BROUSSE Léa
DURET Guillaume
Groupe A

Année 2017-2018

CPE Lyon – 3 ETI
TD-TP2 Algèbre linéaire

Exercice 4 : Méthode récursive d'inversion matricielle.

Cet exercice nous proposait d'écrire une méthode récursive qui nous permet d'inverser toute matrice carrée inversible.

Soit M ∈ $\mathbb{R}^{nxm}$, on pose $M = \begin{pmatrix} A & U \\ V^T & \alpha \end{pmatrix}$ où α∈ $\mathbb{R}$ et A est inversible.

Il nous est proposé d'étudier l'identité suivante :

$$\begin{pmatrix} A & U \\ V^T & \alpha \end{pmatrix} \cdot \begin{pmatrix} P & Q \\ W^T & 1/s \end{pmatrix} = \begin{pmatrix} In-1 & 0 \\ 0 & 1 \end{pmatrix}$$

Cette identité est vérifiée si la matrice M est inversible.

Nous allons déterminer les expressions du réel s, des vecteurs W et Q, et de la matrice P. Pour cela, nous allons résoudre le système suivant :

$$\begin{cases} AP + UW^T = In - 1 \\ AQ + U\dfrac{1}{s} = 0 \\ V^T P + \alpha W^T = 0 \\ V^T Q + \dfrac{\alpha}{s} = 1 \end{cases}$$

Après plusieurs étapes de calcul, nous arrivons au système d'équations suivant :

$$\begin{cases} s = \alpha - V^T A^{-1} U \\ Q = \dfrac{1}{V^T A^{-1} U - \alpha} A^{-1} U \\ W^T = \dfrac{1}{V^T A^{-1} U - \alpha} V^T A^{-1} \\ P = A^{-1}(In - 1 - \dfrac{1}{V^T A^{-1} U - \alpha} U V^T A^{-1} \end{cases}$$

Ces expressions nous seront utiles dans la fonction Matlab que nous allons écrire pour inverser n'importe quelle matrice inversible de la forme de M

Cette fonction utilisant le principe de récursivité est la suivante :

```
function M_inv=inverse(M)
        taille = length(M);
        if taille >=2
               A=M(1:(taille-1),1:(taille-1));
               VT=M(taille,1:taille-1) ;
               U=M(1:(taille-1),taille) ;
               alpha=M(taille,taille) ;
               A1=inverse (A); %Appel récursif
```

```
            P=A1*(eye(taille-1)-U*(VT)*A1/(VT*A1*U-alpha));
            Q=A1*U/(VT*A1*U-alpha);
            s=alpha-VT*A1*U;
            WT=VT*A1/(VT*A1*U-alpha) ;
            M_inv=[P,Q;WT,1/s] ;

      else %Si la matrice est de taille 1
            M_inv=1/M;
      end

end
```

Nous souhaitons inverser la matrice $M = \begin{pmatrix} 2 & 3 & -1 \\ 1 & 4 & 2 \\ -2 & 1 & 4 \end{pmatrix}$

Pour cela, nous appelons la fonction réalisée précédemment dans notre script pour obtenir M^{-1}.

```
clear all; close all;
M=[2 3 -1 ; 1 4 2 ; -2 1 4];
M_inv=inverse(M) %Appel de la méthode récursive

M^(-1) %Inverse de M calculé par Matlab pour vérifier notre méthode
```

```
M_inv =
   -2.8000    2.6000   -2.0000
    1.6000   -1.2000    1.0000
   -1.8000    1.6000   -1.0000

M^(-1)

ans =
   -2.8000    2.6000   -2.0000
    1.6000   -1.2000    1.0000
   -1.8000    1.6000   -1.0000
```

Nous obtenons la même matrice inverse de M par ces deux méthodes ce qui nous permet de valider notre fonction récursive.

Exercice 6 : Directions principales d'un nuage de points.

Cet exercice nous propose de déterminer dans un premier temps les directions principales d'un nuage de points donnés par deux méthodes différentes.

La première méthode consiste à une optimisation des projections. Pour cela, nous allons considérer la somme des produits scalaires de tous les points M_k du nuage de points sur la droite de vecteur directeur $\vec{u} = (cos\ \theta, sin\ \theta)^T$. A l'aide de Matlab, nous allons chercher à maximiser puis minimiser cette somme.

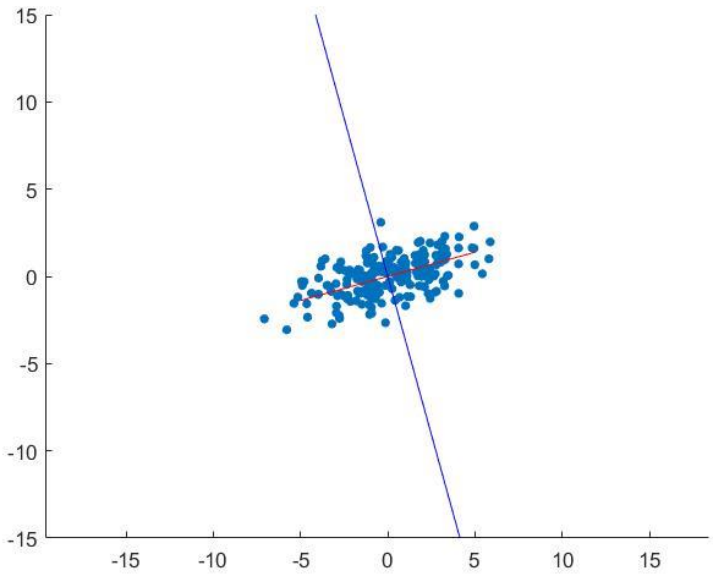
Nous commençons par recopier le début de script donné dans l'énoncé qui nous permet de créer et d'afficher un nuage de points gaussien centré sur l'origine.

```
clear all;close all;hold on; axis equal;

n=200;
y=randn(1,n);
x=y+2*randn(1,n);
X=x-mean(x);
Y=y-mean(y);
scatter(X,Y,20,'filled');
```

A la suite de ce début de script, nous calculons la somme des produits scalaire et affichons les droites matérialisant les directions principales du nuage de points en question.

```
lgr=length(x);
t=-5:0.01:5;
theta=0:0.01:2*pi;
s=0;

for k=1:lgr
    s=s+(cos(theta)*X(k)+sin(theta)*Y(k)).^2;
end

theta_max=theta(find(s==max(s)));
theta_min=theta(find(s==min(s)));

D1=(sin(theta_max)/cos(theta_max))*t;
D2=(sin(theta_min)/cos(theta_min))*t;
figure(1);
plot(t,D1,'r',t,D2,'b');
axis ([-15,15,-15,15]);
axis equal;
```

Nous obtenons la figure suivante :



Intéressons-nous maintenant à la seconde méthode qui s'appuie sur la diagonalisation de la matrice $G=NN^T$ dont les vecteurs propres seront les vecteurs directeurs des droites de directions du nuage de points étudié. Pour cela, nous déterminons tout d'abord les valeurs propres de la matrice G, puis ces vecteurs propres associés pour ainsi tracer les droites correspondantes. Nous calculons ensuite le déphasage de la droite associée à la plus grande valeur propre de G.

Nous ajoutons donc les lignes suivantes à la suite de la création du nuage de points précédente.

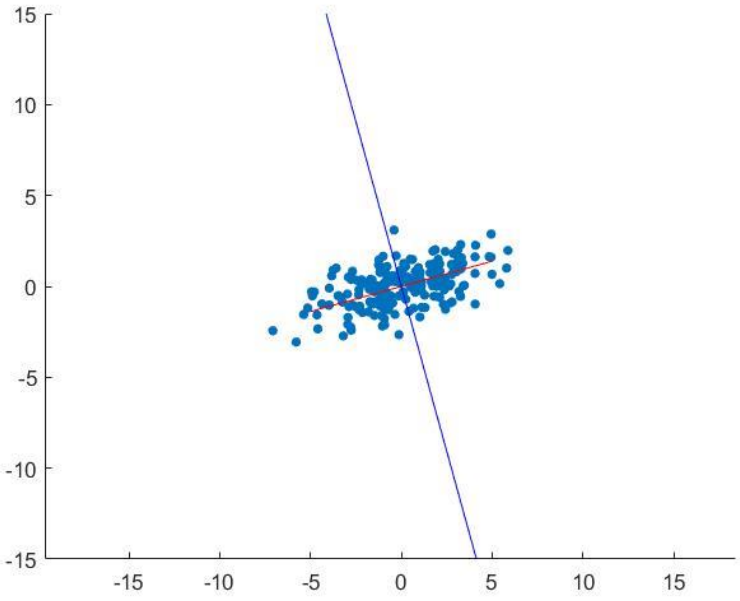
```
N=[X;Y];

G=N*N';

[P,D]=eig(G);

D11=(P(2,1)/P(1,1))*t;

D22=(P(2,2)/P(1,2))*t;

figure(2);

hold on;
scatter(X,Y,20,'filled')
plot(t,D11,'b',t,D22,'r');
axis ([-15,15,-15,15])
axis equal;

theta2=atan(P(2,2)/P(1,2));
theta2
theta_max
```

Ce code nous renvoie par exemple :

```
theta2 =

    0.2578

theta_max =

    0.2600
```

Dans ce code, nous avons tout d'abord diagonalisé la matrice G à l'aide de la fonction prédéfinie de Matlab `eig`() (qui nous renvoie la matrice diagonale D et la matrice de changement de base P). Nous avons ensuite tracé les droites D1 et D2 de vecteurs directeurs les vecteurs propres de G qui sont les colonnes de P.

Nous obtenons la figure suivante :



Nous remarquons que les droites de cette figure sont quasiment identiques à celles obtenues par la première méthode, avec approximativement les mêmes angles thêta.

Ces deux méthodes différentes nous permettent bien d'obtenir le résultat souhaité. Ceci s'explique par le fait que les vecteurs propres d'une matrice diagonale forment une base de l'endomorphisme E. Or une base orthonormée d'un espace euclidien (E,(.|.)) est formée à partir du produit scalaire des vecteurs de E.

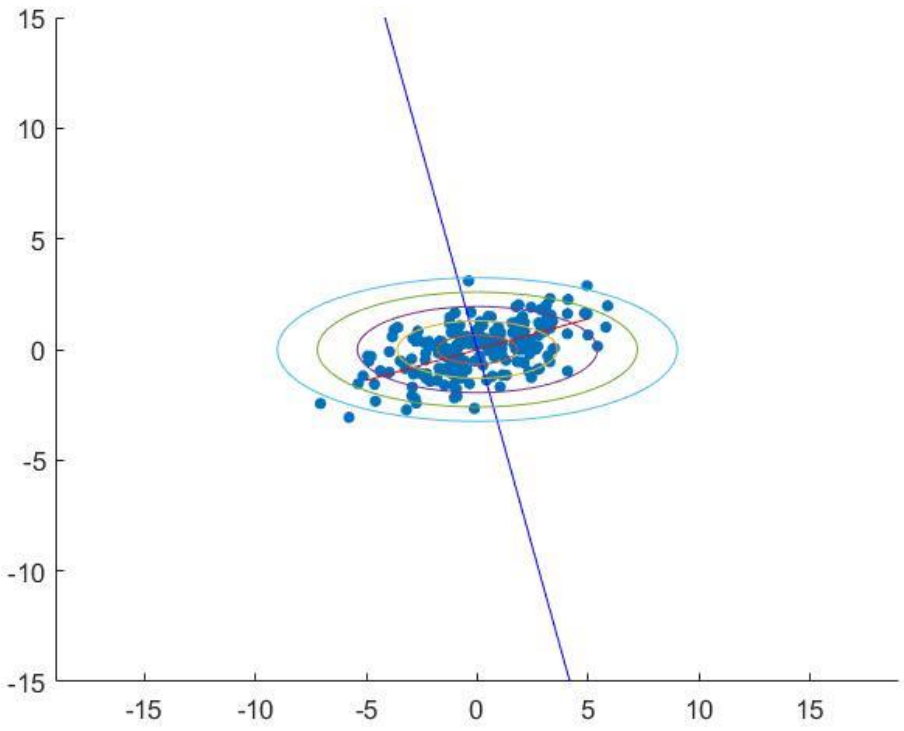
Nous souhaitons désormais tracer sur les graphiques précédents les ellipses de confiance de demi grand axe $\alpha\sqrt{\lambda 2}$ et de demi petit axe $\alpha\sqrt{\lambda 1}$. Ces les ellipses ont pour équation paramétrique : $\begin{cases} x(\theta) = A\cos\theta \\ y(\theta) = B\ sin\ \theta' \end{cases}$ avec $A = \alpha\sqrt{\lambda 1}$, = $B = \alpha\sqrt{\lambda 1}$, $\theta \in [0; 2\pi]$.

Nous réalisons tout d'abord le script suivant :

```
figure (3)

hold on;
scatter(X,Y,20,'filled');
plot(t,D11,'b',t,D22,'r');
for a=0.05:0.05:0.25
    M=[a*sqrt(D(1,1))*cos(theta);a*sqrt(D(2,2))*sin(theta)];
    plot( M(2,:),(M(1,:)));
    axis ([-15,15,-15,15]);
    axis 'equal';
end
```

Nous faisons varier le paramètre α à l'aide d'une boucle for en affichant dans la boucle les arcs paramétrés.

Nous obtenons la figure suivante :

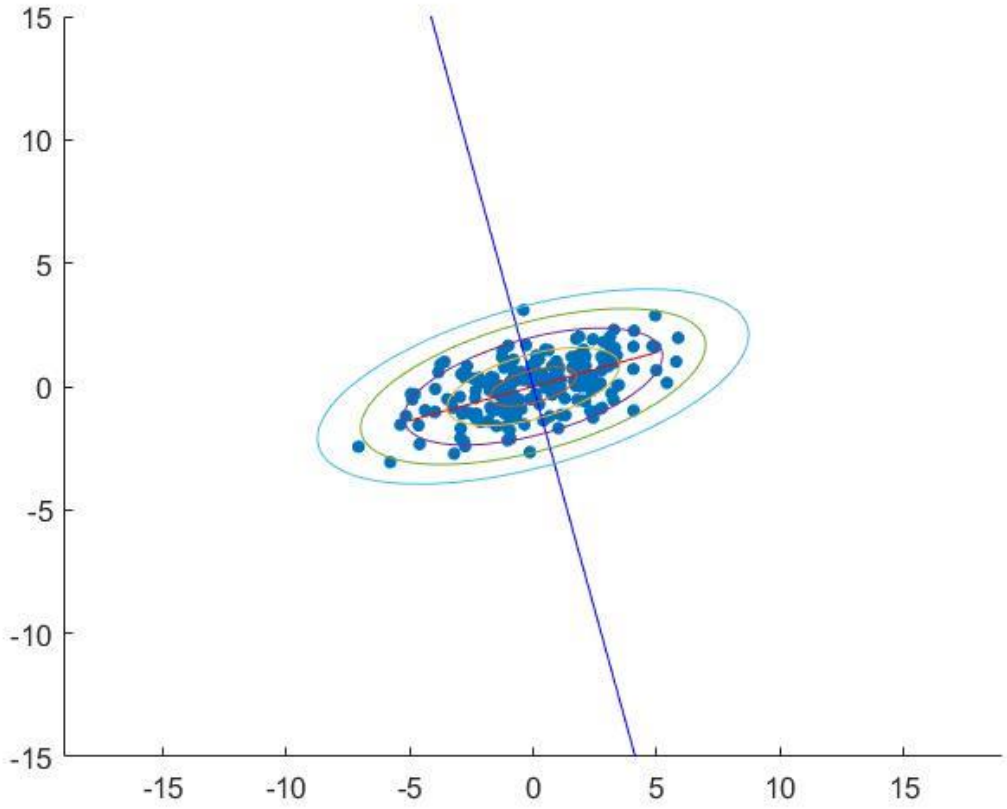


Nous remarquons que les arcs tracés ne sont pas bien orientés ce qui est dû au fait que nous traçons les ellipse dans la base (x,y) alors qu'il faudrait les tracer dans la base engendrée par les droites principales pour avoir des ellipses de confiance.

Pour résoudre ce problème d'orientation, il suffit de multiplier l'équation des ellipses par la matrice de changement de base.

```
figure (4)

hold on;
scatter(X,Y,20,'filled');
plot(t,D1,'r',t,D2,'b');
for a=0.05:0.05:0.25
    M3=[a*sqrt(D(1,1))*cos(theta);a*sqrt(D(2,2))*sin(theta)];
    M2=(M3')*P;
    plot (M2(:,1),M2(:,2));
    axis ([-15,15,-15,15]);
    axis 'equal';
end
```

Nous avons donc multiplié le vecteur $\begin{cases} x(\theta) = A\cos\theta \\ y(\theta) = B\ sin\ \theta \end{cases}$ par la matrice P obtenue précédemment.



Nous voyons cette fois ci que les ellipses sont bien orientées par rapport au nuage de point et qu'elles l'englobent de façon optimale.
Nous remarquons aussi que plus le paramètre α est élevé plus l'ellipse correspondant contient de point.

Cet exercice nous a donc permis de déterminer les directions principales d'un nuage de points aléatoires ainsi que les ellipses de confiance associées.