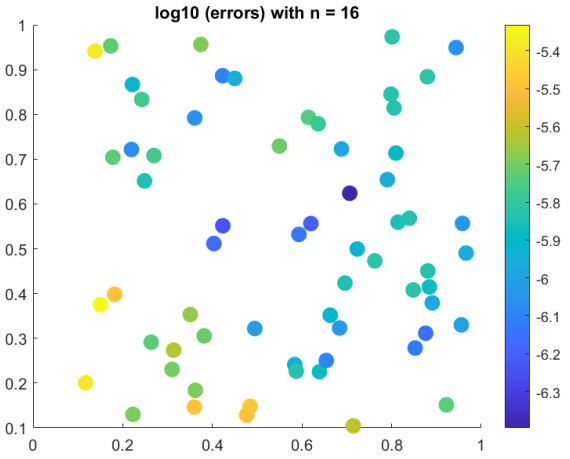


FIGURE 2.11 – errors avec RB Greedy

FIGURE 2.12 – errors avec RB POD $N = 16$

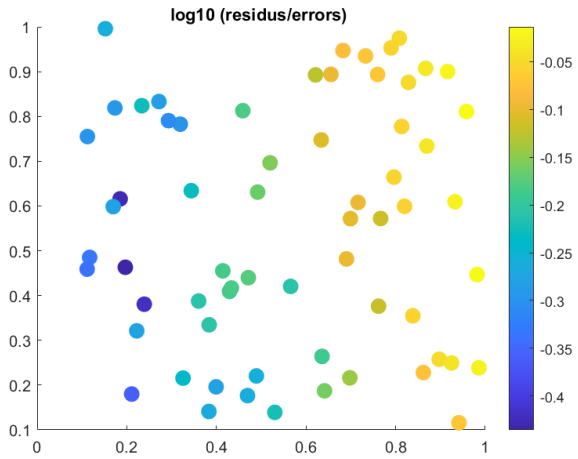


FIGURE 2.13 – $\log_{10} \frac{\texttt{résidus}}{\texttt{errors}}$ avec RB Greedy

Les images FIGURE 2.11 et 2.13 sont le résultats pour la base réduite de Greedy avec la méthode stabilisé avec une tolérance à $10e-10$. En comparant avec FIGURE 2.12, on peut observer que la précision pour la base réduite de Greedy est beacoup plus précise que celle pour la base POD lors de $N = 16$.

De plus, on a observé aussi que le rapport `residus/errors` pour la base réduite de Greedy avec la méthode stabilisé a une bonne qualité. Il y a environ plus de moitié de points qui sont entre 0 et -0.15.

Chapitre 3

Application

3.1 Offline et Online

Pour la phase `Offline`, On a $j^{\mathtt{rb}} = P^T \cdot j$.
Pour la phase `Online`, on a $Q^{\mathtt{rb}}(\mu) = {j^{\mathtt{rb}}}^T \cdot x(\mu)$.
La compléxité pour obtenir $x(\mu)$ est $\mathcal{O}(N^3)$.
La compléxité pour le produit ${j^{\mathtt{rb}}}^T \cdot x(\mu)$ est $\mathcal{O}(N)$.

3.2 Estimation à posteriori

Pour obtenir estimation $|Q^{\mathtt{rb}}(\mu) - Q^{\mathcal{N}}(\mu)| \leq \delta_N^Q(\mu)$, on a $|Q^{\mathtt{rb}}(\mu) - Q^{\mathcal{N}}(\mu)| = |j^T \cdot \left(P \cdot x(\mu) - u^{\mathcal{N}}(\mu)\right)|$.
Comme on a $\|j\| = \sup_{\|v\| \neq 0} \frac{|j^T \cdot v|}{\|v\|}$, on a alors $|j^T \cdot \left(P \cdot x(\mu) - u^{\mathcal{N}}(\mu)\right)| \leq \|j\| \cdot \left\|\left(P \cdot x(\mu) - u^{\mathcal{N}}(\mu)\right)\right\|$.

De plus, on a déjà $\left\|\left(P \cdot x(\mu) - u^{\mathcal{N}}(\mu)\right)\right\| \leq \Delta_N(\mu)$, où $\Delta_N(\mu)$ est l'estimation à posteriori par le résidu avec un algorithme stabilisé.

Finalement, on peut poser $\delta_N^Q(\mu) = \|j\| \cdot \Delta_N(\mu)$.

3.3 Résultats

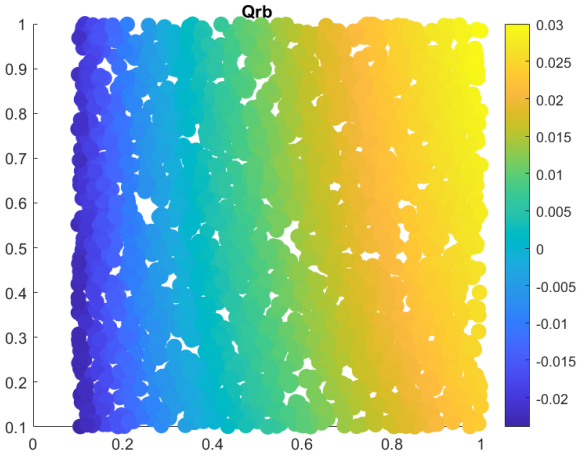


FIGURE 3.1 – $Q^{\mathtt{rb}}(\mu)$ pour 3000 points

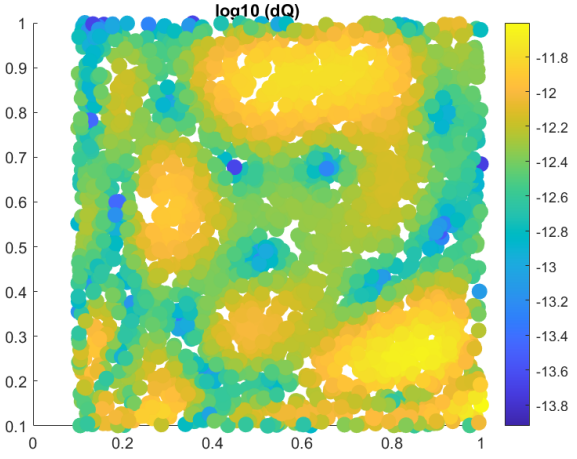


FIGURE 3.2 – $\delta_N^Q(\mu)$ pour 3000 points

Les images FIGURE 3.1 et 3.2 sont le résultats pour $Q^{\mathtt{rb}}(\mu)$ et $\delta_N^Q(\mu)$.

Selon la valeur de $\delta_N^Q(\mu)$, on peut dire qu'on a une très bonne précision pour $Q^{\mathtt{rb}}(\mu)$.

De plus, pour le temps de calcul, le temps utilisé total est 0.5251s. Le temps pour calculer les 3000 différents $Q^{\mathtt{rb}}(\mu)$ est 0.2280s. Le temps pour calculer l'estimation à posteriori est 0.2971s. Grâce à la méthode de base réduite, on a vraiment gagné beaucoup de temps de calculs et obtenir une très bonne précision (environ $10e-11$).