Université de Strasbourg
Cours Calcul Scientifique 2

TP 3 : Méthode de Lanczos

L'objectif de ce TP est de programmer la méthode de Lanczos pour déterminer les valeurs propres d'une matrice symétrique. Commencez par importer les modules python :

```
import numpy as np
import scipy as sp
import scipy.sparse as spsp
import scipy.sparse.linalg as spsplin
```

Partie 1. (Lanczos)

1. Programmer une fonction `iter_Arnoldi_sym( A, v, vold, beta)`, qui effectue une itération de l'algorithme d'Arnoldi dans le cas symétrique. A partir des vecteurs `v, vold` $\in \mathbb{R}^n$ (correspondant à v_p et v_{p-1}) et `beta` $\in \mathbb{R}$ (correspondant à β_{p-1}), la fonction modifie les vecteurs `v, vold` pour qu'ils deviennent égaux à v_{p+1} et v_p et renvoie les coefficients `beta` et `alpha` correspondant à β_p et α_p. Nous rappelons l'algorithme de Lanczos :

$$
\begin{array}{l}
\text{Donnée } v_0 \in \mathbb{R}^n \\
v_0 = v_0 / \|v_0\| \\
\beta_{-1} = 0, v_{-1} = 0 \\
\text{Pour } p = 0, \ldots, N-1 \\
\quad w_p = A v_p \\
\quad \alpha_p = (w_p, v_p) \\
\quad w_p = w_p - \alpha_p v_p - \beta_{p-1} v_{p-1} \\
\quad \beta_p = \|w_p\| \\
\quad v_{p+1} = w_p / \beta_p
\end{array}
$$

2. Programmer une fonction `Lanczos(A, nbiter)` qui, à partir d'une matrice $A \in M_n(\mathbb{R})$ et d'un entier `nbiter` renvoie un tableau `eigval` de taille (`nbiter`,n) dont chaque ligne contient les valeurs de Ritz, **triées par ordre croissant**, obtenues par l'algorithme de Lanczos.
Nous rappelons ci-dessous l'algorithme:

$$
\begin{cases}
v_0 \text{ aléaloire} \\
v_0 = v_0 / \|v_0\| \\
\text{Tant que condition non satisfaite} \\
\quad \text{Déterminer } \alpha_p, \beta_p \\
\quad \text{Assembler } T_p \\
\quad \text{Calculer les valeurs propres de } T_p
\end{cases}
\qquad \text{avec } T_p = \begin{bmatrix}
\alpha_0 & \beta_0 & & \\
\beta_0 & \ddots & \ddots & \\
 & \ddots & & \beta_{p-1} \\
 & & \beta_{p-1} & \alpha_p
\end{bmatrix}
$$

Vous pourrez utiliser les fonctions `np.linalg.eig` pour déterminer les valeurs propres des matrices T_p.

3. Tester votre programme sur la matrice suivante :

```
A = spsp.diags([[4.]*n,[-1]*(n-1),[-1]*(n-1),[-1]*(n-d),[-1]*(n-d)], [0,1,-1,d,-d])
```

avec $n = d^2$ et $d = 10$. Prendre `nbiter = 40`. Vérifier la convergence de la valeur de Ritz la plus grande vers la plus grande valeur propre (tracer l'erreur en échelle logarithmique). On utilisera pour valeurs exactes les valeurs propres obtenues à l'aide de la fonction `np.linalg.eig`. Ajouter la

convergence de la plus petite valeur propre de Ritz vers la plus petite valeur propre. Donner une estimation du conditionnement de la matrice.

Remarque : les plus grandes valeurs propres peuvent également être obtenues à l'aide de la fonction de `scipy.sparse. linalg.eigsh`.

4. (Facultatif) Afficher l'évolution de toutes les valeurs de Ritz sachant que, pour cette matrice, la p-ième itération de l'algorithme fournit des approximations des $(p+1)/2$ plus grandes valeurs propres et des $(p+1)/2$ plus petites valeurs propres (si p est impair).
5. Considérer la matrice

```
B = spsp.diags(L, 0, dtype=float64)
```

avec $L = [0,0.01,0.02,\ldots,1.99,2,2.5,3]$ de taille 203. Afficher au cours des itérations les deux valeurs de Ritz les plus grandes. Le phénomène est appelé valeurs propres fantômes. Commentez.

Partie 2. (Méthode QR)

6. Programmer une fonction `facto_QR_hessenberg( A)` qui, à partir d'une matrice $A \in M_n(\mathbb{R})$ de type Hessenberg renvoie les matrices `Q` $\in O_n(\mathbb{R})$ et `R` $\in M_n(\mathbb{R})$ triangulaire inférieure telles que $A = QR$ en utilisant la méthode de Givens. Nous rappelons que la k-ième étape consiste à annuler le coefficient $A[k+1,k]$ en multipliant (à gauche) par une matrice de Givens

$$G_k = \begin{bmatrix} 1 & & & \\ & c & -s & \\ & s & c & \\ & & & 1 \end{bmatrix}$$

avec c et s vérifiant $c^2+s^2 = 1$ et $sR[k,k]+cR[k+1,k] = 0$, soit $c = \pm R[k,k]/\sqrt{R[k,k]^2 + R[k+1,k]^2}$ et $s = \mp R[k+1,k]/\sqrt{R[k,k]^2 + R[k+1,k]^2}$. A noter que la multiplication par la matrice G_k consiste juste à faire une rotation sur les k et $k+1$-ième lignes. Pour obtenir $Q = (G_{n-1} \cdots G_1)^T$, il faudra faire les mêmes multiplications sur la matrice identité.

7. Testez votre fonction sur une matrice aléatoire de petite taille. Vous pouvez utiliser la commande `np.triu` pour construire une matrice de Hessenberg.
8. Programmer une fonction `QR_hessenberg( A)` qui à partir d'une matrice $A \in M_n(\mathbb{R})$ de type Hessenberg renvoie une approximation des n valeurs propres de A par la méthode QR. Pour critère de convergence, vous pourrez utiliser le maximum des valeurs absolues des coefficients de la sous-diagonale -1 ainsi qu'un nombre maximal d'itération (it $\leqslant 2000$ par exemple).
9. Testez votre fonction sur une matrice aléatoire, symétrique tridiagonale, de petite taille et vérifier votre résultat avec la fonction `np.linalg.eig`. Que se passe-t-il pour une matrice de Hessenberg quelconque ? Commentez.
10. Dans l'algorithme de Lanczos, remplacer la fonction `np.linalg.eig` par votre fonction `QR_hessenberg`. Vérifier que le calcul des valeurs propres fonctionne encore. Comparer les temps de calcul.