PROJET D'ANALYSE NUMÉRIQUE

MÉTHODE QR, TRANSFORMATION DE HOUSEHOLDER

Manon Rose et Roxane Leduc

Institut National Des Sciences Appliquées de Rouen-Normandie

Décembre 2021

Table des matières

1 Présentation générale

Ce projet a été réalisé en Fortran et en Matlab. Nous avions pour objectif de déterminer les valeurs propres et vecteurs propres d'une matrice symétrique. Notre programme est divisé en trois étapes. La première étape consiste à réduire la matrice carrée symétrique initiale à une forme tridiagonale. Ensuite, on calcule d'abord les valeurs propres de cette matrice et enfin on recherche les vecteurs propres.
Pour la tridiagonalisation de la matrice carrée symétrique, nous avons implémenté les méthodes de transformations de Householder expliquées au cours du rapport.
Afin de déterminer les valeurs propres de la matrice A, nous allons itérer sur le processus qui permet d'obtenir une factorisation QR, pour, au final, trouver les valeurs propres cherchées (en repartant de la matrice T tridiagonale, semblable à A). Les vecteurs propres correspondants pourront être calculés par itérations inverses.

Nous avons testé nos algorithmes sur des matrices de Hilbert, ainsi que sur la matrice A :

$$A = \begin{pmatrix} 4 & 1 & -2 & 2 \\ 1 & 2 & 0 & 1 \\ -2 & 0 & 3 & -2 \\ 2 & 1 & -2 & -1 \end{pmatrix}$$

Dans une dernière partie, nous allons comparer nos résultats en Matlab et Fortran. Nous conclurons sur les avantages d'une telle méthode à l'aide de nos matrices tests.

2 Détails des étapes

2.1 Première étape : tridiagonalisation symétrique

L'algorithme de Householder tel que nous l'avons vu dans le cours d'analyse numérique et que nous décrivons dans la deuxième étape ne permet pas d'obtenir une factorisation avec matrice tridiagonale symétrique. L'idée va donc être ici de modifier quelque peu l'algorithme afin d'obtenir une matrice T tridiagonale, semblable à A (ayant les mêmes valeurs propres).

Nous avons pu lire que les matrices tridiagonales étaient le point de départ de nombreux algorithmes de calculs de valeurs propres, car les entrées nulles réduisent la complexité du problème.

Cette procédure de tridiagonalisation est présentée en analyse numérique par Burden et Faires. Elle utilise une fonction signe que nous avons du coder puisqu'elle est quelque peu modifiée (sgn(0)=1). Afin de former la matrice de Householder à chaque étape, nous allons devoir déterminer les α et r, tels que :

$\alpha = -sgn(t^k_{k+1,k}) * ||t^k_{j,k}|| \ \ \forall k = 2, 3, ..., n-2$
et
$r = (0,5 * (\alpha^2 - t^k_{k+1,k}\alpha))^{0,5} \ \ \forall k = 2, 3, ..., n-2$

Puis on itère, en calculant :

$t^k_{k+1,k} = (\frac{t^k_{k+1,k} - \alpha}{2*r})$

Pour enfin tridiagonaliser la matrice T, on calcule :
Q = Q * H avec H la matrice de Householder décrite plus bas.
$T = H^t * T * H$
On obtient alors une matrice T tridiagonale.

2.2 Deuxième étape : factorisation QR et calcul des valeurs propres

Une fois la matrice T (tridiagonale) obtenue, semblable à A, l'idée va être de :
-poser T(0)=T
-décomposer T(0)=QR
-construire T(1)=RQ
-puis itérer (Q,R)=decomposeQR(T(k)) et T(k+1)=R*Q.
La suite diag(T(k)) converge alors vers les valeurs propres de A.

Nous allons donc maintenant voir en détail le processus de factorisation QR.

Dans le processus de factorisation QR, Q est une matrice orthogonale ($QQ^t = I$) et R une matrice triangulaire supérieure. Pour appliquer cette méthode, nous avons besoin du théorème de décomposition QR :
Théorème : Soit A une matrice inversible, elle admet une décomposition QR, telle que A = QR.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & \dots & \dots & a_{1n} \\ \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & & \ddots & \vdots \\ \vdots & & & & \ddots & \vdots \\ a_{n1} & \dots & \dots & \dots & \dots & \dots & a_{nn} \end{pmatrix}$$

Nous allons donc construire cette factorisation en utilisant les matrices de transformations de Householder.

Dans notre cours d'analyse numérique, nous avons défini la méthode de Householder telle que :

Soit $v \in \mathbb{R}^n$
On définit la matrice élémentaire de Householder, symétrique et orthogonale $H(v)$:

$$H(v) = I - 2\left(\frac{vv^t}{||v||^2}\right)$$

avec I la matrice identité, v un vecteur et ||.|| la norme euclidienne.

Pour obtenir la décomposition QR que nous attendons, nous devons multiplier la matrice A initiale par des matrices de Householder.

Proposition : pour tout vecteur u $\in \mathbb{R}^n$, il existe v$\in \mathbb{R}^n$ tel que $H(v)u = ||u||e_1$ où e_1=(1,0,...,0)$\in \mathbb{R}^n$.

Ainsi on pose : $v = u - ||u||e_1$ si u n'est pas co-linéaire à e_1, sinon $v = 0$.

On peut donc calculer : $A = \begin{pmatrix} a_{11} & a_{12} & \dots & \dots & \dots & a_{1n} \\ \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & & \ddots & \vdots \\ \vdots & & & & \ddots & \vdots \\ a_{n1} \dots & \dots & \dots & \dots & \dots & \dots a_{nn} \end{pmatrix} .H^{(1)}(u_1 - ||u_1||e_1)$

avec A symétrique et $u_1 = (a_{11}, ..., a_{n1})^t$

Puis on réitère,

$A^{(2)} = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & \dots & \dots & a_{1n}^{(2)} \\ 0 & & & & & \vdots \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & & \ddots & \vdots \\ \vdots & & & & \ddots & \vdots \\ 0 \dots & \dots & \dots & \dots & \dots & \dots a_{nn}^{(2)} \end{pmatrix} .H^{(2)}(u_2 - ||u_2||e_2)$

avec $u_2 = (0, a_{22}^{(2)}, ..., a_{n2}^{(2)})^t$

Et ainsi de suite, à l'étape k, on construit une matrice $A^{(k)}$ comme $A^{(k)} = H^{(k-1)}H^{(k-2)}...H^{(1)}A$
avec $H^{(k)} = H(u_k - ||u_k||e_k)$, $u_k^{(k)} = (0, ..., a_{kk}^{(k)}, ..., a_{nk}^{(k)})^t$ et

$$A^{(k)} = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & \dots & \dots & a_{1n}^{(k)} \\ 0 & a_{22}^{(k)} & & & & \vdots \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & 0 & a_{kk}^{(k)} & \vdots \\ \vdots & & & & & \ddots \\ 0 & \dots & \dots & 0 & a_{nk}^{(k)} & \dots a_{nn}^{(k)} \end{pmatrix}$$

On en déduit alors :

$A^{(n)} = H^{(n-1)}H^{(n-2)}...H^{(1)}A = R$ où R est bien triangulaire supérieure.
et $Q = (H^{(1)})^t...(H^{(n-2)})^t(H^{(n-1)})^t = H^{(1)}...H^{(n-2)}H^{(n-1)}$ où Q est bien orthogonale.

On obtient donc bien finalement : A = QR.
Un des avantages de cette méthode est sa stabilité numérique.

En réalisant les étapes décrites plus tôt, on parvient à calculer plus facilement les valeurs propres de la matrice T tridiagonale qui est semblable à A, et donc qui possède les mêmes valeurs propres.

2.3 Troisième étape : calcul des vecteurs propres et séquences de Sturm

Nous avons pu lire au cours de nos recherches que : lorsqu'une valeur propre précise d'une matrice tridiagonale symétrique réelle est connue, le vecteur propre correspondant peut être calculé par itération inverse (ce qui constitue peut-être la méthode la plus efficace pour déterminer les vecteurs propres correspondant à un nombre relativement faible de valeurs propres sélectionnées).

Le "Accurate ordering of eigenvectors and singular vectors without eigenvalues and singular values" fournit en page 15 (section 7) un algorithme permettant le calcul de ces vecteurs propres.

D'après le lemme 9, le i-ième vecteur propre est donné par :

$$V(i) = (-1)^{i-1} * p_{i-1} * \left(\prod_{k=1}^{i-1} b_k \right)^{-1}, i = 2...n$$

Avec : $p_1 = t_{1,1} - \lambda$
Et : $p_i = (t_{i,i} - \lambda) * p_{i-1}(\lambda) - t_{i,i-1}^2 * p_{i-2}(\lambda), i = 2...n$

Remarque : "t" désigne les coefficients de la matrice tridiagonale symétrique T.

Enfin, on détermine les vecteurs propres de A en multipliant Q par le vecteur V.

3 Analyse des résultats numériques

Dans cette partie, nous allons tester notre programme sur plusieurs matrices, une matrice symétrique bien conditionnée A :

$$A = \begin{pmatrix} 4 & 1 & -2 & 2 \\ 1 & 2 & 0 & 1 \\ -2 & 0 & 3 & -2 \\ 2 & 1 & -2 & -1 \end{pmatrix}$$

et des matrices de Hilbert. Ces dernières sont réputées pour être très mal conditionnées, ce qui rend leur usage délicat en analyse numérique. Nous allons analyser le nombre d'itérations nécessaires pour déterminer les valeurs propres de nos matrices. Dans notre programme, pour tester l'algorithme sur la matrice A, on choisit ichoix=0, et sur les matrices de Hilbert, ichoix=1.
On rappelle que T est semblable à A. On va donc récupérer les valeurs propres de nos matrices grâce aux décompositions successives QR effectuées sur T.

3.1 Analyse de la matrice A

Nous avons dans un premier temps analysé le nombre d'itérations nécessaires à l'obtention des valeurs propres pour la matrice A (donnée ci-dessus) en fonction du seuil de précision défini dans le code. Sur le graphe (cf. FIGURE 1), nous remarquons que la précision augmente très rapidement. Plus le seuil de précision est faible, plus le nombre d'itérations augmente. Nous avons imposé un seuil de précision à 1e-6 dans notre algorithme et nous avons pu le comparer avec des matrices de Hilbert.
Pour un seuil de précision à 1e-6, à l'exécution du programme Fortran, nous obtenons la FIGURE 2. Pour un seuil de précision à 1e-6, à l'exécution du programme Matlab, nous obtenons la FIGURE 3.

Nous remarquons que la matrice T est tridiagonale en considérant que 1e-16 = 0 (1e-16 est très petit devant 0 alors on peut le supposer). Nous obtenons bien les valeurs propres attendues après vérification en 221 itérations.

3.2 Analyse des matrices de Hilbert

Dans un second temps, nous avons analysé les matrices de Hilbert, qui contrairement à ce que l'on pourrait imaginer, fournissent des résultats très rapidement (au bout de 5 itérations pour une taille 4*4, cf. FIGURE 4). Le tableau ci dessous regroupe les informations concernant des matrices de Hilbert de tailles différentes. Nous avons fixé ici, un seuil de précision à 1e-6 pour comparer l'algorithme avec la matrice A :

Taille de la matrice	Nombre d'itérations	Valeurs propres
4	5	1.5002 0.1691 0.0067 0.0001
7	5	1.6609 0.2719 0.0213 0.0010 ...
10	6	1.7519 0.3429 0.0357 0.0025 ...

Nous remarquons ainsi que, à partir de notre programme, nous obtenons les valeurs propres des matrices de Hilbert en moins d'itérations que celles de la matrice A. Pour une matrice carrée 4*4, le programme nous donne les valeurs propres de la matrice A en beaucoup plus d'itérations (221) que celles de la matrice de Hilbert (5 itérations).

3.3 Comparaison Fortran et Matlab

Dans les deux langages les résultats sont les mêmes, le nombre d'itérations nécessaires à l'obtention des valeurs propres est équivalent. La principale difficulté du langage fortran réside dans l'allocation dynamique

des tableaux puisque cela alourdit rapidement le code. Matlab est plus simple d'utilisation (nous possédons notamment les matrices de Hilbert déjà codées).

3.4 Un résultat surprenant

Nous étions partis du postulat qu'en tridiagonalisant la matrice et en calculant les valeurs propres depuis la matrice T (semblable à A), nous allions gagner en temps de calcul. Cependant, force est de constater que le nombre d'itérations nécessaires pour le calcul des valeurs propres sur T est de 221, soit largement supérieur au nombre d'itérations nécessaires pour le calcul des valeurs propres en partant directement de A (148 itérations nécessaires pour le même seuil de 1e-6). En effet, puisque l'équation aux valeurs propres de la matrice tridiagonale revêt une forme simple, nous nous attendions à ce que les racines se calculent plus rapidement.

4 Conclusion

Ce projet nous a permis de bien comprendre la méthode de Householder vue en cours d'analyse numérique et même d'aller plus loin, puisque nous avons appris à tridiagonaliser une matrice symétrique à l'aide de la factorisation QR. Nous avons notamment pu découvrir que cette méthode était très efficace pour la matrice de Hilbert ce qui, du fait de son mauvais conditionnement, était très surprenant. Cette méthode, bien qu'utile puisqu'elle sert à l'obtention des valeurs et donc des vecteurs propres, nécessite cependant un grand nombre d'itérations (ce qui est coûteux). Ce projet a été très enrichissant et nous a permis de nous améliorer et d'apprendre à mieux manipuler les langages Matlab et Fortran mais aussi de mieux appréhender les avantages et les inconvénients de chacun.

5 Bibliographie

Références

[1] Un algorithme performant pour le calcul des valeurs propres et des vecteurs propres en analyse factorielle des correspondances. (A.Karakos)

[2] Wikipedia, Householder transformation

[3] Une introduction à la modélisation mathématique et à la simulation numérique (G.Allaire)

[4] http ://asi.insa-rouen.fr/enseignement/siteUV/ananum/08valeurpropres.pdf

[5] https ://core.ac.uk/download/pdf/82190245.pdf (Accurate ordering of eigenvectors and singular vectors without eigenvalues and singular values, K.V.Fernando)

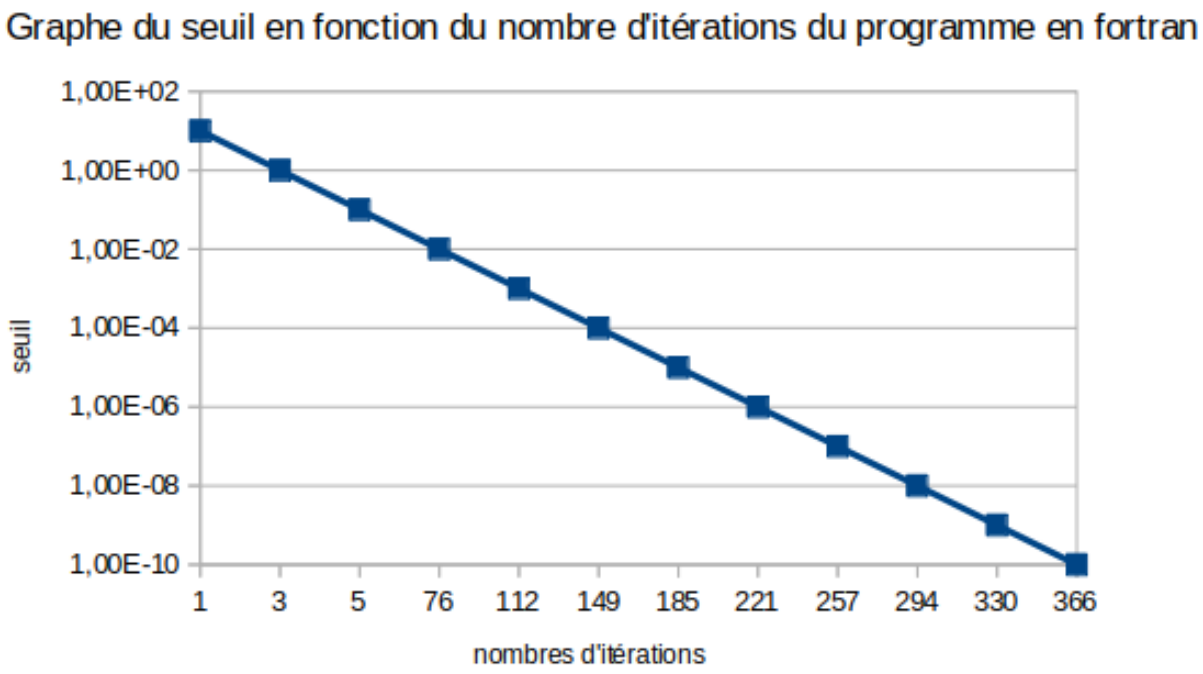

FIGURE 1 – Analyse sur la matrice A.

```
Phase 1 : Triangularisation
Matrice T:
  4.0000000000000000          -3.0000000000000009           1.3322676295501979E-016     9.3258734068513134E-016
 -3.0000000000000009           3.3333333333333357          -1.6666666666666672          3.3306690738754696E-016
  1.3322676295501979E-016     -1.6666666666666674          -1.3200000000000023         -0.90666666666666273
  9.3258734068513134E-016      0.0000000000000000          -0.90666666666666273          1.9866666666666697

Phase 2 : Valeurs propres de A
Nombres d iterations:          221
Valeurs propres:
  6.8446211072349685
  2.2685209021999899
 -2.1975064732081986
  1.0843644637732084
```

FIGURE 2 – Affichage à l'exécution du programme Fortan sur la matrice A.

6 Annexe

```
>> householder
===== PHASE 1: Tridiagonalisation =====
    4.0000   -3.0000         0         0
   -3.0000    3.3333   -1.6667         0
         0   -1.6667   -1.3200   -0.9067
         0         0   -0.9067    1.9867

On peut verifier a l aide de la fonction eig (pour l'instant), que A et T sont semblables:
Valeurs propres de A:
   -2.1975    1.0844    2.2685    6.8446

Valeurs propres de T:
    6.8446   -2.1975    1.0844    2.2685

===== PHASE 2: Valeurs propres de A =====
Nombre d iterations necessaires
   221

    6.8446    2.2685   -2.1975    1.0844

===== PHASE 3: Vecteurs propres de A =====
Vecteur propre :
    1.0000
  -15.8161
   -7.8027
    1.5277

Vecteur propre :
    1.0000
   -1.5286
   -0.3380
   -0.4394

Vecteur propre :
    1.0000
    5.9903
   -1.1164
   -7.2103

Vecteur propre :
    1.0000
   -3.5255
   -1.4144
   -1.1094
```

FIGURE 3 – Affichage à l'exécution du programme Matlab sur la matrice A.

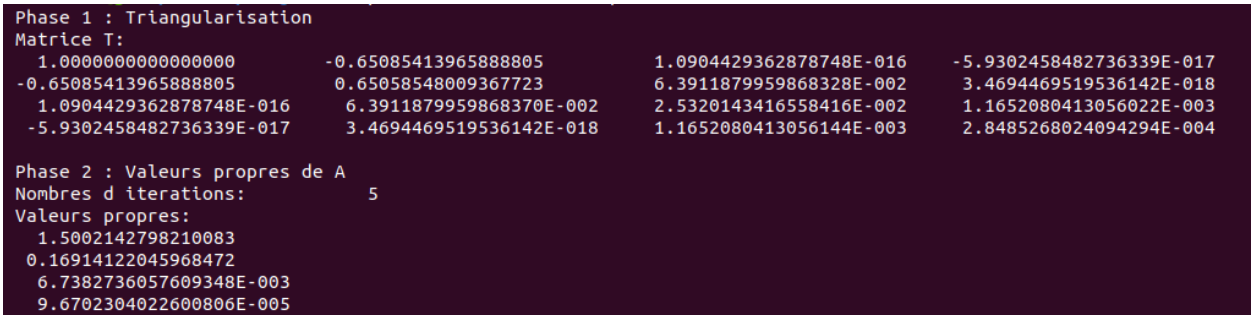

FIGURE 4 – Affichage à l'exécution du programme Fortran sur une matrice de Hilbert 4*4.