UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

UFR SEN
Sciences Exactes et Naturelles

RAPPORT DE PROJET.

MA0604
PAR

GNAGO YANNICK.

ENSEIGNANT: PIERRE MOLLO.

2020/2021

Partie 1/2

Application de la méthode de la puissance : l'algorithme du PageRank de Google.

Objectif :

Cette partie a pour but de mettre en application le cours d'analyse numérique sur la méthode de la puissance à travers un cas concret, celui d'un moteur de recherche "GOOGLE" qui fut créer en 1998, un projet de recherche de deux étudiants.
Nous allons essayer de mettre un algorithme de tri des pages web appelé "L'algorithme du PageRank".
Qu'est-ce que c'est un moteur de recherche ? [2]
Quand un utilisateur lance une requête (mots clés cherchés) sur internet. Le moteur de recherche fait :
$\Longrightarrow$ Une liste des pages contenant les mots clés
$\Longrightarrow$ Un tri des pages par ordre de pertinence.
$\Longrightarrow$ Un affichage des pages des résultats (sur $n = 10^{14}$ pages référencées en 2016)
Un utilisateur normal ne lira que les premières pages après une requête dans un moteur de recherche alors comment calculer la pertinence (PageRank) pour lister les pages relatives à la requête ?

Partie théorique :

Il y'a donc plusieurs étapes pour trouver le PageRank.

1. La matrice des liens et la matrice de transition
   –La matrice des liens(connexion) correspond aux pages web et leurs liens qu'on notera L définit par :

$$l_{ij} = \begin{cases} 1 & \text{la page i a un lien vers la page j.} \\ 0 & \text{sinon.} \end{cases} \quad (a)$$

   Par exemple :
   la page 1 a un lien vers la page 2
   la page 2 a un lien vers la page 3.
   la page 3 a un lien vers la page 2 et 4.
   la page 4 a un lien vers la page 5.
   la page 5 a un lien vers la page 1.
   Alors $L = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$.
   –La matrice de transition notée Q désigne la manière dont le moteur choisit les pages lors de la requête, au fait il choisit de manière aléatoire et équiprobable un des liens présents sur cette page. (Voir dans le script validat, partie Matrice Q)

$$q_{ij} = \frac{l_{ij}}{n_i} \quad avec \quad n_i = \sum_{k=1}^{n} l_{ik} \quad (b)$$

   Alors $Q = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$.
   Cette matrice de transition a des limites dans l'application car si l'utilisateur tombe sur 0, le processus de recherche du PageRank s'arrête, alors on modifiera Q par P de telle sorte que si l'utilisateur tombe sur une page qui n'a aucun lien ($n_i = 0$), alors le moteur choisit de façon aléatoire une nouvelle page parmi les autres pages de web.(Voir dans le script validat, partie Matrice P)

$$P = Q + \frac{1}{n} d e^t, \quad d \in \mathbb{R}^n \quad tel \quad que \quad d_i = \begin{cases} 0 \text{ si } n_i \neq 0. \\ 1 \quad \text{sinon.} \end{cases} \quad (c)$$

$$1$$

$et \quad e \in \mathbb{R}^n \quad$ défini $\quad par \quad e^t = (1, ..., 1) \in \mathbb{R}^n$.
Remarque : P est une matrice stochastique alors 1 est valeur propre et P^t vérifie alors l'une des hypothèses du théorème de Perron-Frobenius[5] vu dans l'énoncé, alors le vecteur propre associé à la valeur propre r (r positif et supérieur au module de autres valeurs propres) représente bien les indices de pertinence.
Mais un autre problème se pose encore : si P^t n'est pas nécessairement irréductible i.e. beaucoup de sites n'ont de lien entre eux (non fortement connexe) alors on va devoir modifier P^t.

2. La matrice de Google.
Comme l'utilisateur peut, à tout moment abandonner sa recherche pour recommencer avec une nouvelle page choisie au hasard et de façon équiprobable parmi les pages disponibles. On pose alors :

$$G = \alpha P + (1 - \alpha)\frac{1}{n}ee^t. \quad (d)$$

avec $\alpha \in [0, 1[$ proche de 1 car cela arrive rarement.

3. La méthode la puissance.
Nous considérons à nouveau G^t stochastique au lieu de P^t afin de trouver le vecteur propre associé à la valeur propre dominante 1 grâce à la méthode de la puissance.

Démonstration :
On a : G stochastique alors $\rho(G) = 1$ et $\rho(G) = \rho(G^t)$; de plus $G_{ij} > 0 \Leftrightarrow G^t_{ij} > 0$.
Donc le théorème de Perron-Frobenius[5] est vérifié.

La mise en œuvre : La méthode de la puissance pour le calcul du PageRank.

Nous essayerons de mettre en place un algorithme sur Octave de la méthode de la puissance pour le calcul du PageRank.
Rappelons que le vecteur du PageRank est un vecteur propre de G^t associé à la valeur propre $lambda = 1$ qui est la valeur propre dominante de G^t.

1. Méthode de la puissance pour une matrice stochastique S.
Cette fonction permet d'avoir la valeur propre dominante λ et son vecteur propre associé x à S^t.
Nous allons donc optimiser l'algorithme de la méthode puissance vue en TP car nous savons déjà que pour une matrice stochastique $\lambda = 1$ et la norme 1 du x initialisé doit être non nulle.

```
function [lambda,x,niter]=puissance1(S,x0,tol,kmax)

x=x0;
test=tol+1;
niter=0;
lambda=1;

while (niter<kmax && test>tol)
    z=S'*x;
    test=norm((z-x),1);
    x=z;
    niter=niter+1;
endwhile
```

FIGURE 1 – Fonction puissance1.

2. Validation.
Appliquons notre fonction puissance1 à la matrice G (cf. mini-réseaux des paragraphes 1 et 2).
On a :
La page 1 n'a pas de lien.
La page 2 a un lien vers la page 1 et 3.

La page 3 a un lien vers la page 1 et 2.
La page 4 a un lien vers la page 2 et 3.
La page 5 a un lien vers la page 2,3 et 4.

Alors $L = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$, $Q = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \end{pmatrix}$ et $P = \begin{pmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \end{pmatrix}$.
D'abord, on calcule G dans le script grâce à sa formule $G = \alpha P + (1-\alpha)\frac{1}{n}ee^t$ avec $\alpha = 0.85$ et $tol = 10^{-15}$ et ensuite on applique la fonction puissance1 à G. (Voir script valid).

```
%Validation de puissance1 avec le mini-reseaux du paragraphe 1 et 2

printf("La matrice de transition du réseau est:")
P=[1/5 1/5 1/5 1/5 1/5;1/2 0 1/2 0 0;1/2 1/2 0 0 0;0 1/2 1/2 0 0;0 1/3 1/3 1/3 0]

n=size(P,1);
e=ones(n,1);
alpha=0.85;
G=alpha*P+(1-alpha)*(1/n)*e*e';

x0=[1 0 0 0 0]';
tol=10^(-15);
kmax=100;

[lambda,x,niter]=puissance1(G,x0,kmax,tol);
printf("l'approximation de la valeur propre dominante est:")
lambda
printf("Le vecteur PageRank du reseaux est:")
x=sort(x,1,'descend')
[V,val_prop]=eigs(G');
printf("Le vecteur propre dominante de G' avec eigs est:")
V(:,1)
```

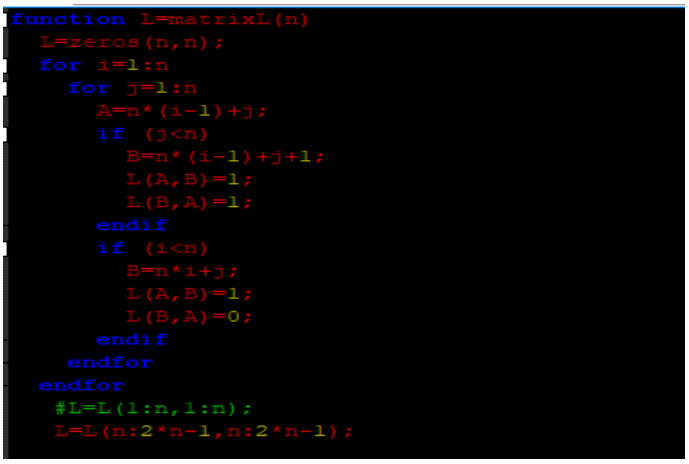
FIGURE 2 – Script de validation de puissance1.

Analyse : On remarque que le vecteur PageRank est bien ordonné par ordre décroissant et il est pareil à celui de la fonction *eigs* de G^t.

```
>> valid

La matrice de transition du réseau est:P =

    0.2000    0.2000    0.2000    0.2000    0.2000
    0.5000         0    0.5000         0         0
    0.5000    0.5000         0         0         0
         0    0.5000    0.5000         0         0
         0    0.3333    0.3333    0.3333         0

l'approximation de la valeur propre dominante est:lambda = 1.0000
Le vecteur PageRank du reseaux est:x =

    0.6122
    0.5258
    0.5258
    0.2122
    0.1653

Le vecteur propre dominante de G' avec eigs est:ans =

    0.6122
    0.5258
    0.5258
    0.2122
    0.1653
```

FIGURE 3 – Résultat de validation de puissance1 avec le mini-réseaux des paragraphes 1 et 2.

3. Passage en grande dimension.
   (a) La matrice de connexion L.
   On essayera de créer une fonction qui génère une matrice de connexion de taille n contenant beaucoup de 0 et quelques 1. Dans mon script, à la fin des boucles for L est de taille n^2, alors j'extrais de L une sous matrice de taille n grâce à $L = L(n : 2n-1, n : 2n-1)$ ou $L = L(1 : n, 1 : n)$.



```
function L=matrixL(n)
  L=zeros(n,n);
  for i=1:n
    for j=1:n
      A=n*(i-1)+j;
      if (j<n)
        B=n*(i-1)+j+1;
        L(A,B)=1;
        L(B,A)=1;
      endif
      if (i<n)
        B=n*i+j;
        L(A,B)=1;
        L(B,A)=0;
      endif
    endfor
  endfor
   #L=L(1:n,1:n);
   L=L(n:2*n-1,n:2*n-1);
```

FIGURE 4 – Fonction matrixL.

Exemple : Avec n=5 $\Rightarrow$L= $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$.

Alors : la page 1 *a un lien vers la page* 2.
la page 2 *a un lien vers la page* 1 *et* 3.
la page 3 *a un lien vers la page* 2 *et* 4.
la page 4 *a un lien vers la page* 3 *et* 5
la page 5 *a un lien vers la page* 4.

(b) Test de validation.
On appellera la fonction L grâce à matrixL pour un n (=5,20,30), ensuite on crée la matrice Q tel que défini au (b) puis on crée la matrice P tel définit au (c) ; enfin en gardant $\alpha = 0.85$, on calcule G (d) et on l'applique à la fonction puissance1 pour avoir le vecteur du PageRank associé au réseau.
(Voir script validat)
–Pour n=5 : le temps de calcul est : 0.013786 pour 202 itérations.
–Pour n=20 : le temps de calcul est : 0.011102 pour 171 itérations.
–Pour n=30 : le temps de calcul est : 0.013810 pour 176 itérations.
–Pour n=60 : le temps de calcul est : 0.016757 pour 175 itérations.

Conclusion :

L'objectif était de trouver le vecteur PageRank qui matérialise la pertinence des pages affichées lors d'une recherche sur le moteur de recherche GOOGLE .
Nous avons donc plusieurs étapes qui sont : la matrice de connexion L (Recherche des liens entre les pages web qui contiennent les mots clés), la matrice de transition stochastique G (Mise en place de la matrice du moteur de recherche) et la méthode de la puissance appliquée à G^t (le tri et affichage d'une liste ordonnée) ; comme G est une matrice stochastique, on se sert donc des propriétés d'une matrice stochastique et du théorème de Perron-Frobenius pour optimiser la méthode de puissance inverse afin de trouver le vecteur propre de la valeur propre dominante $\lambda = 1$.
Grâce au tests réalisés, nous pouvons déduire que la convergence de la méthode est assurée si le graphe est fortement connexe ce qui est rare dans la réalité car beaucoup de site n'ont pas de lien entre eux cela prouve bien l'utilité de l'algorithme PageRank lors d'une recherche sur le moteur de recherche GOOGLE.

Partie 2/2

Application des méthodes de déflation et QR :
Calcul des fréquences propres d'une membrane.

Objectif :

Nous essayerons de mettre en application la méthode de déflation et QR dans un cas pratique, celui de la membrane élastique tel que la fonction propre du Laplacien avec condition de Dirichlet toute fonction $u : [0,a] \times [0,b] \to \mathbb{R}$ non identiquement nulle pour laquelle il existe une valeur $\lambda \in \mathbb{R}$ valeur propre tel que :

$$-\Delta u(x,y) = \lambda u(x,y) \quad \forall (x,y) \in]0,a[\times]0,b[\quad (e)$$

et

$$\begin{cases} \text{u(x,0)=u(x,b)=0 } \forall x \in [0,a] \\ \text{u(0,y)=u(a,y)=0 } \forall y \in [0,b] \end{cases} \quad (f)$$

Avec $u(x,y)$, la fonction propre du déplacement vertical du point (x,y) d'une membrane élastique qui vibre à la fréquence propre λ.

Partie théorique : Discrétisation et études.

1. Montrons que A admet n valeurs propres réelles strictement positives.
   –A est-elle symétrique ?

   On a : $A = \begin{pmatrix} G & \gamma\mathbb{I}_m & 0 & \dots & 0 \\ \gamma\mathbb{I}_m & G & \gamma\mathbb{I}_m & \dots & 0 \\ \ddots & \ddots & \ddots & \ddots & \\ 0 & \dots & \gamma\mathbb{I}_m & G & \gamma\mathbb{I}_m \\ 0 & \dots & 0 & \gamma\mathbb{I}_m & G \end{pmatrix}$ avec $G = \begin{pmatrix} \alpha & \beta & 0 & \dots & 0 \\ \beta & \alpha & \beta & \dots & 0 \\ \ddots & \ddots & \ddots & \ddots & \\ 0 & \dots & \beta & \alpha & \beta \\ 0 & \dots & 0 & \beta & \alpha \end{pmatrix}$.$(g)$

   et

   $$\alpha = \frac{2}{(\Delta x)^2} + \frac{2}{\Delta y)^2} > 0, \beta = -\frac{1}{(\Delta x)^2}, \gamma = -\frac{1}{(\Delta y)^2} \quad avec \quad \Delta x = \frac{a}{m+1}, \Delta y = \frac{b}{p+1}$$

   On a bien A symétrique réelle alors elle admet des valeurs propres réelles.
   –A est-elle définie positive ?
   $\forall i,j \in [1,n]$, on a : $|a_{ii}| \leq \sum_{i \neq j}^{n} |a_{ij}|$ alors A n'est pas "dsd" mais $\exists k \in [1,n]$ tel que $|a_{kk}| \leq \sum_{k \neq j}^{n} |a_{kj}|$ alors A est diagonale fortement dominante.
   Le graphe de A est fortement connexe car à partir de chaque sommet, on arrive à rejoindre les autres sommets, donc A est irréductible.
   Alors A est définie positive de plus $\forall i \in [1,n], a_{ii} > 0$.
   Par conséquent A admet n valeurs propres réelles strictement positives rangées par ordre croissant $0 < \lambda_1^h \leq \lambda_2^h \leq \dots \leq \lambda_n^h$.

2. Le spectre de B.
   On a : $Sp(A) = \{\lambda_1^h, \lambda_2^h, ..., \lambda_n^h\}$ tel que $0 < \lambda_1^h \leq \lambda_2^h \leq \dots \leq \lambda_n^h$.
   Soit v vecteur propre de A tel que :

   $$A.v = \lambda^h v \Leftrightarrow v = \lambda^h.A^{-1}.v$$
   $$\Leftrightarrow A^{-1}.v = \frac{1}{\lambda^h}.v$$

   alors

   $$Sp(B) = \{\frac{1}{\lambda_1^h}, \frac{1}{\lambda_2^h}, ..., \frac{1}{\lambda_n^h}\} \quad avec \quad 0 < \frac{1}{\lambda_n^h} \leq ... \leq \frac{1}{\lambda_2^h} \leq \frac{1}{\lambda_1^h}.$$

3. Montrons que $\frac{1}{\lambda_2^h}$ est la plus grande valeur propre de B_1.

–On a :

$$B_1 v_1 = Bv_1 - \frac{1}{\lambda_1^h} v_1 v_1^t v_1 \quad or \quad \|v_1\|^2 = 1$$
$$B_1 v_1 = Bv_1 - \frac{1}{\lambda_1^h} v_1 \quad or \quad Bv_1 = \frac{1}{\lambda_1^h} v_1$$
$$B_1 v_1 = 0.v_1$$

$\Rightarrow v_1$ est vecteur propre de B_1 pour la valeur propre 0.

–On a : $\forall i \in [2, N]$,

$$B_1 v_i = Bv_i - \frac{1}{\lambda_1^h} v_1 (v_1^t v_i) \quad or \quad < v_1, v_i >= 0$$
$$B_1 v_i = B.v_i = \frac{1}{\lambda_i^h} v_i$$

$\Rightarrow v_i$ est vecteur propre de B_1 pour la valeur propre $\frac{1}{\lambda_i^h}$.

Alors :

$$Sp(B_1) = \{0, \frac{1}{\lambda_2^h}, ..., \frac{1}{\lambda_n^h}\} \quad avec \quad 0 < \frac{1}{\lambda_n^h} \leq ... \leq \frac{1}{\lambda_2^h}.$$

La mise en œuvre : Méthode et Programmation

1. La fonction Octave de A.

La matrice A dépend de 4 paramètres : a, b, m, p tel que défini au (g), Elle est tridiagonale par blocs de taille $n = m \times p$.

```
function [A]=mat_proj(a,b,m,p)

n=m*p;
nab_x=a/(m+1);
nab_y=b/(p+1);

alpha=(2/((nab_x)**2) )+(2/((nab_y)**2));
beta=-1/((nab_x)**2);
gamma=-1/((nab_y)**2);
G=mat_trig(alpha,beta,m)

I=gamma*eye(m,m)
A=zeros(n,n);
#Pour m,p=3
A=[G,I,zeros(n-2*m,n-2*m);I,G,I,zeros(n-3*m,n-3*m);zeros(m,m),I,G];
#Pour m,p=4
#A=[G,I,zeros(m,m),zeros(m,m);I,G,I,zeros(m,m);zeros(m,m),I,G,I;zeros(m,m),zeros(m,m),I,G]
```

FIGURE 5 – Fonction de la matrice A

Par exemple : si $a = 2, b = 1, m = 2, p = 3$

$$A = \begin{pmatrix} 36.5000 & -2.2500 & -16.0000 & 0 & 0 & 0 \\ -2.2500 & 36.5000 & 0 & -16.0000 & 0 & 0 \\ -16.0000 & 0 & 36.5000 & -2.2500 & -16.0000 & \\ 0 & -16.0000 & -2.2500 & 36.5000 & 0 & -16.0000 \\ 0 & 0 & -16.0000 & 0 & 36.5000 & -2.2500 \\ 0 & 0 & 0 & -16.0000 & -2.2500 & 36.5000 \end{pmatrix}$$

2. La plus petite valeur propre de A.

Explication : Sachant que la méthode de la puissance permet d'avoir la valeur propre dominante en module d'une matrice quelconque, on peut donc l'appliquer à A^{-1} pour avoir le max des valeurs propres de A^{-1} puis on inverse le maximum trouvé, celui-ci correspond au min de valeurs propres de A. (Voir Fonction puissinv)

Soit $\lambda \in Sp(A)$, on a : $\max |\frac{1}{\lambda}| = \frac{1}{\min |\lambda|}$. [7]

```
function [val,x,iter]=puissinv(A,x0,maxiter,tol)

  val=0;
  x=x0;
  L=zeros(maxiter,1);
  test=tol+1;
  iter=1;
  while (iter<=maxiter && test>tol)
      z=inv(A)*x;
      val= x'*z;
      x=z / norm(z,2);
      L(iter)=val;
      if iter>1
        test=abs(L(iter-1)-L(iter));
      endif
        iter=iter+1;
  endwhile
  val=1/val;
%lambda1: l'approximation de la plus petite valeur propre obtenue
```

FIGURE 6 – Fonction puissance Inverse.

Supposons que A soit non inversible, alors cette fonction ne convergera, dans ce cas, on peut factoriser A en décomposition LU($PA = LU$) afin d'avoir successivement les itérées des vecteurs propres. Supposons que A est non inversible, alors cette fonction ne convergera, dans ce cas, on peut factoriser A en décomposition LU($PA = LU$) afin d'avoir successivement les itérées des vecteurs propres. [6] [7] (Voir Fonction puissinvb pour plus détails.)

```
function [val,b]=puissinvb(A,x0,tol)

  [L,U,P]=lu(A);
  val=0;
  val1=1;
  x=x0;

  while (abs(val1-val)>tol)
      val1=val;
      b=x / norm(x);
      x= (L*U) \ (P*b);
      val=1 / (x'*b);
  endwhile
  #val=1/val;
%val: l'approximation de la plus petite valeur propre obtenue
```

FIGURE 7 – Fonction puissance Inverse avec LU.

3. Répresentation du vecteur propre associé à λ_1 avec $a = 2, b = 1, m = 5, p = 4$.
On définit en d'abord la matrice A avec $a = 2, b = 1, m = 5, p = 4$, ensuite on calcule du vecteur propre x de la plus petite valeur propre avec la fonction puissinv puis pour la représentation on se sert du TP2 en réaffectant le vecteur propre x dans la fonction propre $u(x, y)$ notée U.
(Voir script essai_mat_proj)

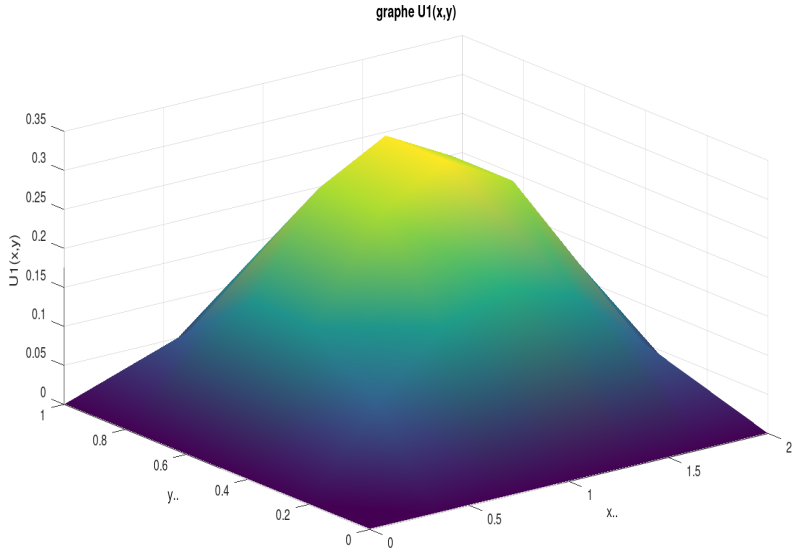


FIGURE 8 – Graphe du vecteur propre de la plus petite valeur propre A avec $a = 2, b = 1, m = 5, p = 4$.

4. Calculons $\lambda_2, ..., \lambda_6$ et répresentons les vecteurs propres associés.
L'idée est pareille à celle précédente, on utilise la méthode de déflation (cf. (h)) pour calculer $\lambda_2, ..., \lambda_6$ et les vecteurs propres associés grâce à λ_1 puis on les répresente graphiquement.
(Voir script essai_mat_proj partie 4.)

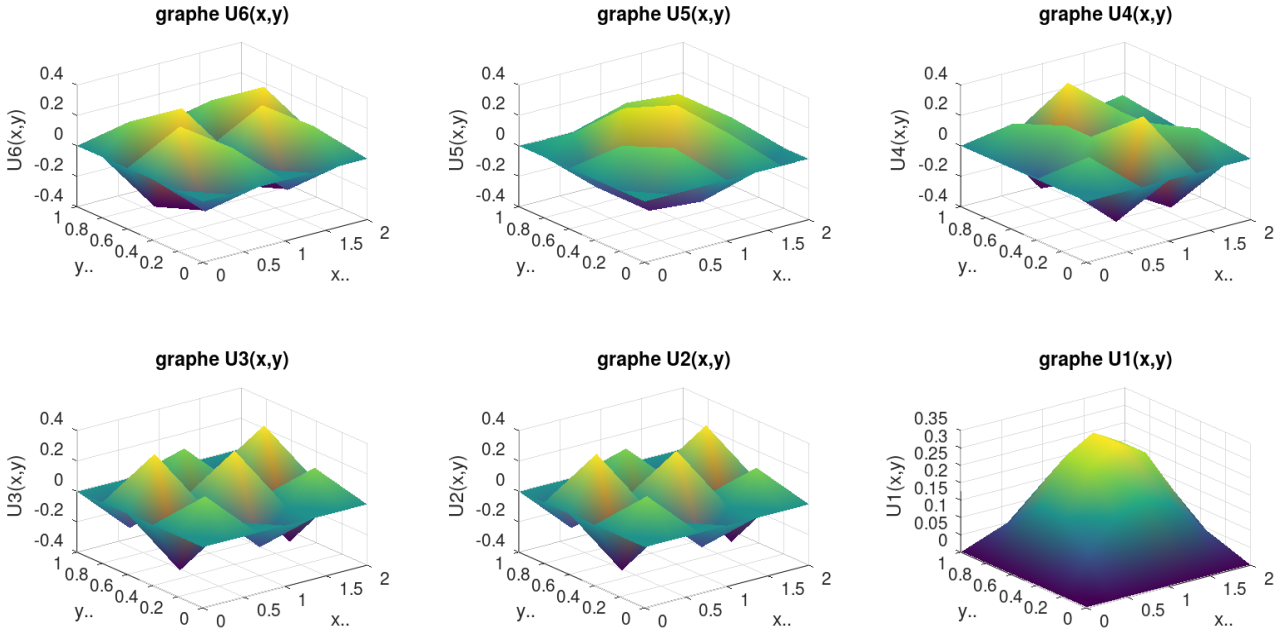


FIGURE 9 – Graphe des vrais fonctions propres associés à $\lambda_2, \ldots, \lambda_6$ avec $my\_eigs\_deflation$ pour $a = 2, b = 1, m = 5, p = 4$.

Remarques : On a : $\lambda_2 \approx 99.039$, $\lambda_3 \approx 108.451$, $\lambda_4 \approx 108.451$, $\lambda_5 \approx 117.451$, $\lambda_6 \approx 124.039$.
Elles sont bien identiques à celle des valeurs propres avec *eigs* et Les vecteurs propres associées aux valeurs propres avec *eigs* ne sont pas tous pareils à celle trouvés par la méthode de déflation, ce qui prouve que cette méthode ne semble pas précise dans le calcul des vecteurs propres.

(h) La fonction my_eigs_deflation : Elle permet de calculer $\lambda_1, \lambda_2, \lambda_3$ à partir d'une valeur propre λ déjà calculée, l'idée ici est de réutiliser la méthode de puissance sur $A_1 = A - \lambda.x.x^t$ (avec x le vecteur propre associé à λ) au fur à mesure pour trouver les valeurs propres. [1] [7]
Ensuite, on crée un vecteur contenant les 3 valeurs propres et l'ordonne grâce à la fonction *sort* pour qu'il affiche les valeurs propres par ordre décroissant comme $eigs(A, 3)$.
(Voir my_eigs_deflation et essai_my_eigs_deflation.)

5. Méthode QR.
La fonction my_eigs_qr : Elle permettra d'avoir directement une approximation de toutes les valeurs propres d'une matrice A comme $eigs()$, en effet en transformant A en une matrice semblable et triangulaire T grâce à la factorisation QR ($A = P.T.P^{-1}$). (Voir my_eigs_qr.)
Comme $A \approx T \Rightarrow Sp(A) = Sp(T) = \{t_{11}, \ldots, t_{nn}\}$ i.e. les valeurs propres de A sont sur la diagonale de T. [1] [6] [7]
Ensuite, on utilise la méthode de puissance inverse (Voir fonction puissinvT) avec $\mu = t_{ii}$ pour avoir une meilleure approximation de valeurs propres et le vecteur propre associé.[6] [7]
(Voir essai_my_eigs_qr pour le calcul des valeurs propres et leurs vecteurs propres associés pour $a = 2, b = 1, m = 5, p = 4$.)

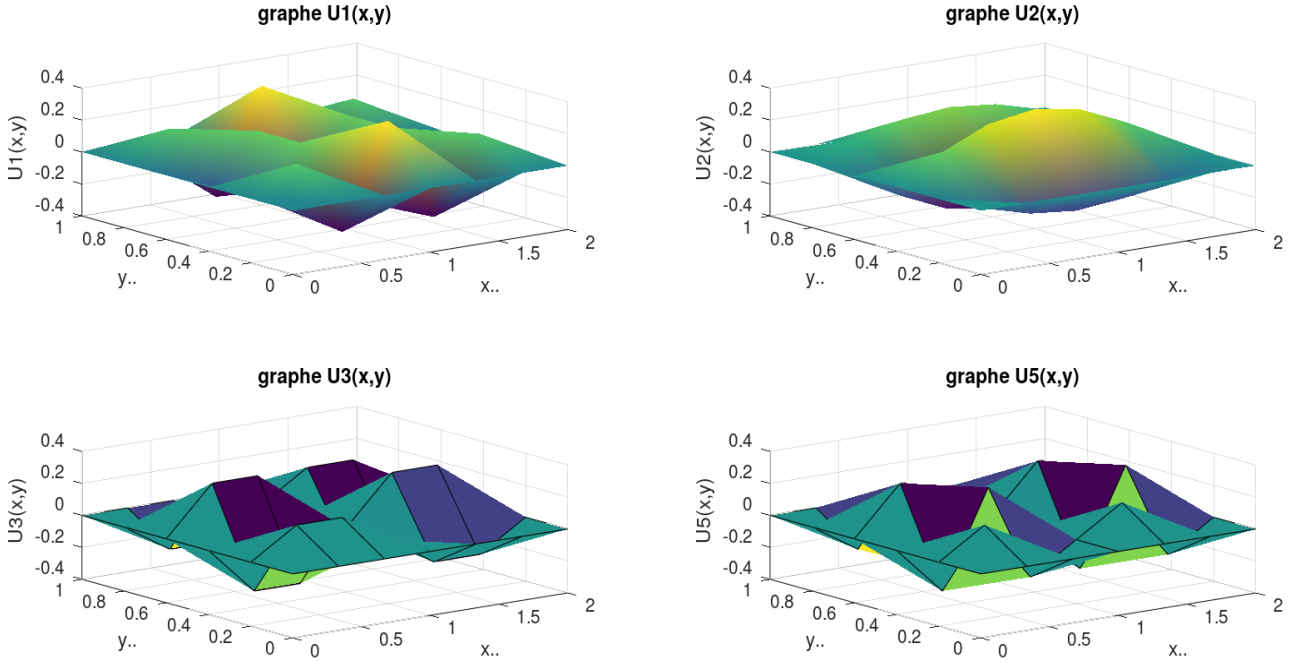


FIGURE 10 – Répresentation de quelques fonctions propres associées au valeurs propres de A par la méthode QR avec $a = 2, b = 1, m = 5, p = 4$.

Remarque : Cette méthode semble plus précise car les vecteurs propres trouvés sont approximatives à ceux de $eigs(A)$.

```
function [T]=my_eigs_qr(A,tol,maxiter)
  T=A;
  d0=diag(T);
  iter=1;

  for iter=1:maxiter
    [Q,R]=qr(T);
    T=R*Q;
    d1=diag(T);
    if (norm(T)<tol && norm(d1-d0)<tol)
      break
    endif
    d0=d1;
  endfor
```

FIGURE 11 – Fonction my_eigs_qr

6. Comparaison des méthodes.
   La méthode de déflation utilisée en 4. est totalement basée sur la méthode de puissance et les résultats obtenus dépendent des précédents trouvés alors s'il y'a une erreur dans la fonction de la puissance, les résultats seront complétement erronés (l'accumulation d'erreurs) oú si la matrice n'est pas inversible, il faut alors passer par une décomposition de A (en LU cf. script puissinvb) pour avoir la plus petite valeur propre, ce qui n'est pas avantageux.
   La méthode QR utilisée en 5. permet d'avoir directement une approximation des toutes les valeurs propres dès le départ, cela demande beaucoup de coûts vu la factorisation QR ; ensuite grâce à la méthode de puissance itérée approchée avec μ (cf. fonction puissinvT), on obtient une meilleure approximation d'une valeur propre et du vecteur propre associé. L'inconvénient est qu'il peut arriver que lors de l'exécution de la puissance itérée on ait le système linéaire presque singulier (le cas du vecteur propre associé à λ_5), ce provoque une convergence vers des résultats erronés
   (Voir FIGURE 10- U3 et U5).

   Pour $a = 2, b = 1, m = 5, p = 4$, La répresentation des vecteurs propres par $eigs()$ est la suivante :

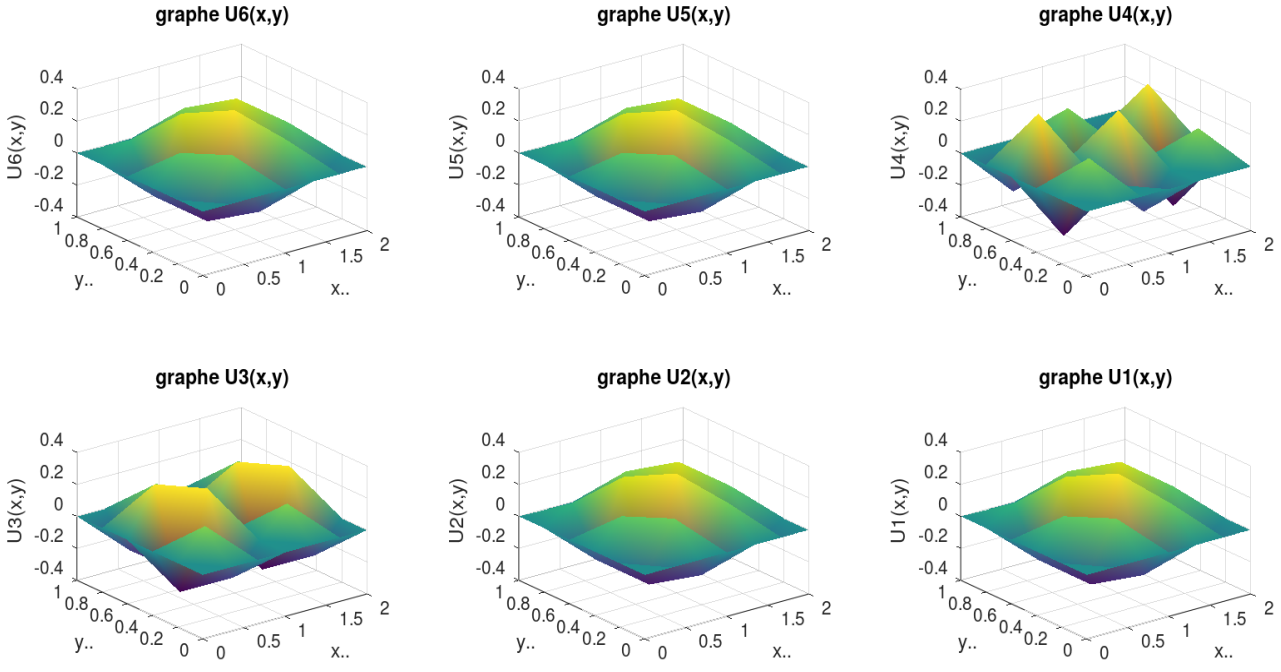


FIGURE 12 – Graphe des vrais fonctions propres associées à $\lambda_1, \ldots, \lambda_6$ avec $eigs$ pour $a = 2, b = 1, m = 5, p = 4$.

–Observation de la méthode de déflation :
En comparant, Il y'a que les graphes $U3, U2, U1$ qui sont vraiment différents à la Figure 12, Cela est dû à certains vecteurs propres qui ont des valeurs opposées. Dans l'ensemble, elles sont approximatives à la Figure 12.
Le temps écoulé est 0.0509059 secondes.
–Observation de la méthode QR :
En comparant, Il y'a que les graphes $U5$ qui sont vraiment différents à la Figure 12, Cela est dû à la résolution le système linéaire presque singulier dans la méthode de puissance itéré.
Le temps écoulé est 0.299125 secondes.

Conclusion :

Au regard des remarques et des analyses précédentes, nous pouvons dire que la méthode QR semble plus performante (convergence rapide) bien que plus élevé en terme de coût vu la complexité de la factorisation QR qui est de 3, on pourrait peut-être transformer A dès le départ (en matrice de Hessenberg) afin de mieux l'optimiser. [1] [7]

Références

[1] *MA604 : Analyse numérique Matricielle II* , **de François Lefrèvre.**

[2] *Les matrices et l'algorithme PageRank de Google* , **de Olivier GUIBE.**

[3] *Google PageRank ou une façon de classer les pages web*, **de Olivier GUIBE.**

[4] *L'algorithme du PageRank expliqué*, **Site :** https ://urlz.fr/fxZT

[5] *Théorème de Perron-Frobenius*, **Site :** https ://urlz.fr/fxZW

[6] *Méthode des puissances et des puissances inverses* , **de Yassine Boulaich.**

[7] *Analyse numérique matricielle appliquée à l'art de l'ingénieur* , **de Patrick Lascaux et Raymond Théodor.**