CPE Lyon 3ETI 2020-2021

Algèbre linéaire (M-ALG) : Travaux pratiques Matlab

Exercice 1

On considère la matrice la matrice $\mathbf{A} = \begin{pmatrix} 1 & -2 & 3 & 1 \\ 2 & 5 & -1 & 3 \\ 4 & 1 & 0 & 2 \\ 0 & -2 & 3 & 1 \end{pmatrix}$.

1) Créer la matrice **A** et l'afficher.
2) Utiliser la méthode des 2 points « : » pour :
    a. Afficher la 3ème colonne
    b. Afficher la 4ème ligne
3) Remplacer la 3ème colonne de **A** par le vecteur $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ et afficher **A**.

    Désormais $\mathbf{A} = \begin{pmatrix} 1 & -2 & 1 & 1 \\ 2 & 5 & 1 & 3 \\ 4 & 1 & 1 & 2 \\ 0 & -2 & 1 & 1 \end{pmatrix}$.
4) Calculer et afficher le déterminant et la trace de **A**.
5) Résoudre le système $\mathbf{Ax} = \mathbf{b}$ où $\mathbf{b} = (1, 2, -1,3)^T$ et afficher la solution.
6) A partir de la matrice **A** (et en utilisant la méthode des 2 points « : »), créer les matrices suivantes :

$$\mathbf{B} = \begin{pmatrix} -2 & 1 \\ 5 & 1 \\ 1 & 1 \\ -2 & 1 \end{pmatrix}, \qquad \mathbf{C} = \begin{pmatrix} 1 & -2 & 1 & 1 \\ 4 & 1 & 1 & 2 \end{pmatrix}, \qquad \mathbf{D} = \begin{pmatrix} 5 & 1 & 3 \\ 1 & 1 & 2 \\ -2 & 1 & 1 \end{pmatrix}$$

Créer la matrice $\mathbf{M} = \mathbf{CC}^T$ et l'afficher, puis diagonaliser **M** à l'aide de la commande `eig` (on affichera la matrice de passage et la matrice diagonale). Vérifier, à l'aide de Matlab, que la matrice de passage est bien orthogonale.

Exercice 2

Partie A : projection orthogonale et méthode des moindres carrés

Compléter le code Matlab de la page suivante afin que celui-ci permette de :

- créer n points P_i du plan dont les abscisses sont des nombres aléatoires compris entre -3 et 5, et dont les ordonnées sont des nombres aléatoires compris entre -4 et 3
- afficher les points sous forme de petits cercles (voir Fig. 1)
- créer une matrice $2 \times n$: la 1ère ligne contient les abscisses, la 2nde les ordonnées des points P_i
- tracer la droite d'équation $y = \frac{1}{2}x$ en trait continu (voir Fig. 1), on note $\mathcal{D}$ cette droite
- projeter les points P_i sur la droite $\mathcal{D}$, on note Q_i les points projetés
- afficher les points Q_i sous forme d'étoiles (voir Fig. 1) et relier les points P_i et Q_i par un trait continu
- apporter une perturbation d'amplitude $\delta = 0{,}8$ sur les coordonnées des points Q_i, on note R_i les points obtenus
- afficher les points R_i sous forme de triangles rouges (voir Fig. 1)
- calculer et tracer, en traits pointillés rouges, la droite passant au plus près des points R_i (voir Fig. 1)

L'exécution du programme doit donner la figure suivante :

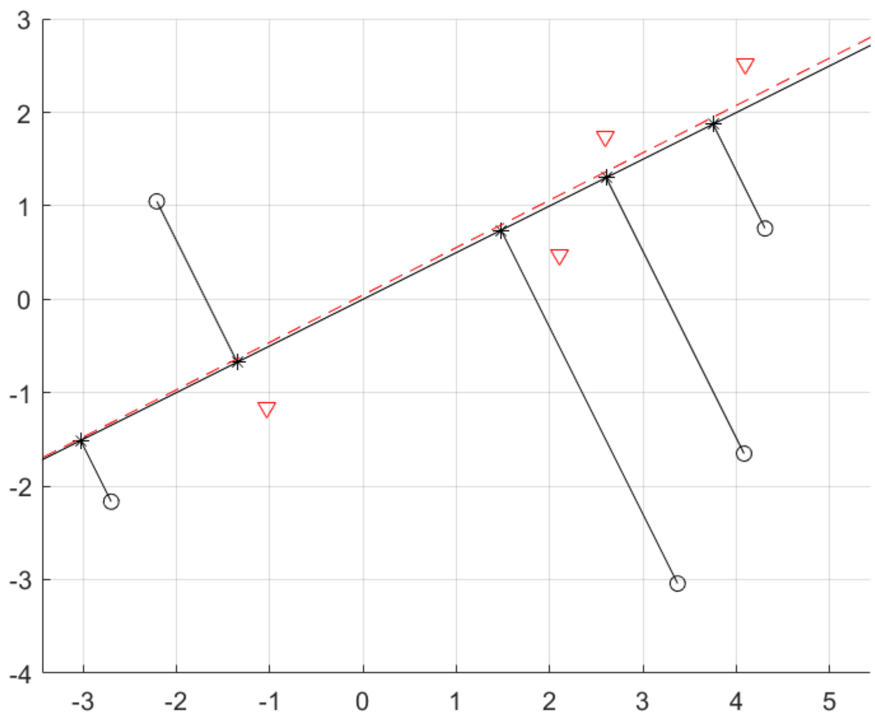


Figure 1 : Points de coordonnées aléatoires (cercles) projetés sur la droite d'équation $y = x/2$. Les coordonnées des points projetés (étoiles) sont perturbées de façon aléatoire (triangles rouges). La droite en pointillés rouges est la droite qui passe au plus près des points perturbés.

Programme à compléter :

```
clear variables;close all;hold on;

n=5;  % nombre de points
x=_______________________________;        % abscisses (vecteur ligne)
y=_______________________________;        % ordonnées (vecteur ligne)
plot(x,y,'ok');                          % affichage des points (cercles noirs)
U=______________________;                % matrice 2xn (concaténation de x et y)

% droite d'équation y=0.5*x
t=[-4,6];                 % abscisses
z=t/2;                    % ordonnées
plot(t,t/2,'k');          % affichage de la droite (trait continu)

% propriétés du repère
axis([-3,5,-4,3]);
grid on;
axis('equal');

% vecteur directeur de la droite
N=____________________________________________;

% matrice de projections sur la droite
P=____________________________________________;
```

```
% initialisation de la matrice 2xn contenant les abscisses (1ère ligne)
% et les ordonnées (2ème ligne) des points projetés sur la droite
V=zeros(2,n);

% calcul des coordonnées des points projetés (BOUCLE FOR OBLIGATOIRE)
for k=_____________
    ____________________________________;
end

% affichage des points projetés (étoiles)
plot(V(1,:),V(2,:),'*k');

% on relie les points initiaux à leurs projections
for k=1:n
    plot(___________________________________________________________,'k');
end

% on perturbe les coordonnées des points projetés
delta=0.8;
dV=-delta/2+delta*rand(2,n);
W=V+dV;

% affichage des points perturbés (triangles)
plot(W(1,:),W(2,:),'vk');

% calcul de la droite passant au plus près des points perturbés (résolution
% au sens des moindre carrés d'un système Au=v, avec A et v à définir)
A=________________________________;
v=________________________________;
u=________________________________;

% affichage de la droite (en pointillés)
plot(_______________________________________________,'k--');
```

Partie B : trajectoire d'un astéroïde

Une conique est une courbe du plan définie par une équation de la forme :

$$ax^2 + by^2 + cxy + dx + ey + f = 0 \quad (1)$$

1) Question préliminaire

- Coder les deux scripts suivants et les enregistrer dans le même répertoire :

Script `myfun.m` :

```
function z=myfun(x,y,a,b,c,d,e,f)
    z=a*x.^2+b*y.^2+c*x.*y+d*x+e*y+f;
```

Script `exercice2.m` :

```
clear variables;close all;

%% question 1
x=-5:1:5;
```

```
y=x;
figure(1);
a=4;b=9;c=0;d=0;e=0;f=-36;
h=ezplot(@(x,y)myfun(x,y,a,b,c,d,e,f));
set(h,'color','k');
```

- Exécuter le script `exercice2.m`
- Changer les valeurs des paramètres `a, b, c, d, e, f` et exécuter à nouveau le script.

2) Le tableau suivant donne les coordonnées d'un astéroïde observé pendant quelques jours avant de disparaître.

x_i	y_i
-1.2500	3.1780
-1.1000	3.0460
-0.9320	2.9300
-0.7460	2.8340
-0.5420	2.7540
-0.3200	2.6960
-0.0740	2.6580
0.1940	2.6420
0.4900	2.6540
0.7860	2.6980

L'objectif est de déterminer la trajectoire complète de cet astéroïde autour du soleil.

- En supposant que chacun de ces points appartient à une conique d'équation de la forme (1), montrer que le problème revient à résoudre au sens des moindres carrés le système $\mathbf{Az} = \mathbf{b}$ où $\mathbf{A}$, $\mathbf{z}$ et $\mathbf{b}$ sont respectivement une matrice et deux vecteurs que l'on précisera.
- Résoudre le système à l'aide de Matlab et afficher sur un même graphique les points d'observation et la trajectoire de l'astéroïde (cf. Fig. 2).
- Afficher la distance entre $\mathbf{Az}$ et $\mathbf{b}$ (commande `norm`).

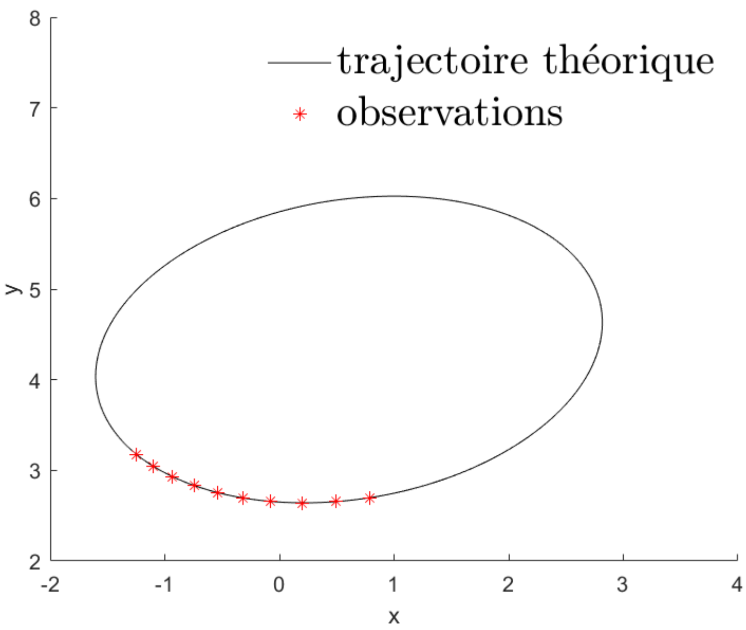


Figure 2 : Points d'observation de l'astéroïde (étoiles rouges) et trajectoire théorique (en noir) passant au plus près de ces points d'observation.

3) Votre programme doit être conçu et développé de façon générique afin de pouvoir l'utiliser pour toute autre trajectoire ; pour vous en assurer, testez votre programme pour un autre astéroïde dont les observations sont données dans le tableau suivant :

x_i	y_i
-2.5815	0.0847
-2.0762	0.3172
-1.3565	0.6058
-0.5279	0.9590
1.0265	1.2828
2.7484	0.8756
3.5944	0.5226
4.6929	0.1792

Exercice 3 : méthodes de la puissance itérée et de la déflation

Dans un script Matlab nommé `exercice3.m` générer la matrice tridiagonale suivante :

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & \ddots & & \\ & & \ddots & \ddots & -1 & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}$$

1) Calculer la valeur propre dominante de **A** (ainsi que le vecteur propre associé) par la méthode de la puissance itérée.

   Indications :

   - Fixer une précision pour le calcul de la valeur propre dominante
   - Utiliser une boucle `while` (sortir de la boucle lorsque la différence entre deux valeurs consécutives de la valeur propre dominante est inférieure à la précision)
   - Prévoir la possibilité de visualiser l'évolution des valeurs prises par la valeur propre dominante (voir Fig. 3)

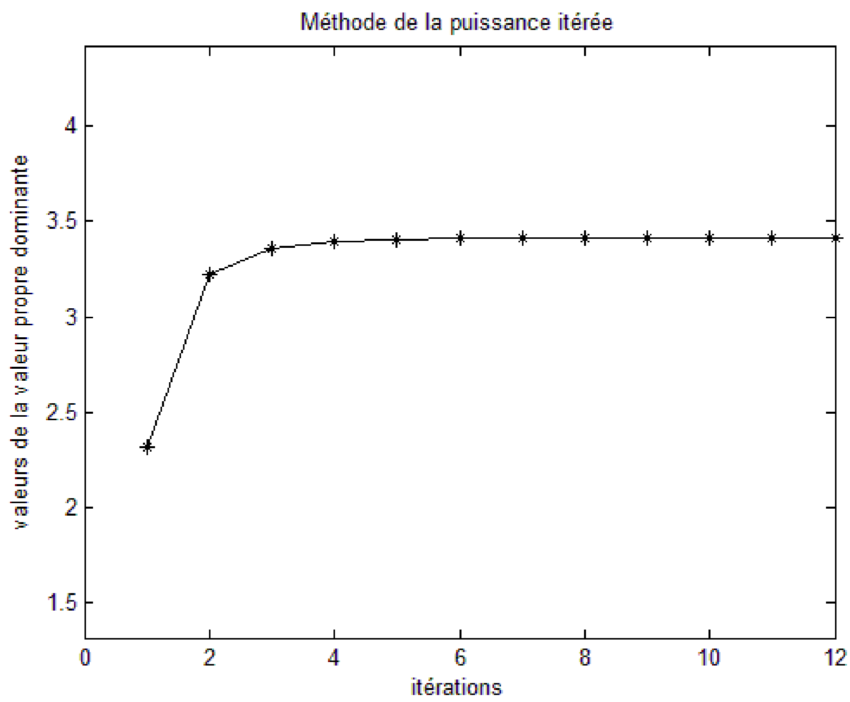


Figure 3 : valeurs successives de la valeur propre dominante dans le cas où $n = 6$.

2) Calculer les valeurs propres et les vecteurs propres de **A** par la méthode de la puissance itérée et la méthode de la déflation.

Indications :

- On remarquera que la matrice **A** est symétrique
- Stocker les valeurs propres dans une matrice diagonale **D** et les vecteurs propres associés dans une matrice **P**
- Proposer une méthode quantitative permettant de vérifier la précision de la diagonalisation ainsi obtenue (utiliser la commande `norm` et <u>ne pas utiliser</u> la commande `eig`)

<u>Conseil</u> : Introduire au début de votre code (juste après la création de la matrice **A**) les quelques lignes suivantes …

```
question=1;
switch question
    case 1
          …
          …
    case 2
          …
          …
end
```

… qui vous permettront d'exécuter votre programme de façon sélective selon que vous voulez répondre à la question 1 ou à la question 2.

Exercice 4 : décomposition en valeurs singulière (SVD)

On considère la matrice suivante :

$$\mathbf{A} = \begin{pmatrix} 1 & -1 & 2 & 5 \\ 1 & 1 & 4 & 5 \\ 2 & 0 & 6 & 10 \end{pmatrix}$$

a. En remarquant qu'il existe deux relations de combinaison linéaire entre les vecteurs colonnes de **A**, déterminer l'image puis le noyau de cette matrice.

b. En utilisant les commandes `svd` et `rank` de Matlab calculer l'image et le noyau de **A**.

<u>Indications</u> :

- construire la matrice `IM_A` dont les vecteurs colonnes sont les vecteurs de la base de Im **A**
- construire la matrice `KER_A` dont les vecteurs colonnes sont les vecteurs de la base de Ker **A**

c. Retrouvez-vous les mêmes résultats qu'à la question *a.* ?

<u>Indications</u> :

- construire la matrice `IM_A_TD` dont les vecteurs colonnes sont les vecteurs de la base de Im **A** trouvés en TD
- construire la matrice `KER_A_TD` dont les vecteurs colonnes sont les vecteurs de la base de Ker **A** trouvés en TD

d. Rajouter quelques lignes à votre programme permettant de s'assurer que les résultats trouvés par Matlab et ceux trouvés en TD sont équivalents ?

Exercice 5 : interprétation géométrique de la décomposition en valeurs singulière (SVD)

On considère la matrice suivante :

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ -1 & 1 \end{pmatrix}$$

a. Créer un fichier `exercice5.m`, ajouter le code suivant et exécuter le programme :

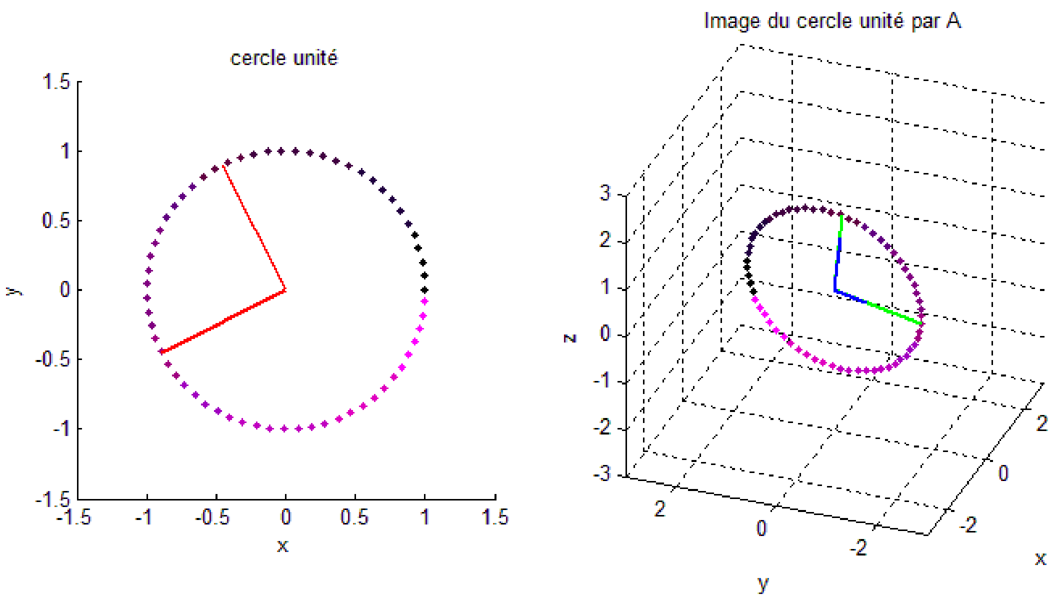
```
clear;close all;
theta=0:0.1:2*pi;
x=cos(theta);
y=sin(theta);
```

```
subplot(121);hold on;
for k=1:length(theta)
   plot(x(k),y(k),'marker','.','color',[k/length(theta),0,k/length(theta)]);
end
title('cercle unité');
xlabel('x');
ylabel('y');
axis('equal');
axis([-1.5,1.5,-1.5,1.5]);
```

b. Calculer la SVD de la matrice **A** (commande `svd`) et ajouter en rouge au graphe de gauche les vecteurs colonnes de **V** ($\mathbf{v}_1$ et $\mathbf{v}_2$: cf. cours chapitre SVD).

c. Afficher dans la même fenêtre mais dans un autre graphique l'image du cercle unité par la matrice **A** (commandes `subplot` et `plot3`) en utilisant le même marqueur et les mêmes codes couleur que pour le cercle unité (cf. figure ci-dessous).

d. Ajouter en bleu au graphe de droite les deux premiers vecteurs colonnes de **U** ($\mathbf{u}_1$ et $\mathbf{u}_2$) puis en vert les vecteurs $\sigma_1\mathbf{u}_1$ et $\sigma_2\mathbf{u}_2$.



Exercice 6 : SVD et compression d'image

Partie A

On considère la matrice suivante :

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

Compléter le code Matlab de la page suivante afin que celui-ci permette de :

- créer la matrice **M** par la concaténation d'une matrice carrée **A** et d'une matrice rectangulaire **B**
- calculer la décomposition en valeurs singulières de **M**

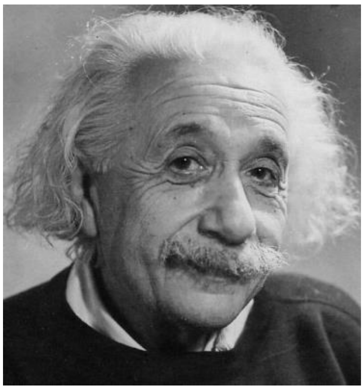
- reconstruire la matrice **M** à l'aide de la formule :

$$\mathbf{M} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

```
% construction des matrices A et B (SANS BOUCLE FOR NI WHILE)
A=_______________________________________;
B=_______________________________________;

% concaténation des matrices A et B
M=_______________________________________;

% décomposition en valeurs singulières de M
________________________________________;

% reconstruction de la matrice M
M2=_________________;

for i=_______________
    M2=_________________________________;
end
```

Partie B

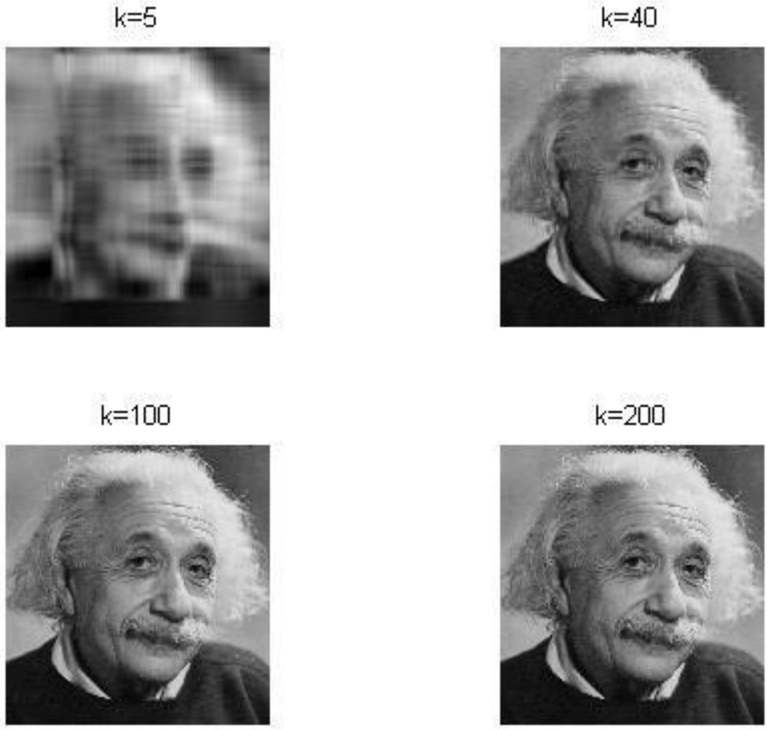
Une image en niveaux de gris peut être représentée par une matrice $\mathbf{A} = (a_{ij})$ dont chaque coefficient a_{ij} représente l'intensité (en niveaux de gris) du pixel (i, j). Par exemple, l'image d'Albert Einstein ci-contre possède 590×629 pixels (39,4 Ko), elle peut être représentée par une matrice de $m = 629$ lignes et $n = 590$ colonnes, c'est-à-dire par 371110 nombres compris entre 0 et 255.



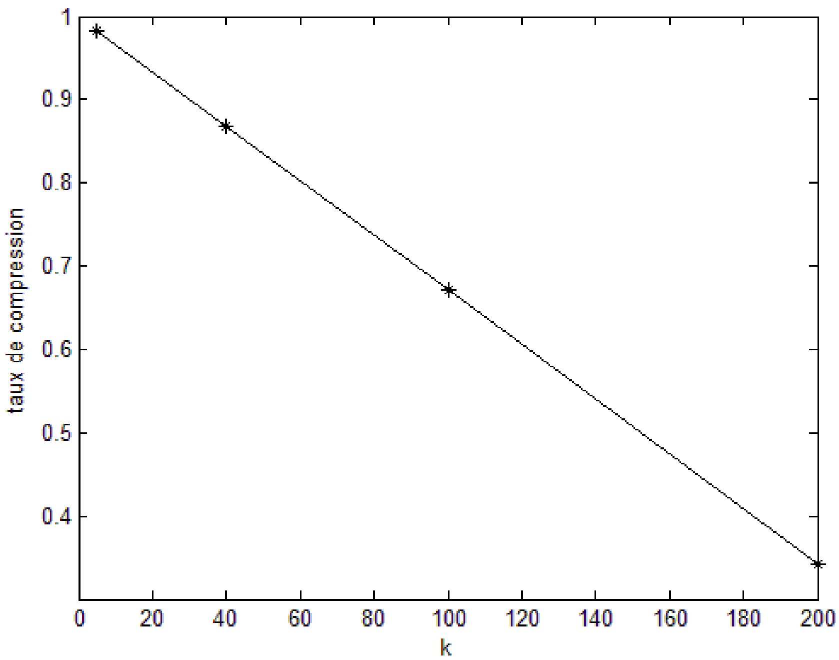
1) Récupérer l'image `einstein.jpg` dans CPe-campus et écrire un programme `exercice6.m` permettant de :
   - convertir l'image `einstein.jpg` sous la forme d'une matrice **A** et l'afficher (commandes `imread` et `imshow`)
   - calculer la SVD de cette matrice (commandes `double` et `svd`)
   - tracer la courbe des valeurs principales σ_i en fonction de i
   - calculer les matrices :

   $$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \quad \text{pour} \quad k = 5, 40, 100, 200$$

   On rappelle que l'image originale correspond à $k = n = 590$.
   - afficher dans une même fenêtre les images correspondant à ces matrices (commandes `uint8`, `imshow` et `subplot`) :



2) On souhaite transférer (d'un ordinateur à un autre) l'image compressée associée à la matrice $\mathbf{A}_k$. En dénombrant les informations à transmettre, proposer une expression du taux de compression τ (différence entre le nombre de pixels de l'image d'origine et le nombre d'informations transmises, rapportée au nombre de pixels) en fonction de m, n et k. Ajouter quelques lignes à votre programme permettant de tracer la courbe du taux de compression en fonction de k.



3) Votre programme doit être conçu et développé de façon générique afin de pouvoir l'utiliser pour toute autre image ; pour vous en assurer, testez votre programme sur une image de votre choix

Exercice 7 : Projection orthogonale et méthode des moindres carrés

1) Créer un script Matlab nommé `exercice7.m` et copier le code ci-dessous :

```
clear variables;
close all;
figure(1);hold on;
a=2;b=-5;c=1;
x2=-3:0.1:3;
y2=-2.5:0.1:4;
z2=-8:1:8;
[X2,Y2]=meshgrid(x2,y2);
Z2=-(a*X2+b*Y2)/c;
C(:,:,1)=zeros(size(Z2));      % red
C(:,:,2)=0.8*ones(size(Z2));   % green
C(:,:,3)=0.8*ones(size(Z2));   % blue
mesh(X2,Y2,Z2,C);
```

Exécuter le programme et commenter.

2) Générer de façon aléatoire les coordonnées de `n=10` points de l'espace P_i tels que :

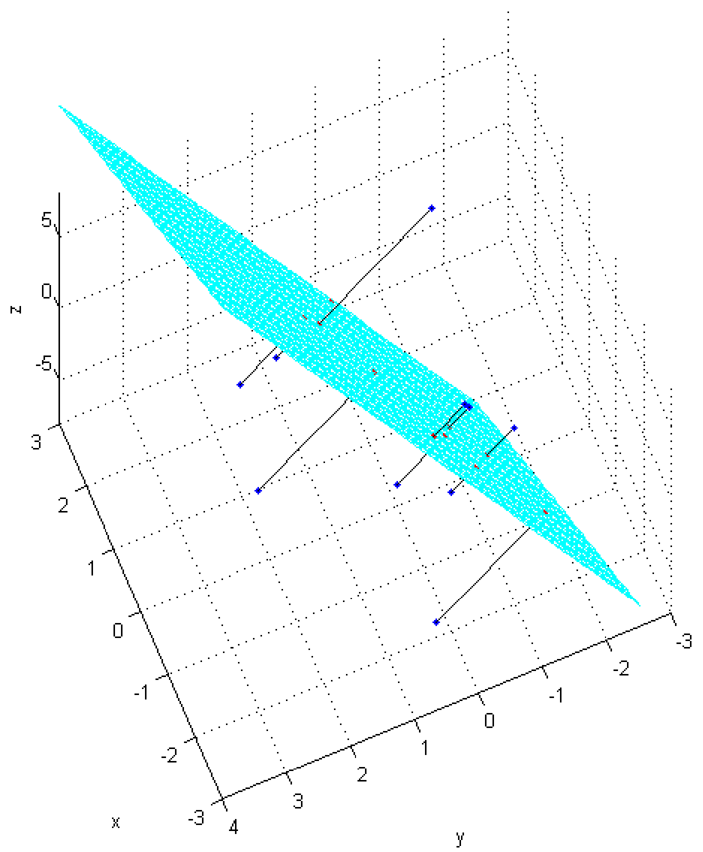
$$-2 < x_i < 2, \qquad -2 < y_i < 2, \qquad -8 < z_i < 6$$

Afficher (en bleu) les `n` points P_i dans le même repère que le plan créé et affiché à la question 1.

Indication : ne pas faire de boucle `for`, utiliser les commandes `rand` et `plot3`.

3) On note Q_i les images des points P_i par la projection orthogonale sur le plan.

- Construire la matrice de projection orthogonale sur le plan puis calculer et afficher (en rouge) les `n` points Q_i (utiliser les commandes `eye` et `norm`).
- Relier deux à deux les points P_i et Q_i (en d'autres termes afficher les `n` segments $[P_iQ_i]$, cf. figure ci-dessous).



4) Perturbation des points P_i

- Apporter une perturbation aléatoire d'amplitude `delta=0.1` sur les coordonnées des `n` points Q_i (cela signifie que l'on ajoute aux coordonnées un nombre aléatoire compris entre `-delta/2` et `delta/2`) ; on obtient alors `n` nouveaux point que l'on notera R_i.
- Déterminer l'équation du plan passant au plus près des `n` points R_i (indication : méthodes des moindres carrés)
- Calculer l'angle α que fait ce nouveau plan avec le 1er plan de projection (utiliser la commande `acos`)
- Afficher le nouveau plan (s'inspirer du code de la question 1 en changeant toutefois les couleurs pour que l'on puisse facilement le distinguer du plan précédent)
- Comment évolue l'angle α lorsqu'on fait varier l'amplitude de la perturbation de 0 à 1 ?
  - ajouter quelques lignes à votre programme de sorte à ce que l'on puisse facilement visualiser cette évolution
  - traiter les cas d'une perturbation aléatoire et d'une perturbation déterministe de votre choix

Exercice 8 : translation, symétrie, rotation … et feuilles d'automne

Cet exercice est très librement inspiré d'un exercice proposé par Jean-Marie Becker, ancien responsable des enseignements de mathématiques à CPE.

Copier et exécuter le programme suivant :

```
clear variables;
close all;

% motif de base
F0=2*[-0.5,-0.5,-5,-3,-10,-8,-9,-6,-6,-2,-5,-2,0,2,5,2,6,6,9,8,10,3,5,0.5,0.5;  % abscisses
       0,10,9,12,17,17,20,20,22,17,27,25,30,25,27,17,22,20,20,17,17,12,9,10,0]; % ordonnées
[m,n]=size(F0);

% dimension du cadre d'affichage
L=200;
cadre=[-L,L,-L,L];

% affichage du motif de base
color=[0,0,0];
fill(F0(1,:),F0(2,:),color);
axis('equal');
axis(cadre);
hold on;
```

Ce programme affiche un motif (une feuille d'arbre de couleur noire) dans une fenêtre. Les données du motif (abscisses et ordonnées des points) sont enregistrées dans une matrice à 2 lignes notée F_0 : la 1ère ligne donne les abscisses et la 2ème donne les ordonnées.

<u>Conseil</u> : Introduire à la suite du code les quelques lignes suivantes …

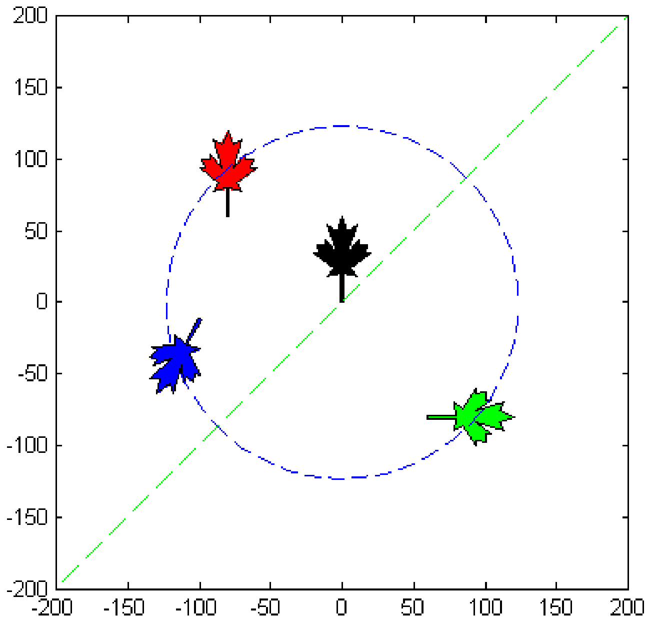
```
question=1;
switch question
    case 1
        …
        …
    case 2
        …
        …
    case 3
        …
        …
end
```

… qui vous permettront d'exécuter votre programme de façon sélective selon que vous voulez répondre à la question 1, à la question 2 ou à la question 3.

1) Translation, symétrie, rotation

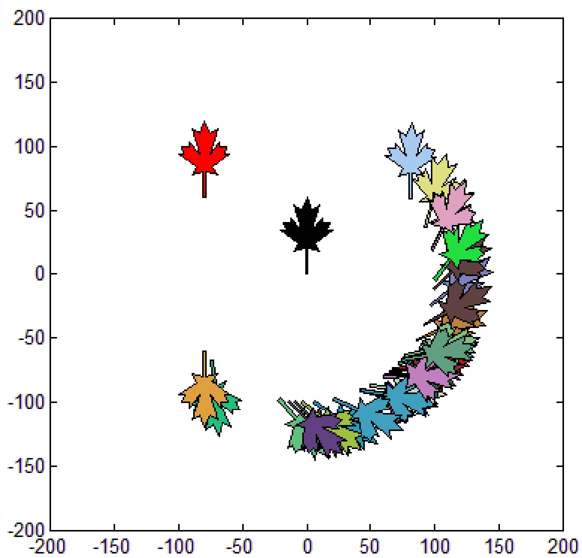
- on note F_1 l'image de F_0 par la translation de vecteur $\mathbf{u} = (-80, 60)^T$; calculer et afficher F_1 en rouge
- on note F_2 l'image de F_1 par la symétrie par rapport à la droite de vecteur directeur unitaire $\mathbf{N} = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)^T$; calculer et afficher F_2 en vert (indication : matrice de symétrie). On tracera également l'axe de symétrie.
- on note F_3 l'image de F_2 par la rotation de centre O et d'angle $-2\pi/3$; calculer et afficherF_3 en bleu (indication : matrice de rotation). On tracera également ce cercle de centre 0 de rayon $r = \|\overrightarrow{0G_2}\|$ où G_2 est le centre de gravité de F_2 (utiliser les commandes `mean` et `norm`)

L'exécution du programme donne :



2) Afficher $N = 30$ feuilles de sorte que chacune d'entre elles soit l'image de F_1 par une symétrie par rapport à une droite passant par l'origine et dont le vecteur directeur (unitaire) possède des composantes positives choisies de façon aléatoire (indications : matrice de symétrie, commande `rand`). La couleur de chaque feuille est également choisie aléatoirement.

L'exécution du programme donne :



Expliquer pourquoi les feuilles se répartissent sur un demi-cercle.

3) Afficher $N = 300$ feuilles de sorte que chacune d'entre elles soit l'image de F_1 par une rotation d'angle θ choisi aléatoirement entre 0 et 2π, suivie d'une translation de vecteur $\mathbf{u}$ dont les composantes sont choisies aléatoirement dans l'intervalle $[-L,\ L]$. La couleur des feuilles doit être choisie aléatoirement dans une gamme de tonalité automnale.

L'exécution du programme donne :