DIF Clément
LOXOL Nicolas

Compte rendu TP : Calcul des Valeurs propres

1. Rappel des méthodes :

Avant de rentrer directement dans l'explication des méthodes ; il est important de rappeler certaines définitions :

Valeur propre : Soit λ une valeur réelle ; alors λ est valeur propre de f un endomorphisme de E dans E s'il existe u appartenant à E non nul tel que f(u) = λu.

Vecteur propre : Soit u non appartenant à E, E non vide. Alors u est un vecteur propre de l'endomorphisme f s'il existe λ une valeur réelle tel que f(u) = λu.

Polynôme caractéristique : Un polynôme caractéristique d'un endomorphisme f : E ⟶ E est un polynôme appartenant à $R_n[X]$, noté et défini par $C_f(X) = \det(\mathcal{M}_{\mathcal{B}}(f) - X\mathrm{Id}_E)$ où $\mathcal{M}_{\mathcal{B}}(f)$ est la matrice de f dans une base $\mathcal{B}$ de E quelconque.

Trace d'une matrice :

$$Tr(A) = \sum_{i=1}^{n} a_{ii} = \sum_{i=1}^{n} \lambda_i.$$

Remarques :

- Il est très facile de passer d'un endomorphisme a une matrice et réciproquement.
- $R_n[X]$ est l'ensemble des polynômes de degré inférieur ou égal à n.
- Id_E est l'endomorphisme identité sur l'espace vectoriel E.

Méthode de Leverrier basique :

Soit $C_A(\lambda)$ le polynôme caractéristique associé à la matrice A.

On a $C_A(\lambda) = |A - \lambda I_n| = a_n + a_{n-1}\lambda + a_{n-2}\lambda^2 + \cdots + a_0\lambda^n$

Et on note $S_p = \mathrm{Tr}(A^p) = \sum_{i=1}^{n} a_{ii}^p = \sum_{i=1}^{n} \lambda_i^p$

La méthode de Leverrier permet de déterminer les coefficients du polynôme caractéristique précédemment défini.

D'après la propriété suivante issue du cours, $A^p x = \lambda^p x \ \forall p \in \mathbb{N} \ \forall x \in \mathbb{R}^n$, les identités de Newton nous permettent de relier les traces des différentes puissances de A aux coefficients du polynôme caractéristique de la manière suivante :

$$\begin{cases} a_0 &= (-1)^n \\ a_1 &= -a_0 S_1 \\ 2a_2 &= -(a_0 S_2 + a_1 S_1) \\ 3a_3 &= -(a_0 S_3 + a_1 S_2 + a_2 S_1) \\ \dots &\dots &\dots \\ pa_p &= -(a_0 S_p + a_1 S_{p-1} + \dots + a_p S_1) \\ \dots &\dots &\dots \\ na_n &= -(a_0 S_n + a_1 S_{n-1} + \dots + a_n S_1) \end{cases}$$

On obtient alors le résultat exact de chaque coefficient a_i de $C_A(\lambda)$ $\forall i = 0, \dots, n$.

Remarque :

Cette méthode de Leverrier basique sera comparée en termes d'efficacité et de stockage, dans le programme, avec une méthode de Leverrier améliorée.

Méthode des puissances itérées :

Soit une matrice carrée A.

Cette méthode permet de calculer la plus grande valeur propre en valeur absolue du spectre (ensemble des valeurs propres) de la matrice A.

On suppose que toutes les valeurs propres sont distinctes. Des vecteurs propres x_i associés respectivement aux valeurs propres λ_i sont alors linéairement indépendants.

Un vecteur $v \in \mathbb{R}^n$ peut ainsi s'écrire

$$v = \sum_{i=1}^{n} \alpha_i x^i$$

Si on multiplie m fois cette égalité par A, on obtient :

$$A^m v = \sum_{i=1}^{n} \alpha_i \lambda_i^m x^i$$

Si le spectre de A est tel que $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$ alors pour m très grand, le rapport $\frac{\lambda_i^m}{\lambda_1^m}$ tend vers 0 pour i = 2, ..., n.

Notons $v_{m+1} = A^m v$, alors d'après ce qui précède, lorsque m est grand, $v_{m+1} = A^m v$ tend vers $\alpha_1 x^1 \lambda^m_1$.

Par conséquent, le rapport des deux vecteurs successifs $\frac{v_{m+1,i}}{v_{m,i}}$ tend vers λ_1, $\forall i = 1, ..., n$.

On en déduit le processus itératif suivant :

- Choisir un vecteur v_1 initial.
- Pour $k \geq 1$, calculer $v_{k+1} = A^k v_1 = A v_k$
- Arrêter lorsque que l'on obtient : $\frac{v_{m+1,i}}{v_{m,i}} \approx \frac{v_{m+1,j}}{v_{m,j}}$ pour toute paire de composantes i et j de ces vecteurs.

Un vecteur propre x_1 associé à la valeur propre λ_1 peut être calculé en utilisant l'équation suivante : $Ax_1 = \lambda_1 x_1$

2. Présentation du programme :

Notre programme est constitué de plusieurs parties distinctes :

- Les « *includes* » afin d'utiliser les fonctions de la bibliothèque standard.
- Une structure « optimisation » et un typedef qui nous servira à comparer l'efficacité des méthodes Leverrier et évaluer la complexité temporelle et spatiale de chaque méthode en fonction de différents facteurs comme l'ordre de la matrice, la nature de la matrice, etc.
- Les prototypes repartis en plusieurs catégories : les méthodes définies précédemment, les opérations appliquées aux matrices, les fonctions d'allocation dynamique, les fonctions de génération de matrice et une fonction annexe.
- La fonction main ou l'on injecte les jeux d'essais c'est-à-dire la matrice, la taille de celle-ci, la précision ainsi que la méthode de calcul souhaitée.
- Les définitions des fonctions présentées dans les prototypes :
    - *methode_leverrier_base* → Applique la méthode de Leverrier de base
    - *methode_leverrier_amelioree* → Applique la méthode de Leverrier améliorée
    - *methode_puissance* → Applique la méthode des puissances
    - *generer_identite* → Génère la matrice identité
    - *copier_matrice* → Copie une matrice vers une autre
    - *puissance_matrice* → Calcule la puissance d'une matrice
    - *multiplier_matrices* → Multiplie deux matrices
    - *multiplier_matrice_scalaire* → Multiplie un matrice par un scalaire
    - *additionner_matrices* → Additionne deux matrices
    - *calcule_norme* → Calcule la norme d'un vecteur
    - *calcule_trace* → Calcule la trace d'une matrice
    - *allouer_memoire_matrice* → Allocation dynamique d'une matrice
    - *liberer_memoire_matrice* → Libération de la mémoire allouée pour une matrice
    - *afficher_matrice* → Affiche une matrice
    - *generer_matrice_creuse* → Génère un matrice creuse
    - *generer_matrice_bord* → Génère un matrice de Bord

generer_matrice_ding_dong → Génère un matrice de Ding Dong
generer_matrice_de_franc → Génère un matrice de Franc
generer_matrice_de_hilbert → Génère un matrice de Hilbert
generer_matrice_kms → Génère un matrice de kms
generer_matrice_de_lotkin → Génère un matrice de Lotkin
generer_matrice_de_moler → Génère un matrice de Moler
min → Calcul du minimum entre deux entiers

3. Présentation des jeux d'essais :

Nous avons choisi d'utiliser les matrices du TP1 comme jeux d'essais (creuse, de Bord, de Ding Dong, de Franc, de Hilbert, kms, de Lotkin et de Moler).

Pour la méthode de Leverrier, nous ferons varier la nature de la matrice ainsi que sa taille.

Pour la méthode des puissances, nous rajouterons comme paramètre à faire varier, la précision c'est-à-dire l'ordre de grandeur de la différence entre deux itérations.

Remarque :

Ce que l'on appelle « nature de la matrice » est en fait le type de cette dernière parmi ceux énoncés plus haut (kms, creuse, de Bord, etc.).

4. Commentaires des jeux d'essais :

Taille	Leverrier		Leverrier améliorée	
	Nombre de clocks processeur	Nombre d'octets alloués	Nombre de clocks processeur	Nombre d'octets alloués
2	4	124	3	172
3	6	156	3	220
4	11	188	5	268
50	12 762 062	1 660	31 325	2 476

Taille	Méthode des puissances	
	Nombre de clocks processeur	Nombre d'octets alloués
2	23	48
3	29	56
4	31	64
50	1 991	432
Précision	Méthode des puissances	
	Nombre de clocks processeur	Nombre d'octets alloués
10^{-1}	26	56
10^{-2}	28	56
10^{-3}	30	56
10^{-10}	33	56

On constate que la méthode de Leverrier améliorée est plus efficace que la méthode de Leverrier classique. En effet, le temps nécessaire à la résolution est bien plus élevé pour cette dernière et cela est d'autant plus vrai lorsque la taille de la matrice augmente. Le nombre d'octets nécessaires est, quant à lui, environ 1,5 fois plus élevé pour la méthode améliorée.

Concernant la méthode des puissances itérées, lorsque la taille de la matrice augmente, les clocks et les octets nécessaires augmentent en proportion. Cependant lorsque la précision augmente d'une manière notable, le temps de résolution augmente que très peu et les octets nécessaires au fonctionnement de la méthode restent exactement identiques.

Lors des tests, nous avons également remarqué que la nature des matrices n'influe pas sur le temps d'exécution ni sur le nombre d'octets nécessaires pour la résolution des méthodes de Leverrier et de la méthode des puissances.