Introduction à la Recherche en Laboratoire
Méthode de bases réduites

Simon Clément

Printemps 2018

Introduction

Dans les domaines de la météorologie, de l'océanographie et du climat, de nombreuses applications requièrent l'utilisation conjointe d'un modèle atmosphérique et d'un modèle océanique. C'est le cas par exemple pour la prévision du climat, ou encore pour la simulation de cyclones.
L'équipe AIRSEA (Inria - LJK) développe des méthodes numériques avancées pour ces modèles. Elle étudie notamment les méthodes de couplage, afin d'améliorer les techniques actuelles et de représenter au mieux les échanges à l'interface air-mer.
Dans le cadre des modèles océan-atmosphère, on est amené à résoudre, en chaque point de la grille de calcul horizontale et à chaque pas de temps, une équation de diffusion sur la verticale. Le fait de coupler les modèles induit de plus une méthode itérative (algorithme de Schwarz), qui va multiplier le nombre de résolutions.
Afin de diminuer les coûts de calcul, une piste pourrait être d'utiliser une méthode de base réduite pour la résolution de chaque équation de diffusion.

Ce projet d'introduction à la recherche en laboratoire a ainsi pour objectif d'apporter un premier aperçu de cette méthode et de voir comment elle pourrait accélérer les calculs. On étudie donc une équation de diffusion avec une diffusivité variable. En résolvant (numériquement) cette équation par différences finies et par la méthode de bases réduites, on compare les résultats en termes d'erreur sur la solution et de temps de calcul.
La méthode de base réduite consiste à restreindre l'espace des solutions du schéma numérique à un espace plus petit. Le choix de cet espace est bien sûr très important et constitue une part importante du travail : il a été choisi à l'aide d'un ensemble d'apprentissage, constitué de plusieurs jeux de paramètres différents pour la résolution de l'équation de diffusion.

Remerciements

Je remercie le LJK de m'avoir accueilli dans ses resplendissants nouveaux locaux.
Merci également à M. Idani, pour m'avoir donné envie de m'engager dans ce module IRL.
Je remercie surtout l'équipe Airsea, et plus particulièrement Charles Pelletier pour les réponses rapides et complètes à mes requêtes, Sophie Thery pour m'avoir accompagné durant tout le projet, et enfin Eric Blayo, pour tout le temps et l'énergie qu'il m'a consacré.

Table des matières

1 Présentation du problème

Dans le cadre de la modélisation des échanges océan-atmosphère, on est amené à résoudre un très grand nombre d'équations de diffusion en une dimension avec des coefficients variables en espace :

$$\frac{\partial u}{\partial t} - \frac{\partial}{\partial z}\left(\nu(z)\frac{\partial u}{\partial z}\right) = 0,\ z \in [0, H[,\ t > 0 \tag{1}$$

$$\frac{\partial u}{\partial z}(0,t) = a(t),\ u(H,t) = b,\ t > 0 \tag{2}$$

$$u(z,0) = u_0(z) \tag{3}$$

avec ν, de la forme :

$$\nu(z) = Cz\left(1 - \frac{z}{h_{CL}}\right)^2 \tag{4}$$

La valeur $z = 0$ correspond à l'interface air-mer et $z = H$ correspond au fond de l'océan ou au sommet de l'atmosphère. Les différentes équations à résoudre diffèrent par les conditions aux limites a et b, la condition initiale $u_0(z)$, ainsi que les paramètres h_{CL} et C de la fonction $\nu(z)$.

Après la discrétisation en espace de l'équation de diffusion, le problème devient pour $u(t) \in \mathbb{R}^N$:

$$\begin{aligned} M\dot{u} + Ku &= f \\ u(0) &= u_0 \end{aligned} \tag{5}$$

où M est la matrice égale à Id_N sauf pour $M_{N,N} = M_{1,1} = 0$, pour intégrer les conditions aux bords (ici de type Dirichlet-Neumann). On aura également $f_0 = a$, $f_N = b$

2 Présentation de la méthode de bases réduites

Les méthodes de base réduites ont été développées depuis une vingtaine d'années, notamment à l'initiative de mathématiciens du MIT. Les résultats sur la méthode de bases réduites présentés dans ce document proviennent de [2].

Pour réduire le temps de calcul engendré par la résolution de ces équations, on va chercher à représenter l'espace des solutions au temps t par un espace de dimension $m \ll N$, engendré par une base $(\phi_1, ...\phi_m)$.

On notera $\Phi \in \mathbb{R}^{N\times m}$ la matrice de projection dont chaque colonne i est le vecteur ϕ_i. Ainsi, une fonction de cet espace sera de la forme

$$\widehat{u}(t) = \Phi\alpha(t) \tag{6}$$

avec $\alpha(t) \in \mathbb{R}^m$.

En utilisant (6), on peut reformuler le problème (5) en $M\Phi\dot{\alpha} + K\Phi\alpha = f$, d'où

$$\begin{aligned} \Phi^T M\Phi\dot{\alpha} + \Phi^T K\Phi\alpha &= \Phi^T f \\ \Phi^T M\Phi\alpha(0) &= \Phi^T M u_0 \end{aligned} \tag{7}$$

En utilisant par exemple une discrétisation d'Euler implicite, le système (7) est rapidement solvable : les matrices sont de taille $m \times m$. Le choix de la base sera détaillé dans la section suivante.

3 Choix de la base Φ

3.1 Introduction

L'ensemble engendré par Φ doit représenter l'ensemble des solutions de nos équations tout en permettant une résolution du système (7) rapide. Il faut donc choisir les vecteurs ϕ_i de manière à avoir une bonne précision, avec m le plus petit possible.

On dispose pour cela d'un ensemble d'apprentissage de taille S : on a résolu numériquement l'équation S fois, en faisant varier les paramètres d'entrée. Pour le $k^{\text{ième}}$ élément de l'ensemble d'apprentissage, on note u^k la solution numérique "exacte" (solution de (5)), et on note $\widehat{u}^k = \Phi\alpha^k$ la solution approchée correspondante (solution de (7), avec la base Φ). De même, la matrice K associée au $k^{ième}$ élément de l'ensemble d'apprentissage sera notée K^k. Notre démarche sera de minimiser une fonction coût qui quantifie l'écart entre ces solutions numériques exactes et les solutions numériques approchées par base réduite. L'optimisation sera faite selon une descente de gradient (BFGS), sur le Lagrangien et son gradient que nous expliciterons.

3.2 Fonction de minimisation

Pour trouver cette base Φ, on résout le problème d'optimisation suivant :

$$\begin{aligned}\min_{\Phi} \mathcal{G} =& \frac{1}{2}\sum_{k=1}^{S}\int_0^{t_f}(u^k-\widehat{u}^k)^T(u^k-\widehat{u}^k)dt \\ &+\frac{\beta}{2}\sum_{j=1}^{m}(1-\phi_j^T\phi_j)^2\end{aligned} \tag{8}$$

avec les contraintes suivantes pour $k = 1...S$:

$$\begin{aligned}\Phi^T M\Phi\dot{\alpha}^k + \Phi^T K^k\Phi\alpha^k &= \Phi^T f \\ \Phi^T M\Phi\alpha^k(0) &= \Phi^T M u_0^k\end{aligned} \tag{9}$$

Le premier terme de la fonction coût $\mathcal{G}$ permet de cibler une base qui minimise l'erreur quadratique moyenne lorsqu'on l'utilise sur les échantillons. Le second terme est un terme de régularisation qui permet de s'assurer que les vecteurs sont presque normalisés. En effet, selon [2], la sortie u est peu affectée par ce terme pour des valeurs de β suffisamment petites, mais le conditionnement du problème d'optimisation est meilleur.

Dans le cas où la quantité d'intérêt du problème n'est pas u, mais $g = Cu$, $C \in \mathbb{R}^{Q\times N}$, $g(t) \in \mathbb{R}^Q$ on peut "pondérer" la minimisation de l'erreur quadratique moyenne par la matrice $H^k = {C^k}^T C^k$. On obtient ainsi la minimisation suivante :

$$\begin{aligned}\min_{\Phi} \mathcal{G} =& \frac{1}{2}\sum_{k=1}^{S}\int_0^{t_f}(u^k-\widehat{u}^k)^T H^k(u^k-\widehat{u}^k)dt \\ &+\frac{\beta}{2}\sum_{j=1}^{m}(1-\phi_j^T\phi_j)^2\end{aligned} \tag{10}$$

avec les contraintes pour $k = 1...S$:

$$\begin{aligned} \Phi^T M \Phi \dot{\alpha}^k + \Phi^T K^k \Phi \alpha^k &= \Phi^T f \\ \Phi^T M \Phi \alpha^k(0) &= \Phi^T M u_0^k \end{aligned} \tag{11}$$

Dans la suite, on gardera cette matrice H^k, pour plus de généralité, quitte à la remplacer par la matrice identité à la fin. Si la discrétisation spatiale utilisée n'est pas uniforme, cette matrice peut permettre d'ajuster le produit $(u^k - \widehat{u}^k)^T H^k (u^k - \widehat{u}^k)$ pour avoir une approximation de $||u^k - \widehat{u}^k||_2^2$

Ce problème d'optimisation n'est ni linéaire ni convexe, on n'a donc aucun résultat sur la convergence vers un minimum global : c'est pourquoi il est important d'avoir un premier itéré assez bon pour démarrer l'algorithme d'optimisation. On détaillera dans une section suivante le choix de ce premier itéré.

3.3 La descente de gradient

3.3.1 Lagrangien du problème d'optimisation

Le problème (10)-(11) mène au Lagrangien suivant :

$$\begin{aligned} \mathcal{L}(\Phi, \alpha^k; \lambda^k, \mu^k) =& \frac{1}{2} \sum_{k=1}^{S} \int_0^{t_f} (u^k - \widehat{u}^k)^T H^k (u^k - \widehat{u}^k) dt \\ &+ \frac{\beta}{2} \sum_{j=1}^{m} (1 - \phi_j^T \phi_j)^2 \\ &+ \sum_{k=1}^{S} \int_0^{t_f} {\lambda^k}^T \left(\Phi^T M \Phi \dot{\alpha}^k + \Phi^T K^k \Phi \alpha^k - \Phi^T f \right) dt \\ &+ \sum_{k=1}^{S} {\mu^k}^T \left(\Phi^T M \Phi \alpha^k(0) - \Phi^T M u_0^k \right) \end{aligned} \tag{12}$$

où $\lambda^k = \lambda^k(t) \in \mathbb{R}^m$ et $\mu^k \in \mathbb{R}^m$ sont des multiplicateurs de Lagrange qui proviennent des contraintes (11).

3.3.2 Equations déduites du Lagrangien

En posant $\nabla_{\alpha^k} \mathcal{L} = 0$, on trouve les contraintes suivantes pour $k = 1, ..., S$ (on suppose M symétrique) :

$$-\Phi^T M \Phi \dot{\lambda^k} + {K^k}^T \Phi \lambda^k = \Phi^T H^k (u^k - \Phi \alpha^k) \tag{13}$$

$$\lambda^k(t_f) = 0 \tag{14}$$

$$\mu^k = \lambda^k(0) \tag{15}$$

La démonstration est donnée dans l'Annexe en fin de document.
Le gradient du Lagrangien selon les vecteurs de la base Φ donne l'équation matricielle sui-

vante :

$$\begin{aligned}\nabla_\Phi \mathcal{L} =& \sum_{k=1}^{S} \int_0^{t_f} \left(H^k \Phi \alpha^k \alpha^{k^T} - H^k u^k \alpha^{k^T} \right) dt \\ &+ 2\beta \Phi \mathrm{diag}(1 - \phi_i^T \phi_i) \\ &+ \sum_{k=1}^{S} \int_0^{t_f} \left[M\Phi(\lambda^k \dot{\alpha}^{kT} + \dot{\alpha}^k \lambda^{k^T}) + K^{k^T} \Phi \lambda^k \alpha^{k^T} + K^k \Phi \alpha^k \lambda^{k^T} - f^k \lambda^{k^T} \right] dt \\ &- \sum_{k=1}^{S} M u_0^k \mu^{k^T} = 0\end{aligned} \tag{16}$$

3.3.3 Algorithme de résolution de l'optimisation

Pour résoudre le problème d'optimisation (10) - (11), on utilisera une méthode de descente de gradient, qui requiert à chaque itération des évaluations de la fonction et de son gradient. Dans ce cas, le gradient de l'objectif $\mathcal{L}$ selon Φ est donné par $\nabla_\Phi \mathcal{L}$.
Ainsi, l'algorithme de calcul du Lagrangien et de son gradient est le suivant :

Entrée : base Φ
- Résoudre (11) pour obtenir les α^k
- Résoudre (13)- (14) pour obtenir les λ^k
- Utiliser (15) pour obtenir μ^k

Sortie : $\mathcal{L}(\Phi, \alpha^k, \lambda^k, \mu^k)$ et $\nabla_\Phi \mathcal{L}(\Phi, \alpha^k, \lambda^k, \mu^k)$ (utiliser (12) et (16).)

3.3.4 First guess de Φ

Le problème de minimisation étant non-linéaire et non-convexe, il est important de bien choisir la base Φ dont on part pour la descente de gradient. Deux stratégies sont possibles : la première est d'utiliser en point de départ la base POD ("Proper Orthogonal Decomposition") (encore appelée ACP - Analyse en Composantes Principales - en vocabulaire statistique), c'est à dire la base réduite de taille m maximisant la variance conservée parmi l'ensemble d'apprentissage. Une autre stratégie est de partir de la solution au problème d'optimisation pour $m-1$ vecteurs, et d'ajouter arbitrairement un $m^{\text{ième}}$ vecteur. On initialise l'algorithme à n'importe quelle valeur $m \geq 1$ avec la base POD. La décomposition POD est beaucoup plus rapide que l'optimisation et donne déjà une bonne approximation de l'ensemble des solutions, donc la première stratégie est très efficace. Dans les expériences réalisées, la deuxième stratégie a très mal fonctionné et ne donnait pas de résultat satisfaisant.
Nous avons utilisé l'excellente bibliothèque python "modred" pour la décomposition POD.

4 Discrétisation temporelle

Dans cette partie, on va appliquer une discrétisation d'Euler implicite pour pouvoir résoudre le problème.

4.1 Résolution de l'équation

Lorsqu'on a au temps t_n une solution $\widehat{u}_n$, $n \in [\![0, N-1]\!]$ on peut trouver la solution approchée au temps $\widehat{u}_{n+1}$ avec le système suivant qu'on déduit de (7) :

$$\begin{aligned} \Phi^T M \Phi \alpha_n &= \Phi^T M \widehat{u}_n \\ (\frac{1}{\delta t}\Phi^T M \Phi + \Phi^T K \Phi)\alpha_{n+1} &= \frac{1}{\delta t}\Phi^T M \Phi \alpha_n + \Phi^T f_{n+1} \\ \widehat{u}_{n+1} &= \Phi \alpha_{n+1} \end{aligned} \tag{17}$$

avec K la matrice obtenue en appliquant la discrétisation spatiale (voir (5)).

4.2 Recherche de la base optimale

On modifie notre problème d'optimisation [1] :

$$\begin{aligned} \min_{\Phi} \mathcal{G} =& \frac{1}{2}\sum_{k=1}^{S}\sum_{n=1}^{N-1}(u_n^k - \widehat{u}_n^k)^T H^k (u_n^k - \widehat{u}_n^k) \\ &+ \frac{\beta}{2}\sum_{j=1}^{m}(1 - \phi_j^T \phi_j)^2 \end{aligned} \tag{18}$$

avec les contraintes pour $k = 1...S$, $n = 1...N-1$:

$$\begin{aligned} (\frac{1}{\delta t}\Phi^T M \Phi + \Phi^T K \Phi)\alpha_{n+1} &= \frac{1}{\delta t}\Phi^T M \Phi \alpha_n + \Phi^T f_{n+1} \\ \Phi^T M \Phi \alpha_0^k &= \Phi^T M u_0^k \end{aligned} \tag{19}$$

Le Lagrangien (12) devient ainsi :

$$\begin{aligned} \mathcal{L}(\Phi, \alpha^k, \lambda^k, \mu^k) =& \frac{1}{2}\sum_{k=1}^{S}\sum_{n=1}^{N-1}(u_n^k - \widehat{u}_n^k)^T H^k (u_n^k - \widehat{u}_n^k) \\ &+ \frac{\beta}{2}\sum_{j=1}^{m}(1 - \phi_j^T \phi_j)^2 \\ &+ \sum_{k=1}^{S}\sum_{n=0}^{N-1}\lambda_n^{kT}((\frac{1}{\delta t}\Phi^T M \Phi + \Phi^T K^k \Phi)\alpha_{n+1}^k - \frac{1}{\delta t}\Phi^T M \Phi \alpha_n^k - \Phi^T f_{n+1}) \\ &+ \sum_{k=1}^{S}\mu^{kT}\left(\Phi^T M \Phi \alpha_0^k - \Phi^T M u_0^k\right) \end{aligned} \tag{20}$$

1. On va ici de 1 à $N-1$ pour simplifier le calcul du gradient. Si les échantillons ne sont pas observés sur l'ensemble de la fenêtre temporelle, il faudrait penser à minimiser l'erreur sur l'intégralité des pas de temps.

En annulant ses dérivées, on obtient les contraintes (voir Annexe pour la démonstration) :

$$\begin{aligned}(\Phi^T M\Phi + \delta t\Phi^T K^{k^T}\Phi)\lambda_{n-1}^k &= \Phi^T M\Phi\lambda_n^k + \delta t\Phi^T H^k(u_n^k - \Phi\alpha_n^k),\ n = 1...N-1\\ \lambda_{N-1}^k &= 0\\ \mu_k &= \frac{\lambda_0^k}{\delta t}\end{aligned} \tag{21}$$

4.2.1 Calcul du gradient

Le gradient du Lagrangien (20) est donné par l'expression suivante :

$$\begin{aligned}\nabla_\Phi\mathcal{L} = &\sum_{k=1}^{S}\sum_{n=1}^{N-1}\left[H^k\Phi\alpha_n^k\alpha_n^{k^T} - H^k u_n^k\alpha_n^{k^T}\right]\\ &- 2\beta\Phi\, diag(1-\phi_i^T\phi_i)\\ &+\sum_{k=1}^{S}\sum_{n=0}^{N-1}\frac{M\Phi}{\delta t}\left(\lambda_n^k(\alpha_{n+1}^k-\alpha_n^k)^T + (\alpha_{n+1}^k-\alpha_n^k)\lambda_n^{k^T}\right)\\ &\qquad + K^k\Phi\alpha_{n+1}^k\lambda_n^{k^T} + K^{kT}\Phi\lambda_n^k\alpha_{n+1}^{k^T} - f_{n+1}\lambda_n^{k^T}\\ &+\sum_{k=1}^{S}\left(M\Phi(\alpha_0^k\mu^{k^T} + \mu^k\alpha_0^{k^T}) - Mu_0^k\mu^{k^T}\right)\end{aligned} \tag{22}$$

Pour calculer ce gradient, on commence par résoudre le système (19). Une fois les α^k obtenus, on résout (21) pour obtenir les λ^k et les μ^k. On a ensuite toutes les valeurs requises pour calculer $\nabla_\Phi\mathcal{L}$.

4.2.2 Résultats du BFGS

En lançant l'algorithme d'optimisation du BFGS sur $\mathcal{G}(\Phi)$ avec $\nabla_\Phi\mathcal{L}$ et la base de la POD en premier itéré, les résultats obtenus sont très bons : la valeur de la fonction objectif obtenue est en effet bien plus petite (entre 40 et 60% de diminution par rapport au premier itéré).
Le calcul de $\nabla_\Phi\mathcal{L}$ a été testé sur la base POD avec plusieurs ensembles d'apprentissage différents en le comparant avec le gradient qu'on peut obtenir en calculant les taux d'accroissement.
Cette optimisation est donc utile pour réduire l'erreur sur les solutions. Résoudre ce problème a un certain coût : l'évaluation de $\mathcal{G}$ et de $\nabla_\Phi\mathcal{L}$ prend un temps proportionnel au nombre d'échantillon dans l'ensemble d'apprentissage. Mais cette optimisation est à faire une seule fois (calcul *offline*) et permet de pouvoir réduire la taille m de la base réduite, réduisant ainsi le temps de calcul *online*.

5 Calcul direct

Dans cette partie, on va donner la méthode de calcul direct qui sera utilisée afin d'avoir un point de comparaison pour la méthode de bases réduites.

5.1 Différences finies

On va résoudre (1) à l'aide de différences finies. On notera $(z_0, ..., z_N)$ les hauteurs considérées (c'est à dire la grille de discrétisation sur la verticale), $(u_0, ...u_N)$ les approximations de $(u(z_0), ..., u(z_N))$, $(\nu_{\frac{1}{2}}, ..., \nu_{j+\frac{1}{2}}, ..., \nu_{N-\frac{1}{2}})$ les approximations de ν en $(\frac{z_0+z_1}{2}, ..., \frac{z_j+z_{j+1}}{2}, ..., \frac{z_{N-1}+z_N}{2})$ et $\forall j, h_j = z_{j+1} - zj$. On pose :

$$\frac{\partial u}{\partial z}(z_{j+\frac{1}{2}}) \approx \frac{u_{j+1} - u_j}{h_j}$$
$$\frac{\partial}{\partial z}\left(\nu(z_j)\frac{\partial u}{\partial z}(z_j)\right) \approx \frac{\nu_{j+\frac{1}{2}}\frac{\partial u}{\partial z}(z_{j+\frac{1}{2}}) - \nu_{j-\frac{1}{2}}\frac{\partial u}{\partial z}(z_{j-\frac{1}{2}})}{\frac{1}{2}(h_j + h_{j-1})}$$

Le système d'équations à résoudre est donc le suivant :

$$\frac{u_j^{n+1} - u_j^n}{\delta t} - 2\frac{h_{j-1}\nu_{j+\frac{1}{2}}u_{j+1}^{n+1} + h_j\nu_{j-\frac{1}{2}}u_{j-1}^{n+1} - (h_j\nu_{j-\frac{1}{2}} + h_{j-1}\nu_{j+\frac{1}{2}})u_j^{n+1}}{(h_j + h_{j-1})h_j h_{j-1}} = f^{n+1}(z_j)$$
$$\frac{u_1^{n+1} - u_0^{n+1}}{h_0} = a^{n+1}$$
$$u_N^{n+1} = b^{n+1}$$

On reformule la première équation pour trouver une forme matricielle :

$$\begin{aligned}\forall j \in [\![1, N-1]\!], f^{n+1}(z_j) - \frac{u_j^{n+1} - u_j^n}{\delta t} =& u_{j+1}^{n+1}\left(-\frac{2\nu_{j+\frac{1}{2}}}{h_j(h_j + h_{j-1})}\right) \\ &+ u_j^{n+1}\left(\frac{2h_{j-1}\nu_{j+\frac{1}{2}} + 2h_j\nu_{j-\frac{1}{2}}}{h_j h_{j-1}(h_j + h_{j-1})}\right) \\ &+ u_{j-1}^{n+1}\left(-\frac{2\nu_{j-\frac{1}{2}}}{h_{j-1}(h_j + h_{j-1})}\right) \\ &= u_{j+1}^{n+1}K_{j+1,j+1} + u_j^{n+1}K_{j+1,j} + u_{j-1}^{n+1}K_{j+1,j-1}\end{aligned}$$

avec K la matrice tri-diagonale satisfaisant cette égalité, et telle que $K_{1,1} = \frac{-1}{h_0}$, $K_{1,2} = \frac{1}{h_0}$, $K_{N+1,N+1} = 1$ et $K_{N+1,N} = 0$. On a donc bien la relation

$$M\frac{u_j^{n+1} - u_j^n}{\delta t} + Ku^{n+1} = f^{n+1}$$

avec $f^{n+1}(z_0) = a^{n+1}, f^{n+1}(z_N) = b^{n+1}$

6 Expériences numériques

6.1 Conditions des expériences

Les données utilisées pour l'ensemble d'apprentissage et pour l'ensemble de tests sont tirées d'une simulation de LMDZ (modèle de circulation générale atmosphérique) de 20 jours sur une colonne d'air avec un pas de temps de 10 minutes. On crée donc la base d'apprentissage en résolvant l'équation de diffusion par le shéma présenté en partie 5 pour chaque situation issue de LMDZ. L'ensemble d'apprentissage obtenu est de taille $S = 30$. On fait de même pour l'ensemble test, à partir de situations de LMDZ différentes (les 10 premiers jours de la simulation servent pour l'apprentissage, les autres pour la base de test).

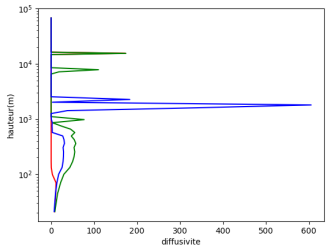
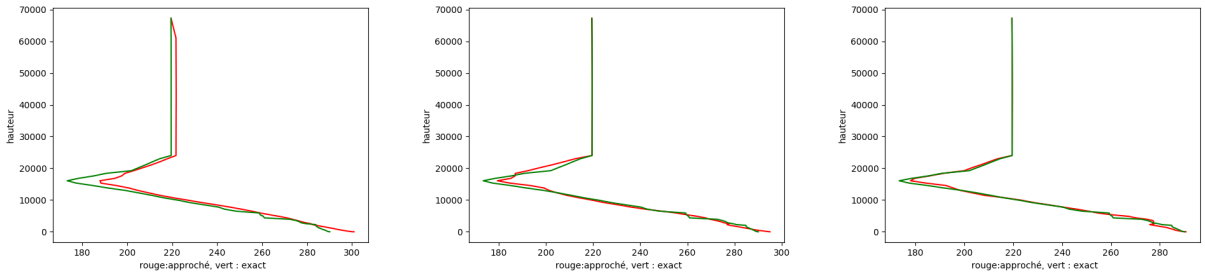


FIGURE 1 – Exemples de profils de diffusivité thermique. En abscisse : diffusivité (en $m^2 s^{-1}$) ; en ordonnée : hauteur z. Les 3 courbes de couleur correspondent à 3 cas différents dans la base d'apprentissage. Les pics qui apparaissent sont dûs aux calculs complexes de LMDZ.

Pour chaque pas de temps, on a :
— La discrétisation verticale
— La diffusivité thermique ν (voir figure 1)
— La température

6.2 Erreur commise par l'approximation "base réduite"

On met maintenant en oeuvre la méthode de bases réduites et on compare dans l'ensemble de test (indépendant de l'ensemble d'apprentissage) les solutions numériques exactes et approchées. Même dans les pires cas, l'erreur est assez faible et le profil de température trouvé avec la méthode de bases réduites est proche de celui trouvé avec les différences finies (voir figure 6.2). Sur la figure 3, on peut voir que la répartition de l'erreur L2 est centrée en 2^oC, avec un écart-type de l'ordre de 1^oC pour $m = 3$. L'absence de valeurs extrêmes dans les erreurs provient de l'optimisation de $\mathcal{G}$, qui est une somme des normes L2 (à la multiplication par la matrice H près). Cette optimisation tend à homogénéiser les erreurs et éviter les extrêmes. On peut quand même noter un moins bon résultat pour la base optimisée dans le pire cas (voir la légende de la figure 6.2).
En augmentant la taille de la base, l'erreur moyenne se déplace vers 0^o. Ces erreurs sont toutefois à prendre avec des précautions. Tout d'abord, la norme L2 est assez trompeuse puisque la température est constante sur la moitié de l'intervalle de z. De plus, les données étaient relativement homogènes et les ensembles d'apprentissages et de tests étaient assez proches. Avant d'appliquer la méthode systématiquement, il faudrait donc faire ces tests



Pires cas de l'ensemble test

FIGURE 2 – En vert, la solution (en ^oK) des différences finies la moins bien approchée par l'approximation de la base réduite (au sens de la norme L2). En rouge, son approximation avec la méthode des bases réduites. De gauche à droite, les bases réduites sont de taille 1, 3 et 7. Le fait que la solution soit constante à partir d'une certaine altitude correspond à une approximation faite par le modèle LMDZ.
La base Φ utilisée ici est la POD. Lorsqu'on applique ici l'optimisation pour trouver la "meilleure" base, le pire cas est bien moins bon que si on avait pris la POD. On a un problème de sur-apprentissage, qu'on pourrait corriger avec un terme de régularisation.

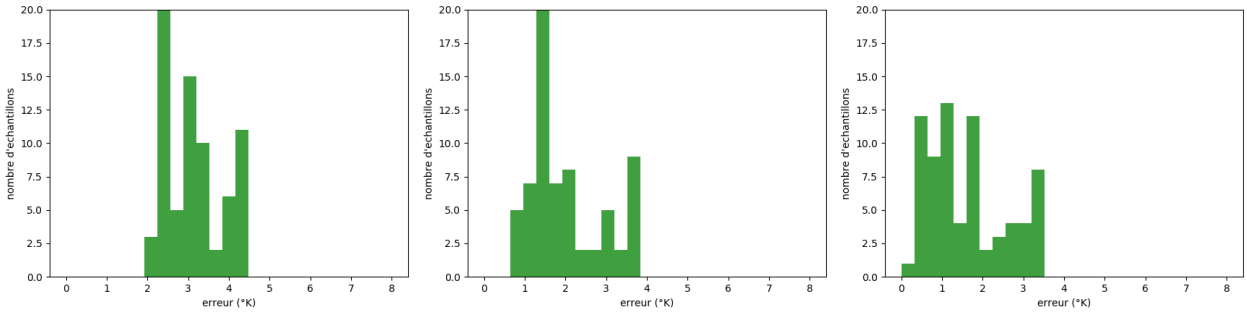


FIGURE 3 – Histogramme de l'erreur L2 (entre 0 et 8^oC) dans un échantillon de 72 tests pour une base de taille $m = 1$, $m = 3$ et $m = 5$ (de gauche à droite)

d'erreur sur des jeux de paramètres plus variés qu'une simulation continue de 20 jours à un lieu fixé.

6.3 Temps de calcul

Le temps de calcul est un élément très important dans la décision d'utiliser la méthode des bases réduites. En effet, cette méthode est un compromis entre la précision du calcul et sa vitesse. Si la perte en précision est trop importante pour pouvoir gagner suffisamment en temps, la méthode ne vaut alors pas le coup d'être appliquée.
Ici, tous les calculs sont fait en python, à l'exception des parties critiques (résolutions de systèmes), qui sont des appels de routines très optimisées. Dans notre implémentation python, le plus long n'est ainsi pas de résoudre les systèmes linéaires, mais de charger et calculer les données d'entrées. Une comparaison de performance en l'état aurait donc été trop biaisée par le language : on se réduit donc à tester la vitesse des deux fonctions critiques :

1. "scipy.linalg.solve", le solveur de matrice pleines, utilisé dans la méthode de bases réduites sur des matrices de taille $m \times m$.
2. "scipy.linalg.solve_banded", le solveur de matrices tridiagonales, utilisé dans la résolution des différences finies, sur des matrices de taille 79×79.

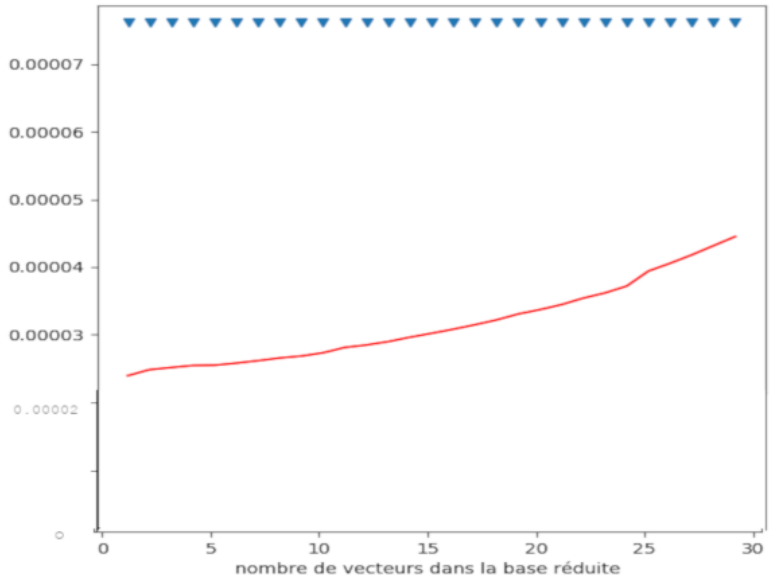


FIGURE 4 – En bleu, le temps moyen d'un appel à solve_banded (en secondes) (matrice de taille 79×79). En rouge, le temps moyen mit par scipy.linalg.solve en fonction de m (la matrice est de taille $m \times m$). Chaque donnée sur le graphe est la moyenne sur 6.10^7 appels de fonction. Tests réalisé sur un processeur AMD FX-4100 cadencé à 3.6 GHz.

Pour m pris autour de 5, on constate dans la figure 4 une multiplication des performances par 3. Ces résultats peuvent être biaisés par le temps de traitement des arguments, et par l'utilisation de scipy.linalg.solve_banded : cette routine permet de résoudre un système linéaire dont la matrice est une matrice bande : le code de cette fonction (accessible en ligne, voir [1]) fait plus de travail en python que scipy.linalg.solve. La comparaison des deux est donc délicate, et le gain de temps trouvé ici est approximatif.

7 Conclusion

Nous avons vu un aperçu de la méthode de bases réduites, des erreurs qu'elle produit et du gain en temps qu'elle permet. Pour choisir la meilleure base possible, un algorithme d'optimisation a été utilisé pour minimiser l'erreur commise sur les solutions. Une implémentation en python a été développée pour résoudre des équations de diffusion 1D, par différences finies et par la méthode des bases réduites.

Cette implémentation a pu montrer la rapidité du calcul dans la base réduite, et donner des ordres de grandeur des erreurs commises. Toutefois, l'ensemble d'apprentissage et l'ensemble de tests étaient sans doute trop homogènes et ne reflétaient pas un grand nombre de situations.

De plus, les comparaisons de performances ont été faites en python en utilisant différentes routines de SciPy. Ces résultats peuvent être biaisés par le temps de traitement des arguments et les algorithmes de SciPy.

La méthode de bases réduites a prouvé son efficacité et son intérêt. En utilisant cette méthode au sein des itérations de Schwarz, on peut retrouver un temps de calcul raisonnable, avec cette fois un meilleur couplage océan-atmosphère.

Ce projet laisse plusieurs questions :

— Quel est le gain de temps exact de cette méthode dans une simulation ?
— Quelle est l'erreur commise dans un contexte réaliste, en utilisant la méthode de base réduite en chaque point de grille sur un domaine horizontal de la taille d'un océan ?
— Peut-on utiliser une méthode analogue pour simplifier globalement le procédé de la méthode itérative de Schwarz ?

Ce projet d'introduction à la recherche en laboratoire était une expérience très enrichissante qui m'a permis de découvrir le contexte d'un laboratoire, d'avoir une première approche de la lecture d'article de recherche, et de mener à bien un travail assez long.

8 Annexe : Démonstrations

8.1 Conditions d'optimalité du problème d'optimisation

8.1.1 Cas continu

On cherche à optimiser le Lagrangien suivant :

$$\begin{aligned}\mathcal{L}(\Phi,\alpha^k,\lambda^k,\mu^k) =& \frac{1}{2}\sum_{k=1}^{S}\int_0^{t_f}(u^k-\widehat{u}^k)^T H^k(u^k-\widehat{u}^k)dt \\ &+\frac{\beta}{2}\sum_{j=1}^{m}(1-\phi_j^T\phi_j)^2 \\ &+\sum_{k=1}^{S}\int_0^{t_f}\lambda^{kT}\left(\Phi^T M\Phi\dot{\alpha^k}+\Phi^T K^k\Phi\alpha^k-\Phi^T f\right)dt \\ &+\sum_{k=1}^{S}\mu^{kT}\left(\Phi^T M\Phi\alpha^k(0)-\Phi^T M u_0^k\right)\end{aligned}$$

On va calculer la dérivée directionnelle de ce Lagrangien dans la direction des vecteurs $\delta\alpha^k$. On a par la formule de Green :

$$\int_0^{t_f}\lambda^{kT}\Phi^T M\Phi\dot{\alpha}^k dt = -\int_0^{t_f}\dot{\lambda}^{kT}\Phi^T M\Phi\alpha^k dt+\lambda^{kT}(t_f)\Phi^T M\Phi\alpha^k(t_f)-\lambda^{kT}(0)\Phi^T M\Phi\alpha^k(0)$$

Ce qui nous amène à l'expression suivante :

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\delta\alpha^k}(\alpha^k)=\langle\nabla_{\alpha^k}\mathcal{L},\delta\alpha^k\rangle &= \lim_{h\to 0}\frac{\mathcal{L}(\Phi,\alpha^k+h\delta\alpha^k,\lambda^k,\mu^k)-\mathcal{L}(\Phi,\alpha^k,\lambda^k,\mu^k)}{h} \\ &= \int_0^{t_f}\alpha^{kT}\Phi^T H^k\Phi\delta\alpha^k-u^{kT}H^k\Phi\delta\alpha^k dt \\ &\quad+\int_0^{t_f}\left[-\dot{\lambda}^{kT}\Phi^T M\Phi\delta\alpha^k+\lambda^{kT}\Phi^T K^k\Phi\delta\alpha^k\right]dt \\ &\quad+\lambda^{kT}(t_f)\Phi^T M\Phi\delta\alpha^k(t_f) \\ &\quad-\lambda^{kT}(0)\Phi^T M\Phi\delta\alpha^k(0)+\mu^{kT}\Phi^T M\Phi\delta\alpha^k(0)\end{aligned} \tag{23}$$

On pose ensuite $\nabla_{\alpha^k}\mathcal{L}=0$: donc $\forall\delta\alpha^k,\langle\nabla_{\alpha^k}\mathcal{L},\delta\alpha^k\rangle=0$.
On a donc nécessairement :

$$\begin{aligned}\alpha^{kT}\Phi^T H^k\Phi-u^{kT}H^k\Phi &= \dot{\lambda}^{kT}\Phi^T M\Phi-\lambda^{kT}\Phi^T K^k\Phi \\ \lambda^{kT}(t_f)\Phi^T M\Phi &= 0 \\ \lambda^{kT}(0)\Phi^T M\Phi &= \mu^{kT}\Phi^T M\Phi\end{aligned}$$

Donc :

$$\begin{aligned}(\alpha^{kT}\Phi^T-u^{kT})H^k\Phi &= \dot{\lambda}^{kT}\Phi^T M\Phi-\lambda^{kT}\Phi^T K^k\Phi \\ \lambda^k(t_f) &= 0 \\ \lambda^k(0) &= \mu^k\end{aligned}$$

8.1.2 Cas discret

On cherche à optimiser le Lagrangien suivant :

$$\begin{aligned}\mathcal{L}(\Phi,\alpha^k,\lambda^k,\mu^k) =& \frac{1}{2}\sum_{k=1}^{S}\sum_{n=1}^{N-1}(u_n^k-\widehat{u}_n^k)^T H^k(u_n^k-\widehat{u}_n^k)\\ &+\frac{\beta}{2}\sum_{j=1}^{m}(1-\phi_j^T\phi_j)^2\\ &+\sum_{k=1}^{S}\sum_{n=0}^{N-1}{\lambda_n^k}^T((\frac{1}{\delta t}\Phi^T M\Phi+\Phi^T K^k\Phi)\alpha_{n+1}^k-\frac{1}{\delta t}\Phi^T M\Phi\alpha_n^k-\Phi^T f_{n+1})\\ &+\sum_{k=1}^{S}{\mu^k}^T\left(\Phi^T M\Phi\alpha_0^k-\Phi^T M u_0^k\right)\end{aligned}$$

Or avec un changement de variable dans la somme,

$$\begin{aligned}&\sum_{n=0}^{N-1}{\lambda_n^k}^T((\frac{1}{\delta t}\Phi^T M\Phi+\Phi^T K^k\Phi)\alpha_{n+1}^k-\frac{1}{\delta t}\Phi^T M\Phi\alpha_n^k)\\ =&\sum_{n=1}^{N}{\lambda_{n-1}^k}^T(\frac{1}{\delta t}\Phi^T M\Phi+\Phi^T K^k\Phi)\alpha_n^k+\sum_{n=0}^{N-1}-\frac{1}{\delta t}{\lambda_n^k}^T\Phi^T M\Phi\alpha_n^k)\\ =&\sum_{n=1}^{N-1}\left((\frac{{\lambda_{n-1}^k}^T-{\lambda_n^k}^T}{\delta t})\Phi^T M\Phi\alpha_n^k+{\lambda_{n-1}^k}^T\Phi^T K^k\Phi\alpha_n^k\right)\\ &+\frac{{\lambda_{N-1}^k}^T\Phi^T M\Phi\alpha_N^k-{\lambda_0^k}^T\Phi^T M\Phi\alpha_0^k}{\delta t}+{\lambda_{N-1}^k}^T\Phi^T K^k\Phi\alpha_N^k\end{aligned}$$

De la même manière que dans le cas continu :

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\delta\alpha^k}(\alpha^k)=\langle\nabla_{\alpha^k}\mathcal{L},\delta\alpha^k\rangle =& \lim_{h\to 0}\frac{\mathcal{L}(\Phi,\alpha^k+h\delta\alpha^k,\lambda^k,\mu^k)-\mathcal{L}(\Phi,\alpha^k,\lambda^k,\mu^k)}{h}\\ =&\sum_{n=1}^{N-1}{\alpha_n^k}^T\Phi^T H^k\Phi\delta\alpha_n^k-{u_n^k}^T H^k\Phi\delta\alpha_n^k\\ &+\sum_{n=1}^{N-1}\left((\frac{{\lambda_{n-1}^k}^T-{\lambda_n^k}^T}{\delta t})\Phi^T M\Phi\delta\alpha_n^k+{\lambda_{n-1}^k}^T\Phi^T K^k\Phi\delta\alpha_n^k\right)\\ &+\frac{{\lambda_{N-1}^k}^T\Phi^T M\Phi\delta\alpha_N^k}{\delta t}+{\lambda_{N-1}^k}^T\Phi^T K^k\Phi\delta\alpha_N^k\\ &-\frac{{\lambda_0^k}^T\Phi^T M\Phi\delta\alpha_0^k}{\delta t}+{\mu^k}^T\Phi^T M\Phi\delta\alpha_0^k\end{aligned}\tag{24}$$

Et avec le même raisonnement sur $\nabla_{\alpha^k}\mathcal{L}=0\Rightarrow\forall\delta\alpha^k,\langle\nabla_{\alpha^k}\mathcal{L},\delta\alpha^k\rangle=0$:

$$\Phi^T M\Phi\frac{\lambda_{n-1}^k-\lambda_n^k}{\delta t}+\Phi^T{K^k}^T\Phi\lambda_{n-1}^k=\Phi^T H^k(u_n^k-\Phi\alpha_n^k)$$

$$\lambda_{N-1}^k = 0$$
$$\mu_k = \frac{\lambda_0^k}{\delta t}$$

8.2 Calcul de $\nabla_\Phi \mathcal{L}$

On a l'expression discrète de $\mathcal{L}$ suivante :

$$\begin{aligned}\mathcal{L}(\Phi, \alpha^k, \lambda^k, \mu^k) =& \frac{1}{2}\sum_{k=1}^{S}\sum_{n=1}^{N-1}(u_n^k - \widehat{u}_n^k)^T H^k (u_n^k - \widehat{u}_n^k) \\ &+ \frac{\beta}{2}\sum_{j=1}^{m}(1-\phi_j^T\phi_j)^2 \\ &+ \sum_{k=1}^{S}\sum_{n=0}^{N-1}\lambda_n^{kT}((\frac{1}{\delta t}\Phi^T M\Phi + \Phi^T K^k\Phi)\alpha_{n+1}^k - \frac{1}{\delta t}\Phi^T M\Phi\alpha_n^k - \Phi^T f_{n+1}) \\ &+ \sum_{k=1}^{S}\mu^{kT}\left(\Phi^T M\Phi\alpha_0^k - \Phi^T M u_0^k\right)\end{aligned}$$

la première lignes se dérive de la manière suivante :

$$\begin{aligned}&\frac{1}{2}\frac{(u^k - (\Phi + h\delta\Phi)\alpha^k)^T H^k(u^k - (\Phi+h\delta\Phi)\alpha^k) - (u^k - \Phi\alpha^k)^T H^k(u^k - \Phi\alpha^k)}{h} \\ &\xrightarrow[h\to 0]{} \alpha^{kT}\Phi^T H^k\delta\Phi\alpha^k - u^{kT}H^k\delta\Phi\alpha^k = \langle H^k\Phi\alpha^k\alpha^{kT} - H^k u^k\alpha^{kT}, \delta\Phi\rangle\end{aligned}$$

Lorsqu'on dérive la seconde ligne par rapport à un vecteur ϕ_j, on obtient :

$$\frac{\beta}{2}\frac{(1-\phi_j^T\phi_j - 2h\phi_j^T\delta\phi_j + h^2\delta\phi_j^T\delta\phi_j)^2 - (1-\phi_j^T\phi_j)^2}{h} \xrightarrow[h\to 0]{} -2\beta\phi_j^T(1-\phi_j^T\phi_j)\delta\phi_j$$

la matrice Φ étant constituée de vecteurs colonnes ϕ_j, la dérivée selon la matrice Φ nous donne :

$$\langle -2\beta\Phi diag(1-\phi_i^T\phi_i), \delta\Phi\rangle$$

Pour la troisième et la quatrième ligne, on étudie la dérivée selon Φ d'un produit de la forme $\lambda^T\Phi^T A\Phi\alpha$:

$$\begin{aligned}&\lambda^T\Phi^T A\delta\Phi\alpha + \lambda^T\delta\Phi^T A\Phi\alpha = \lambda^T\Phi^T A\delta\Phi\alpha + \alpha^T\Phi^T A^T\delta\Phi\lambda \\ &= \langle A^T\Phi\lambda\alpha^T + A\Phi\alpha\lambda^T, \delta\Phi\rangle\end{aligned}$$

La partie du gradient associé à la troisième ligne est donc la suivante (on a M symétrique) :

$$\sum_{k=1}^{S}\sum_{n=0}^{N-1}\frac{M\Phi}{\delta t}\left(\lambda_n^k(\alpha_{n+1}^k - \alpha_n^k)^T + (\alpha_{n+1}^k - \alpha_n^k)\lambda_n^{kT}\right) + K^k\Phi\alpha_{n+1}^k\lambda_n^{kT} + K^{kT}\Phi\lambda_n^k\alpha_{n+1}^{kT} - f_{n+1}\lambda_n^{kT}$$

Finalement, on a l'expression suivante de $\nabla_\Phi \mathcal{L}$:

$$\begin{aligned}
\nabla_\Phi \mathcal{L} = & \sum_{k=1}^{S} \sum_{n=1}^{N-1} \left[H^k \Phi \alpha_n^k \alpha_n^{k^T} - H^k u_n^k \alpha_n^{k^T} \right] \\
& - 2\beta \Phi \, diag(1 - \phi_i^T \phi_i) \\
& + \sum_{k=1}^{S} \sum_{n=0}^{N-1} \frac{M\Phi}{\delta t} \left(\lambda_n^k (\alpha_{n+1}^k - \alpha_n^k)^T + (\alpha_{n+1}^k - \alpha_n^k) \lambda_n^{k^T} \right) \\
& \qquad + K^k \Phi \alpha_{n+1}^k \lambda_n^{k^T} + K^{kT} \Phi \lambda_n^k \alpha_{n+1}^{k^T} - f_{n+1} \lambda_n^{k^T} \\
& + \sum_{k=1}^{S} \left(M\Phi(\alpha_0^k \mu^{k^T} + \mu^k \alpha_0^{k^T}) - M u_0^k \mu^{k^T} \right)
\end{aligned}$$

Références

[1] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy : Open source scientific tools for Python, 2001–. [http ://www.scipy.org/].

[2] K. Willcox, O. Ghattas, B. Van Bloemen Wa, and B. Bader. An optimization framework for goal-oriented, model-based reduction of large-scale systems. [http ://citeseerx.ist.psu.edu/viewdoc/summary ?doi=10.1.1.146.1591].