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

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'étude de l'algorithme de calcul 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 Introduction

I.1. Présentation Générale

La science de la météorologie est la science de la prédiction des phénomènes atmosphériques. 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.

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).

I.2. Projet initial et actuel

L'année dernière ce même projet nous était proposés, il était découpé en 2 phases. La première se focalisait sur le calcul des EOFs par la méthode des puissances itérées et la compréhension de leur signification et utilisation (la prédiction). La deuxième phase va plus loin en étendant la méthode des puissances en utilisant la projection de Raleigh-Ritz et en implantant la prédiction.

Nous reprenons ce projet mais en partie seulement. Toute la partie codage, qui exigeait d'implanter le calcul des EOFs et la prédiction ainsi que les différents tests permettant d'obtenir des résultats quantitatifs ne sont plus demandés. Nous allons nous focaliser sur la partie algorithmique et le principe du calcul en commentant et expliquant chaque algorithme et en étudiant autant que possible l'efficacité relative de chacun.

II Analyse de données climatiques et des EOFs

II.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. Ainsi, 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.

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

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** = $(a_1, \ldots, a_p)^T$ / 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 :

$$\frac{100 \times \lambda_k}{(\lambda_1 + \ldots + \lambda_p)} = \frac{100 \times \lambda_k}{tr(D)} \text{ (\%)},$$

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 :

$$\Delta \lambda_k \approx \lambda_k \frac{\sqrt{2}}{n}$$

$$\Delta a_k \approx (\Delta \lambda_k / (\lambda_j - \lambda_k)) a_j$$

avec λ_j plus proche valeur propre de λ_k et n la taille de l'échantillon.

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 $\Sigma\mathbf{a} = \lambda\mathbf{a}$, 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

Interprétation des 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.

II.2 Principes d'implantation, utilisation des EOFs et validation de la prédiction

L'algorithme à implanter se fait en deux étapes : calcul des EOFs puis prédiction.

2.3.a Calcul des EOFs

On considère que l'on dispose au début de l'algorithme :

- d'une matrice *F* à 3 dimensions de données : n, p1, p2 (temps, latitude et longitude)
- d'un paramètre *percentReached* : variance que l'on souhaite expliquer par les EOFs que l'on garde.

1. On transforme d'abord cette matrice en matrice à 2 dimensions : 1 de temps et une d'espace, concaténation de la latitude et la longitude :

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

2. On calcule le champ d'anomalie Z (grâce au vecteur ligne des moyennes temporelles calculé au préalable) et la matrice de covariance S :

```
Xbar = mean(X,1);
Z = X-ones(n,1)*Xbar;
S = Z'*Z;
```

3. L'étape suivante consiste à calculer les paires propres de S et à choisir le nombre d'EOFs que l'on garde grâce au *percentReached :*

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

Remarque : i. Dans cette première version du calcul, on fait simplement une décomposition en valeurs singulières (en utilisant une des fonctions de matlab par exemple) sans préciser comment. Ce calcul sera précisé en ***III*** *avec des algorithmes basés sur la méthode des puissances itérées.*

ii. Vérification de la qualité des EOFs : Nous pouvons vérifier la qualité des EOFs ainsi trouvées en calculant le terme $\Delta = ||ZVVT-Z||/||Z||$, celui-ci doit être petit. Il est aussi possible de prendre une autre base de vecteurs pris aléatoirement et de montrer que celle-ci a un plus mauvais Δ.

iii. Par cette décomposition, on trouve bien des EOFs indépendantes du temps et des Pcs en dépendant.

II.2.b Utilisation des EOFs pour la prédiction

On considère au début de l'algorithme :

- que la matrice de données contenait ny années de données.
- qu'à celle-ci on enlève la dernière et on calcule les EOFs avec la méthode vu précédemment.

1. Dans l'année restante, on considère que certaines colonnes de données sont inconnues et on les enlève. Pour cela, on peut par exemple enlever un pourcentage fixé de colonnes de manière aléatoire de la matrice. On a alors sur les colonnes de données restantes :

$$F = \alpha V^T$$ avec F est la matrice des données connues sur la dernière année

V est la matrice des EOFs tronquée des données inconnues (à chaque colonne de données enlevée on retire la ligne correspondante dans l'EOF).

2. Trouver α ! On souhaite résoudre le système surdéterminé précédent ce qui revient à un problème des moindres carrés.

$$\alpha = F\left(V^T\right)^{-1}$$

3. C'est ici qu'apparait le principe de notre prédiction : nous allons considérer que α est non seulement valable pour les colonnes de données connues mais également pour celles de la matrice que l'on ne connait pas à priori. La matrice de données entière est alors :

$$F' = \left(V\,\alpha\right)^T$$

Remarque : i. Ici on ne détaille pas le calcul de α et la résolution du problème des moindre carrés, n'importe quelle méthode pourrait être utilisé.

ii. Vérification de la qualité de la prédiction : Nous pouvons vérifier la qualité de la prédiction ainsi effectuée en calculant le terme Δ = ||G-F'||/||G|| où G est la vraie matrice de donnée, celui-ci doit être petit. Il est aussi possible de prendre une autre base de vecteurs que les EOFs pris aléatoirement et de montrer que celle-ci a un plus mauvais Δ.

III Implantation d'une approximation des espaces propres pour une matrice symétrique d'ordre n

III.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 λ_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 rangées dans l'ordre décroissant. Soit w_1 le vecteur propre v_1 normalisé, on effectue $A = A - \lambda_1 \times w_1 \times w_1^T$ et le spectre de A devient λ_2, λ_3, …, λ_n, 0.

Ainsi successivement pour les n paires propres de A, on calcule la paire puis on la "supprime" du spectre, c'est la méthode de déflation.

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 : $\frac{\beta_{n+1}-\beta_n}{\beta_n} < \varepsilon$ avec β_n approximation de la valeur propre à l'étape n

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.

III.2 Implantation du calcul des valeurs et vecteurs propres

III.2.a Méthode de la puissance

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

- ε écart maximum pour le critère d'arrêt (par ex. 10^{-5})
- imax nombre maximum d'itération (par ex. 10000)
- A matrice n x m

1. Initialisation :

```
v = random(m,1); – vecteur initial aléatoire de dimension m
βn+1 = 0;        – valeur de β1
βn = 1;          – valeur de β0
i = 0;           – nombre d'itérations
```

2. *Boucle de calcul :*

```
Tant que i < imax et | βn+1 – βn |/| βn |

v = A*v ;
v = v / ||v|| ;
βn=βn+1 ;
βn+1=vT*A*v ;
i++ ;

fin Tant que

retourner(v, βn+1) ;
```

III.2.b Méthode des puissances itérées

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). On considère :

- Les mêmes conditions que précédemment.
- *puissance(A)* est la fonction définie précédemment appliquée à A et qui renvoie vecteur et valeur propre.

1. Initialisation :

```
v = ones(A);              – vecteur propre initial : une matrice carrée de taille la plus grande dimension de A
                            et ne contenant que des 1.
lambda = ones(a,1);       – valeur propre initial : un vecteur de taille la plus grande dimension de A et ne
                            contenant que des 1.
[v(:,1) lambda(1)] = puissance(A);   --première application de la méthode de la puissance à A
W = v(:,1)/norm(v(:,1));             – normalisation du vecteur propre obtenu
```

```
A = A – lambda(1)*W*transpose(W);
```
– première application de la méthode de la déflation

A ce niveau-là, la première colonne de v contient le premier vecteur propre de A et lambda la première valeur propre.

***2.** Boucle de calcul :*

```
Pour i de 2 à m faire

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

Fin pour

retourner(v, lambda) ;
```

III.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. On considère que l'on a :

- S la matrice de covariance du champ d'anomalie Xprime.
- p la taille de la dimension d'espace
- *threshold* le pourcentage de variance que l'on cherche à expliquer

1. Initialisation :

```
EOFs = zeros(p);
```
– la matrice des EOFs (carré de dimension p) initialisée à 0
```
lambda = zeros(p,1);
```
– le vecteur des valeurs propres de dimension p initialisé à 0
```
trace = trace(S);
```
– la trace de S
```
percentReached = 0;
```
--pourcentage de variance expliquée actuelle
```
i = 1;
```
– compteur de boucle

***2.** Boucle de calcul :*

```
Tant que percentReached < threshold et i < imax
```

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

W = EOFs(:,i)/norm(EOFs(:,i));
```
– normalisation de l'EOF i
```
S = S – lambda(i)*W*transpose(W);
```
– déflation

--incrémentation du percentReached
```
percentReached = percentReached+(100*lambda(i)/somme_lambda);
percentReached = percentReached;
i = i+1;

fin Tant que

PC =  Xprime*EOFs;
```
– Calcul des PCs

IV Variantes de la méthode des puissances itérées

IV.1 Utilisation de la projection de Raleigh-Ritz

Le principe ici est d'améliorer la méthode en utilisant la projection de Raleigh-Ritz, par cette méthode, on va calculer les valeurs propres d'une autre matrice, de dimensions bien plus petites et ainsi réduire de beaucoup les calculs.

Projection de Rayleigh Ritz
Comme auparavant on possède une matrice A, et une autre V de dimension n dont les vecteurs sont orthogonaux. On calcule ensuite $H = V^T AV$, en appliquant la décomposition spectrale.
Nous obtenons directement les valeurs propres de H qui permettent d'obtenir celles de A mais il est toujours nécessaire de calculer le produit HX pour les vecteurs propres associés.

On considère en entrée :

- une matrice carré A de dimension n
- une famille de vecteurs orthonormée V

Algorithme :

```
H = V'*A*V ;  – calcul du quotient de Raleigh
w = svd(H) ;    – décomposition en valeurs singulières de H et acquisition de ses valeurs propres dans w
V = VH ;        – calcul des vecteurs propres de A
retourner(V, w)
```

Amélioration de l'algorithme
En réalité la matrice A est symétrique puisque c'est une matrice de variance-covariance, on calcule après la matrice V comme dans l'algorithme des puissances itérées basique, puis après on génère la variance exprimée et on calcul V / V=AV, puis on orthogonalise les colonnes de cette matrice, et on applique le théorème de Rayleigh sur A et V.

On considère en entrée :

- *Raleigh(A, V)* est la fonction appliquant la projection de Raleigh-Ritz à A et V
- une matrice carré A de dimension n
- ε écart maximum pour le critère d'arrêt (par ex. 10^{-5})
- imax nombre maximum d'itération (par ex. 10000)
- threshold le pourcentage de variance expliquée souhaitée
- V famille de vecteurs orthonormée de dimension n*m

Algorithme :
Output : m dominant eigenvectors Vout and the corresponding LAMBDAout

1. initialisation :

```
i = 0 ;
percentReached = 0 ;
```

2. Boucle de calcul :

Tant que i < max et percentReached < threshold

```
V = A*V ;
V = ortho(V) ;      – orthonormalisation de V (par le procédé de Gramm-Schmidt par exemple)
[V w] = Raleigh(A, V);  – application de la projection à A et V pour obtenir les paires propres
```

– on test la convergence des vecteurs propres et on conserve ceux qui ont convergé

--calcul du critère de convergence pour chaque vecteur

```
pour i de 1 à m faire
      res_ev(i)=norm(A*V(:,i)-w(i)*V(:,i))/norm2(A,n,n) ;
fin pour
```

```
      i_conv = 1 ;                – indice du vecteur dont on vérifie la convergence
      n_ev = 0 ;          – nombre de vecteurs qui ont convergés
      Tant que i_conv <= m
            si res_ev(i) >  ε alors stop ;
            sinon si (res_ev(i) <  ε et i_conv > n_ev alors n_ev++ ;
            fin si
            i=i+1
      Tant que
      i = i+1 ;
fin Tant que
```

IV.2 Approche par blocs

Il existe une manière très simple d'améliorer la rapidité de l'algorithme précédent qui consiste à simplement remplace V=AV par $V=A^pV$.