Projet d'algèbre linéaire numérique :
Utilisation des EOFs pour prédire la température des Océans

Phase 1 :
Étude de l'algorithmique et création d'un prototype

Naji Boulami
Arthur Manoha
Philippe Leleux

Résumé : Dans l'optique de prédire les variations de température des océans, ce projet implique :

- l'étude de la théorie entourant les EOFs
- l'implantation d'un algorithme de création des EOFs et PCs
- l'étude d'une méthode de prédiction à partir des EOFs et PCs
- l'étude et l'implantation d'une méthode d'approximation des espaces propres

Table des matières

I Présentation générale

1.1 Introduction

La science de la météorologie est la science de la prédiction des phénomènes atmosphérique. Pour déterminer quel sera le temps de demain, il faut posséder beaucoup de données passées et déterminer un « comportement », définir une fonction qui permet d'expliquer les changements au cours du temps et de prédire les prochaines variations. Cette science se rapporte à d'autres types d'étude telles que l'étude des variation de la bourse, du niveau de pression des océans ou, et c'est le phénomène dans lequel cadre ce projet, la température des océans.

Ces phénomènes sont extrêmement complexes et ne permettent pas une étude et une prédiction exacte d'autant plus que le nombre de données à traiter est immense, il faut donc trouver des méthodes permettant de déterminer l'avenir avec la plus grande précision possible d'où l'utilisation des probabilités et des statistiques.

Nous allons nous intéresser à une méthode mise au point dans l'optique de diminuer le nombre de dimensions dans lesquels on étudie le phénomène et de diminuer le nombre de données à traiter tout en permettant de prédire les phénomènes avec une certaine précision : la méthode des EOFs (Empirical Orthogonal Functions).

1.2 Distribution fournie

Certains fichiers sources Matlab sont fournis avec le projet :

- EOF.m contient le code principal et peut lire des données à partir d'un des deux autres fichiers matlab.
- Kaplan.mat contient des données de température de la surface de la mer.
- gendata.m qui génère des données synthétiques, on peut modifier ce programme au travers de certains paramètres et différentes valeurs.

1.3 Spécifications

Les algorithmes doivent être codés sous Matlab. On considérera moins l'efficacité d'un algorithme que son fonctionnement utile et effectif.

II Analyse de données climatiques et des EOFs

2.1 Introduction à l'analyse climatique grâce aux EOFs

Résumé de « A Primer for EOF Analysis of Climate Data » de A. Hannachi :

La variation du climat est due aux interactions non-linéaires qui dépendent de beaucoup de paramètres. En effet, pour l'analyse de ces données et la prédiction du climat, il est nécessaire de pouvoir simplifier les dimensions du système ainsi que réduire le nombre de paramètres (variables). Les climatologues se sont penchés sur ce problème et ont cherché à extraire les comportements qui expliquent les phénomènes climatiques des mesures de variables atmosphériques.
Il en a résulté une méthode mathématique simple appliquée à la science climatique qui est la décomposition orthogonale en valeurs propres (EOF de son nom en anglais Empirical Orthogonal Functions). Elle nous facilite la vie puisqu'elle permet de réduire le nombre de variable des données sans changer la variance pour des grandes mesures. Cette méthode se base sur le principe de la décomposition des données avec des fonctions orthogonales déterminées empiriquement à partir des données.

Créer la matrice de donnée :

Les informations climatiques son données sous la forme d'une matrice à trois dimensions, la latitude, la longitude et le temps. Pour avoir des calculs simples, on essayera de travailler sur des matrices de deux dimensions et cela en considérant les deux dimensions spatiales comme une dimension, ce qui nous permet d'obtenir une matrice X où chaque ligne correspond à toutes les mesures prises à un instant donné.

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1\mathrm{p}} \\ x_{21} & x_{22} & \dots & x_{2\mathrm{p}} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

On crée ensuite le vecteur ligne contenant les moyennes temporelles (moyenne calculée sur une mesure dans le temps) et on le soustrait à X, obtenant ainsi le vecteur X' : le champ d'anomalie.

Pondération spatiale :

La Terre n'étant pas une sphère parfaite et les dimensions étant la latitude et la longitude, certains endroits ont une densité supérieur aux autres. Pour contrer cela, on donne un poids à chaque donnée en se basant sur la surface qu'elle domine. On va donc pondérer les mesures en multipliant matriciellement X' par la matrice diagonale de coefficients le cos des latitude à chaque position.

EOFs et PCs :

Après avoir calculé la matrice X', on introduit la matrice de covariance, qui donne la

corrélation entre les différents points de la grille :

$$\Sigma = \frac{1}{(n-1)} X'^T X'$$

On cherche ensuite les variables qui causent un maximum de variance : **a** = $(a_1, \dots, a_p)^T$ / var(X'**a**) soit maximum.
De plus on prend chaque a unitaire ce qui permet de simplifier le problème, il devient alors :

$$\Sigma a = \lambda a$$

Par définition de la matrice de covariance, Σ est symétrique réelle définie positive, elle est donc diagonalisable dans une base orthonormée constituée de vecteurs propres a_k associés aux valeurs propres λ_k.

Les EOFs sont en fait ces vecteurs propres, ils permettent d'obtenir les directions qui expliquent la variance et grâce aux valeurs propres on peut savoir en quelle proportion ces vecteurs propres participent à la variance, c'est la variance expliquée qui la caractérise :
$\frac{100\times\lambda_k}{(\lambda_1+\dots+\lambda_p)} = \frac{100\times\lambda_k}{tr(D)}(\%)$, avec D matrice diagonale à laquelle Σ est semblable, cette formule donne en pourcentage la participation de chaque EOF à la variance.

La projection du champ d'anomalie sur chacune des EOFs donne les PCs :

$$c_k = X' a_k$$

$$c_k(t) = \sum_{s=1}^{p} x'(t,s) a_k(s)$$

Utilisation des EOFs :

Comme Σ est symétrique, par construction les a_k sont orthogonaux et les PCs forment une famille libre on peut ainsi écrire :

$$X'(t,s) = \sum_{s=1}^{p} c_k(t) a_k(s)$$

Cependant il existe un problème : les phénomènes physiques ne sont en général pas orthogonaux et les EOFs imposent alors des contraintes très grandes sur les phénomènes et de plus, les valeurs propres peuvent être dégénérées (plusieurs valeurs propres non distinctes). Une nécessité apparaît, celle d'avoir une mesure d'incertitude sur le calcul des valeurs propres, on applique pour cela la règle du pouce :

$\Delta\lambda_k \approx \lambda_k\sqrt{\frac{2}{n}}$ avec λj plus proche valeur propre de λ_k et n la taille de l'échantillon.
$\Delta a_k \approx (\Delta\lambda_k/(\lambda_j - \lambda_k))a_j$

Le dernier problème apparaît au moment de la troncation, lorsque l'on sélectionne les λ_k les plus importantes pour l'explication de la variance en se fixant un seuil en pourcentage et en cherchant les EOFs qui l'explique grâce à la variance expliquée.

Aspect computationnel :

En pratique, on ne résout pas l'équation Σ**a** = λ**a**, on va utiliser un outil d'algèbre linéaire : la décomposition en valeurs singulières, on aura ainsi :

$Y = L\Lambda R^T$ avec L matrice pxp orthogonale dont les colonnes donnent les EOFs
R matrice nxn orthogonale dont les colonnes donnent les PCs

Interpretation of EOFs :

Les EOFs ne dépendent pas du temps, elles représentent seulement une direction de variance importante qui définira ensuite un mode d'oscillation, ce sont les PCs qui dépendent du temps en donnant le signe et l'amplitude engendrée par l'EOF pour décrire les différents états du phénomène. Lorsque les valeurs propres ne sont pas dégénérées, on peut étudier les EOFs séparément mais pas si elles le sont malgré l'orthogonalité des EOFs et la liberté de la famille des PCs.
Les EOFs représentent des comportements qui peuvent expliquer les variances rencontrées pourtant leur interprétation physique porte à controverse car cette méthode impose des contraintes purement géométriques et qui ne peuvent être physiques d'où la nécessité d'améliorer cette méthode ce qui a donné naissance aux Rotated EOFs et aux Extended EOFs.

2.2 Scénario illustrant la prédiction grâce aux EOFs

On dispose d'un ensemble de données décrivant les variations de la température des océans sous la forme d'une matrice p,n où p est le produit de p1 (nombre de discrétisation de la latitude), p2 (nombre de discrétisation de la longitude) et n le nombre de mesures effectuées au cours du temps. Grâce à la méthode décrite dans le document de monsieur Hannachi et résumée en 1), nous obtenons les EOFs, correspondant aux vecteurs propres de la matrice de covariance du champ d'anomalie, et les PCs, projection du champ d'anomalie sur les EOFs.

Avec un choix convenable du nombre de termes prépondérants, réalisé au cas par cas grâce au spectre de la matrice de covariance représentée graphiquement avec la variance exprimée associée, on obtient une somme tronquée de termes. Avec ces EOFs, on obtient une très bonne approximation de la matrice d'écart à la moyenne. On peut alors interpréter ces écarts en chaque point du globe.

Pour illustrer la manière dont les EOFs peuvent être utilisées pour la prédiction des variations de température de la mer, nous allons occulter les 12 derniers mois dans les données que l'on possède. Puis, en se servant des résultats des mois et années précédents, nous allons prédire ces variations. Il ne restera alors qu'à observer les différences entre prédiction et réalité dans le temps. Observer si la différence entre les deux augmente dans le temps en observant différentes phases des résultats et ainsi valider ou invalider le modèle utilisé.

2.3 Principes d'implantation, utilisation des EOFs et validation de la prédiction

L'algorithme à implanter se fait en plusieurs étapes :

- récupération les données sous forme de matrice 3D (latitude, longitude, temps)
- transformation en matrice 2D en concaténant les matrices 2D issues de la matrice 3D ; les deux dimensions de cette nouvelle matrice sont alors le temps d'une part, l'union des deux coordonnées spatiales d'autre part :

```
[n,p1,p2] = size(F);
X = reshape(F,n,p1*p2);
```

- Calcul du vecteur ligne des moyennes temporelles X

Xbar = mean(X,1);

- Calcul du champ d'anomalie X'

*Xprime = X-ones(n,1)*Xbar;*

- Calcul de la matrice de covariance Σ du champ d'anomalie

*Sigma = (1/n-1)*Xprime^T*Xprime;*

- application de la décomposition en valeurs singulières sur la matrice de covariance (en utilisant la commande déjà écrite svd) pour trouver les EOFs et les PCs directement : on décompose la transposée de la matrice de covariance en LΛR où les colonnes de L sont les PCs et les colonnes de R les EOFs :

[EOFs, PCs, var] = svd(Sigma);

La raison derrière le scénario de la question précédente est que prédire un phénomène avec cette méthode ou une autre revient à écrire un programme d'apprentissage et ainsi découper la réalisation en trois parties distinctes :

- la première partie est une partie d'apprentissage, on prend des données connues dont on extrait les principaux comportements, pour nous cela correspondra aux EOFs et aux PCs que l'on calcule.

- Une fois l'apprentissage effectué commence une deuxième phase : la validation du modèle de prédiction obtenu. Pour cela on doit disposer d'un autre ensemble de données connues suivant celles déjà étudiées, on va essayer alors de trouver un moyen de vérifier la précision de l'algorithme.

-Quand on obtient un algorithme que l'on estime satisfaisant, il n'y a plus qu'à l'utiliser en cherchant toujours à l'améliorer et le surveiller.

Cette manœuvre se répète puisque plus le nombre de données est important, plus les informations dont on dispose sont importantes et plus on peut espérer que la prédiction soit précise.

Ainsi en gardant les premières mesures, on définit l'échantillon d'apprentissage puis les mesures des 12 mois suivants représente la période de validation et enfin il ne reste plus qu'à appliquer si l'algorithme est bon.

Pour valider la précision de la prédiction, on utilise une partie des données (toutes celles qui précèdent les 12 derniers mois) pour construire le prédicteur, c'est à dire trouver les EOFs et Pcs, puis on utilise ces derniers pour prédire la fin des données et on compare avec les données réelles que l'on connaît : la comparaison des mesures et des prédictions permet de statuer sur la précision et la validité du modèle.

III Implantation d'une approximation des espaces

propres pour une matrice symétrique d'ordre n

3.1 Difficulté avec la méthode de la puissance itérée

La méthode de la puissance itérée

Le but est d'extraire successivement les valeurs propres et vecteurs propres de la matrice A, en commençant par les valeurs propres de plus grand module.

Méthode de la puissance

On applique d'abord la méthode de la puissance, qui donne une estimation de la plus grande valeur propre et du vecteur propre associé (mais seulement de la plus grande). Pour cela, on définit un vecteur x(0) de Rn, et la suite (x(n)) avec x(n+1)=Ax(n). Cette suite converge vers un multiple du premier vecteur propre v1; le quotient <x(n+1)|x(n)> / ||x(n)||2 tend vers la valeur propre lambda(n). Il faut ensuite modifier la matrice A pour « supprimer » la valeur propre λ_n . C'est le rôle de la méthode de déflation.

Méthode de déflation

La matrice A possède comme valeurs propres λ_1, λ_2, …, λ_n.
On effectue $A = A - \lambda_i \times w \times w^T$.
Cela modifie le spectre de A qui devient λ_2 , λ_3, …, λ_n, 0.

Méthode de la puissance itérée

En alternant les deux méthodes précédentes, on extrait successivement les premières valeurs propres et les vecteurs correspondants.

Problème lié à la condition d'arrêt

La difficulté en étendant la méthode des puissances itérées pour créer par ordinateur un bloc de m vecteurs et valeurs propres (m<<n) de A associés à la plus grande valeur propre de A se trouve au niveau de la condition d'arrêt.
Nous avons une condition d'arrêt du type : $\frac{\beta(p-1)-\beta}{\beta} < \varepsilon$et nous avons non seulement un problème au niveau du ε mais également au niveau du calcul du terme de gauche.
En ce qui concerne le ε, il faut pouvoir trouver une valeur qui permette d'être assez précis sans dépasser les capacités de la machine mais cela est difficile car on pourrait s'arrêter trop tôt. La raison en est que les valeurs propres forment une suite décroissante avec des termes très écartées au départ puis très proche (ce qui forme une courbe quasi continue) et de plus, ces valeurs propres deviennent très petites ainsi la machine, effectuant des approximations successives va engendrer des calculs de plus en plus faux.
Il faudrait pouvoir implanter un algorithme avec une condition d'arrêt différente et engendrant moins d'erreurs de calcul pour pouvoir obtenir une méthode des puissances itérées efficace.

3.2 Implantation du calcul des valeurs et vecteurs propres

Dans un premier algorithme, l'implantation de la méthode des puissances expliquée en 1), on cherche à calculer la plus grande valeur propre.

Voici l'algorithme Matlab :

```
function [v,lambda] = puissance(A)

EPS = 0.00001; % utilisé dans la condition d'arrêt
NMAXIT = 10000; % maximum d'itération avant que l'on ne considère que l'algorithme ne converge pas

[n m] = size(A);
v = rand(m,1); % vecteur initial arbitraire
                % si l'algorithme ne converge pas, on relance pour avoir un autre v

%initialisations
b1 = 0;
b0 = 1;
nit = 0;

% boucle de calcul
while ((nit < NMAXIT) && ((norm(b1-b0)/norm(b0)) > EPS))
   y = A*v;
   v = y/norm(y); % les v convergent vers le vecteur propre associé à lambda
   b0 = b1;
   b1 = transpose(v)*A*v;
end

lambda = b1;
```

Dans un second algorithme, on implante la méthode des puissances itératives expliquée en 1) pour calculer un ensemble de vecteurs propres (que l'on précise en paramètre).

Voici l'algorithme Matlab :

```
function [v,lambda] = puissance_iteree(A,m)

% A matrice dont on cherche les valeurs propres
% m nombre de couples (vecteur, valeur) recherchés

%initialisations
a = length(A);
v = ones(a);
lambda = ones(a,1);


[v(:,1) lambda(1)] = puissance(A);
W = v(:,1)/norm(v(:,1));
A = A – lambda(1)*W*transpose(W);

for i = 2:m
   A = A - lambda(i)*W*transpose(W);
```

```
  [v(:,i) lambda(i)] = puissance(A);

   W = v(:,i)/norm(v(:,i));
end
```

3.3 Optimisation de l'algorithme

Voici l'algorithme de la puissance itérée améliorée qui, dans le contexte de notre application peut être plus efficace pour créer une estimation des vecteurs propres dominants de la matrice symétrique A et également des PCs. Pour fonctionner, cet algorithme doit être précédé du contenu de EOF.m :

```
% 1/ EOF on the whole data minus last year.
F=mydata(1:nt-12,:);

% Remove undefined values
F(F==-32768)=0;

[n,p1,p2] = size(F);
X = reshape(F,n,p1*p2);
Xbar= mean(X,1); % vecteur ligne contenant les moyennes de chaque colonne
Xprime = X-ones(n,1)*Xbar; % matrice des écarts à la moyenne

Sigma = (1/n-1)*transpose(Xprime)*Xprime;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%ALGORITHME AMELIORE DE CALCUL DES COUPLES
PROPRES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

%initialisations
p = p1*p2;
EOFs = zeros(p);
lambda = zeros(p,1);
sigmabis = Sigma; % on conserve sigma en mémoire
somme_lambda = trace(sigmabis);
poids_total = 0;

i = 1;
while poids_total < threshold

   %calcul de la ième valeur propre et du vecteur propre associé
   [EOFs(:,i) lambda(i)] = puissance(sigmabis);

   %déflation
   W = EOFs(:,i)/norm(EOFs(:,i));
   sigmabis = sigmabis - lambda(i)*W*transpose(W);

   %incrémentation du poids total
   poids_total = poids_total + (100*lambda(i)/somme_lambda);
   poids_total = poids_total
   i = i+1;
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%FIN
CALCUL%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%

 PC =  Xprime*EOFs; %Calcul des PC
```

IV Conclusion

Cette première phase de projet introduit un sujet aussi intéressant que vaste, rien que pour comprendre la méthode des EOFs ou même à quoi correspondent ces EOFs ou les PCs. Comment utiliser la méthode de décomposition en valeur singulière ou la méthode des puissances itérées dans la vie réelle ? C'est une question qui a dû passer par la tête de tout élève ayant touché de près ou de loin à l'algèbre linéaire numérique. Essayer de créer un algorithme de prédiction concernant le climat, les phénomènes atmosphériques ou la température des océans nous a permis de côtoyer un domaine dont on entend beaucoup parler mais dont on ne sait pas grand chose au final.

Nous avons rencontré de nombreux problème, outre la gestion du temps difficile en raison du délai relativement court pour pouvoir effectuer un travail de cette ampleur mais surtout au niveau de l'utilisation des EOFs pour les prédictions. Cela nous a pourtant permis d'approfondir nos connaissance en allant au delà de la présentation de M. Hannachi en cherchant d'autres informations (nombreuses) pour essayer de créer une méthode. Malheureusement, nous n'avons pas pu atteindre nos objectifs et créer un algorithme efficace et surtout fonctionnant réellement.

Pour la deuxième phase, les objectifs sont clairs : gérer mieux le temps de travail tout en continuant à travailler et réfléchir en groupe et même à plusieurs groupe, le partage d'idée ne peut qu'aboutir à quelque chose de meilleur.