DIF Clément

LOXOL Nicolas

Compte rendu TP : Calcul des Valeurs propres

1. Rappel des méthodes :

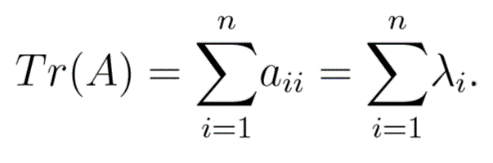
Avant de rentrer directement dans l’explication des méthodes ; il est important de rappeler certaines définitions :

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 appartenant à E, E non nul. 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 est un polynôme P appartenant à Rn[X] est le polynôme noté et définie par Cf(X) = det (f - XIdE)

Trace d’une matrice :



Remarque :

-Il est très facile de passer d’un endomorphisme a une matrice et réciproquement

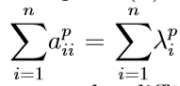
- Rn[X] est l’ensemble de polynôme de degré inferieur ou égale à n

- IdE est l’identité sur l’espace vectoriel E

Méthode de Leverrier :

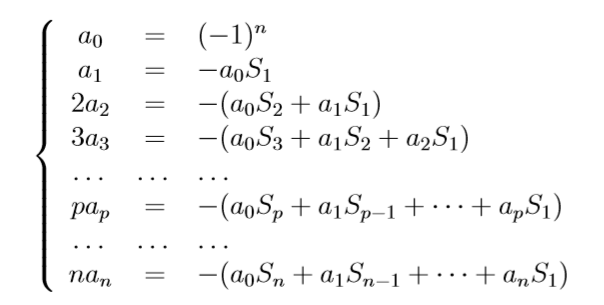
Soit CA(X) le polynôme caractéristique associé à la matrice A.

On a CA(X) = = |A − λIn| = an + an−1λ + an−2λ2 + ··· + a0λn

Et on note Sp = Tr(Ap) = 

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

D’après le propriété Apx = λpx, ∀p entier, les identités de Newton nous permettent de relier les traces des différentes puissances de A aux coefficients du polynôme caractéristique de la manière suivante :



On obtient alors le résultat exact de chaque coefficient ai de CA(X).

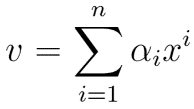
Méthode des puissances itérées :

Soit une matrice carré A :

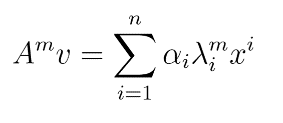
Cette méthode permet de calculer la plus grande valeur propre en valeur absolue du spectre (ensemble des valeurs propres de A) de la matrice A.

On suppose que toutes les valeurs propres sont distinctes. Les vecteurs propres xi sont alors linéairement indépendants.

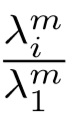
Un vecteur v ∈ Rn peut ainsi peut ainsi s’écrire



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

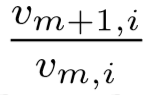


Si le spectre de A est tel que |λ1| > |λ2| > ··· > |λn| alors pour m très grand, le rapport



tend vers 0 pour i = 2,..,n.

Notons vm+1 = Amv, alors d'après ce qui précède, lorsque m est grand, vm+1 = Amv tend vers α1x1λm1. Par conséquent, le rapport de deux vecteurs successifs



tend vers λ1,∀i = 1,...,n.

On en déduit le processus itératif suivant :

* Choisir un vecteur v1 initial.
* Pour k ≥ 1, calculer vk+1 = Akv1 = Avk
* Arrêter lorsque le rapport



pour toute paire de composantes i et j de ces vecteurs.

Le vecteur propre x1 associé à la valeur propre λ1 peut être calculé en utilisant l'équation suivante : Ax1 = λ1x1

1. 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.
* Les prototypes repartie en plusieurs catégorie : Les méthodes de calcul des valeurs propres , 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 souhaité.
* 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 🡪* Copier une matrice

*puissance\_matrice 🡪* Calcul de puissance sur une matrice

*multiplier\_matrices 🡪* Multiplie une matrice

*multiplier\_matrice\_scalaire 🡪* Multiplie un matrice par un scalaire

*additionner\_matrices 🡪* Additionne deux matrices

*calcule\_norme 🡪* Calcule la norme d’une matrice

*calcule\_trace 🡪* Calcule la trace d’une matrice

*allouer\_memoire\_matrice 🡪* Allocation dynamique des matrices

*liberer\_memoire\_matrice 🡪* Libération de la mémoire allouée

*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

1. Jeux d’essais :

Nous avons choisi les matrices du TP1 comme jeux d’essai (creuse, bord, ding dong, franc, hilbert, kms, lotkin, 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.

1. Commentaires des jeux d’essais :

|  |  |  |
| --- | --- | --- |
| Taille | Leverrier (clock/Octet) | Leverrier Amélioré (clock/Octet) |
| 2 | 4/124 | 3/172 |
| 3 | 6/156 | 3/220 |
| 4 | 11/188 | 5/268 |
| 50 | 12762062/1660 | 31325/2476 |

|  |  |
| --- | --- |
| Taille | Méthode des puissances (clock/Octet) |
| 2 | 23/48 |
| 3 | 29/56 |
| 4 | 31/64 |
| 50 | 1991/432 |
| Précision |  |
| 0,1 | 26/56 |
| 0,01 | 28/56 |
| 0,001 | 30/56 |
| 0,0000000001 | 33/56 |

On constate que la méthode de Leverrier amélioré est plus efficace que la méthode de Leverrier classique. En effet le temps nécessaire à la résolution est plus faible dans Leverrier améliorer. Cela est d’autant plus vrai quand la taille de la matrice augmente. Les octets nécessaires quant a eu sont à peu près similaire selon la méthode.

Pour la méthode des puissances itérées lorsque la taille de la matrice augmente les clocks et les octets nécessaire augmente ce qui semble normal. Cependant lorsque la précision augmente le temps de résolution augmente peu alors que les octets nécessaires au fonctionnement de la méthode restent identiques.

La nature des matrices n’influe pas sur le temps d’exécution ni sur le nombre d’octet nécessaire pour la résolution de la méthode.