

2 Introduction au calcul matriciel et à Matlab.

2.1 Introduction à Matlab

1. Entrer un nombre; exemple

```
>>1  
ans 1  
>>1;  
pas de réponse : c'est le rôle de;
```

Donc ; n'affiche pas le resultat

2. % = mise en commentaire exemple:

```
>>1+3 % exemple d'addition
```

% commentaire

3. Entrer un nombre complexe avec j ou i, exemples :

```
>>1+2j  
>>1+2*j  
>>1+2i  
conclusion * n'est pas nécessaire. i et j sont interprétés comme i  
exemples supplémentaires :  
>>(1+2j)*(1-2j)  
>> a=1+i  
>> b=1-i  
>> a*b  
>> conj(a) donne le complexe conjugué de a  
>> abs(a) donne le module de a  
>> angle(a) donne l'argument de a en radians  
>> angle(a)* 180/pi donne l'argument de a en degrés
```

i ou j ==> Nombres complexes

4. >> `help angle` fournit une aide sur la fonction angle.
La fonction "help" permet d'obtenir de l'aide sur toute fonction de Matlab.
A chaque fonction de Matlab (ex. angle) correspond un fichier .m (ex. angle.m).
La commande >>`edit angle` ouvre directement le fichier `angle.m`.
Comprendre le contenu de cette fonction et comparer les commentaires en début de ce fichier avec le contenu de l'aide de cette fonction.

help retourne les lignes de commentaire du fichier .m

5. Essayer le programme “why”

- (a) `>>why` exécute le programme
- (b) `>>help why` affiche de l’aide sur le programme
- (c) `>>edit why` affiche le fichier `why.m`

Why → documentation

6. Calculer le module et l’argument des nombres complexes suivants :

- (a) $2+5j$
- (b) $2-5j$
- (c) 2
- (d) $5j$

Un peu de théorie sur le module et l’argument :

1.1 Définition et représentation

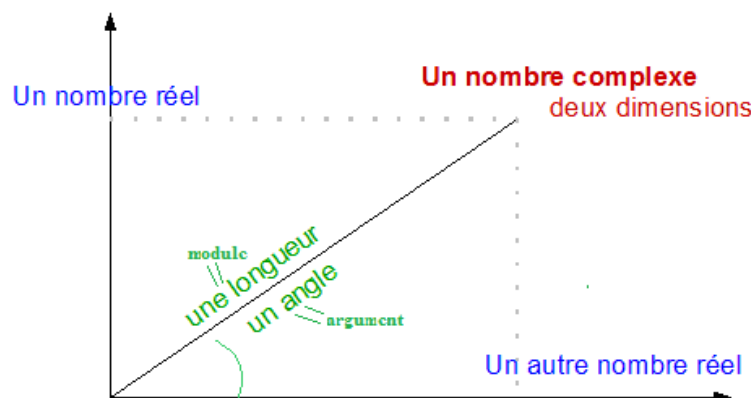
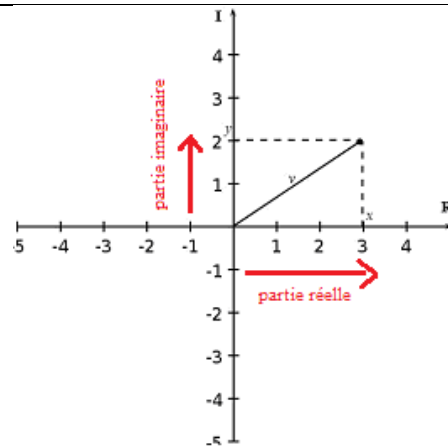
Les nombres complexes sont des nombres composés d’une partie réelle et d’une partie imaginaire. Soit z un nombre complexe, alors z peut s’écrire

$$z = x + yi ,$$

ou x représente la partie réelle et yi représente la partie imaginaire avec i le nombre imaginaire (rappel : $i = \sqrt{-1}$). Le complémentaire de z , c’est à dire son binôme conjugué, noté \bar{z} s’écrit

$$\bar{z} = x - yi .$$

Les nombres complexes peuvent être représentés géométriquement à l’aide d’un repère cartésien dans \mathbf{R}^2 . Dans ce cas, l’abscisse représente la partie réelle et l’ordonnée la partie imaginaire.



Aidé par cette représentation géométrique, on trouve trivialement que l'expression du **module de z** est donné par

$$|z| = \sqrt{x^2 + y^2} \quad (\text{Pythagore})$$

carré de la longueur de l'hypoténuse est égal à la somme des carrés des longueurs des deux autres côté

Parenthèse sur l'utilité des nombre complexes :

Je peux donner un exemple dans mon domaine : l'électronique

En électronique, les nombres complexes sont très utiles car ils permettent de simplifier les calculs sur les signaux sinusoïdaux (or tout signal peut être décomposé en une somme de signaux sinusoïdaux, donc l'étude des signaux sinusoïdaux est un point clef en électronique)

L'**amplitude** du signal est représentée par le **module** du nombre complexe ; l'**argument**, quant à lui, représente la **phase** du signal.

OU

On notera que le module peut également être obtenu de la manière suivante,

$$|z| = \sqrt{z \cdot \bar{z}}$$

En effet,

$$z \cdot \bar{z} = (x + yi)(x - yi) = x^2 + y^2 + xyi - xyi.$$

Donc,

$$z \cdot \bar{z} = x^2 + y^2 = |z|^2.$$

Electronique [\[modifier\]](#)

La **phase** peut être vue aussi comme un angle, dans le plan complexe, entre la partie imaginaire (Im) et la partie réelle (Re). C'est-à-dire **l'argument** de l'onde. Elle est donc calculée par : $\varphi = -\arctan\left(\frac{Im}{Re}\right)$.

Cette formulation est vue dans le **diagramme de Bode**, par exemple.

$$\varphi = -\arctan\left(\frac{Im}{Re}\right)$$



b

a

Donc lorsque l'on nous donne la **partie réelle et imaginaire** d'un **nombre complexe** ($Z = a + ib$) il est facile de retrouver le **module (une longueur)** et l'**argument (un angle)** qui sont des éléments de la **représentation d'un nombre complexe sous le forme cartésienne**.

Maintenant la pratique :

Pour calculer le **module** sur matlab : **abs(complexe)** et pour l'**argument** **angle(complexe)**.

a. Module et argument de 2+5j

```
abs(2+5j) = 5.3852      %module
angle(2+5j) = 1.1903    %argument
```

b. Module et argument de 2-5j

```
abs(2-5j) = 5.3852      %module
angle(2-5j) = -1.1903   %argument
```

c. Module et argument de 2

```
abs(2) = 2              %module
angle(2) = 0             %argument
```

d. Module et argument de 5j

```
abs(5j) = 5             %module
angle(5j) = 1.5708       %argument
```

7. Calculer les parties réelles et imaginaires des nombres complexes suivants :

(a) $5e^{j\alpha}$ avec $\alpha = 25^\circ$ (rappel : $e^{j\alpha} = \cos \alpha + j \sin \alpha$)

(b) $5e^{j\alpha}$ avec $\alpha = 25^\circ$

(c) vérifier que ces deux dernières valeurs sont égales

(d) faire de même avec $7e^{j167^\circ}$

Théorie sur la forme trigo. et expo. d'un complexe :

Forme trigonométrique d'un nombre complexe.

Soit M le point du plan complexe, image de $z = x + iy$.

Le module du complexe non nul $z = x + iy$, noté $|z|$, est le réel positif $|z| = \sqrt{x^2 + y^2}$.

$|z|$ est la distance entre l'origine O du repère et le point M (z). (On note parfois $|z|$ sous la forme r)

L'argument de z, noté $\arg z$, est une mesure de l'angle orienté $\theta = (\vec{u}, \vec{OM})$ en radians.

On a : $x = |z| \cos \theta$, et $y = |z| \sin \theta$.

L'écriture $z = |z| (\cos \theta + i \sin \theta)$ est l'expression trigonométrique de z.

==> On va donc pouvoir travailler sur la **représentation trigonométrique des complexes** et non plus sur l'**axe orthonormé (forme cartésienne)**.

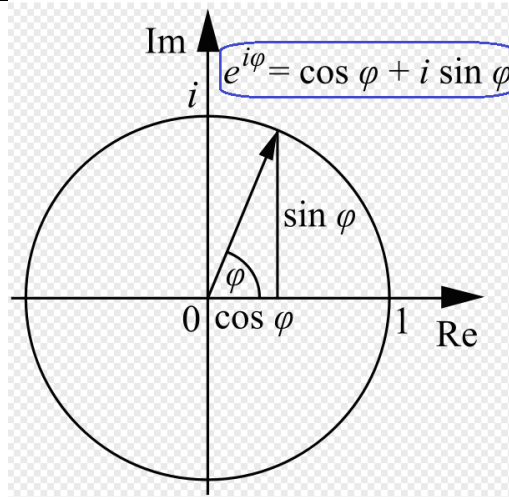
Forme exponentielle d'un nombre complexe.

Formule d'Euler

La très célèbre formule d'Euler permet d'établir la relation existant entre l'exponentielle complexe et les fonctions sinus et cosinus. Elle est de la forme

$$e^{ix} = \cos(x) + i \sin(x)$$





Cependant, il est intéressant de comprendre d'où vient cette formule, dans les grandes lignes tout du moins. Pour rappel, une des définition de l'exponentielle nous donne,

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Dès lors,

$$e^{yi} = \sum_{n=0}^{\infty} \frac{(yi)^n}{n!} = \sum_{n \text{ pair}} (-1)^{n/2} \frac{y^n}{n!} + i \sum_{n \text{ imp.}} (-1)^{\frac{n-1}{2}} \frac{y^n}{n!} = \cos(y) + i \sin(y)$$

De la formule d'Euler il découle les relations suivantes :

$$\cos(x) = \frac{e^{ix} + e^{-ix}}{2} ; \sin(x) = \frac{e^{ix} - e^{-ix}}{2i}$$



Transformation degrés en radians et inversement :

Matlab a besoin d'angle en radian donc si on nous fournis l'angle en degré dans l'énoncé on doit le convertir en radian.

Angle en radian = (angle en degré) * pi / 180

Angle en degré = (angle en radian) * 180 / pi

Maintenant la pratique :

- a. $5cjs\alpha$ avec $\alpha = 25^\circ$ (Rappel : $cjs\alpha = \cos\alpha + j \sin\alpha$)
==> forme trigo d'un complexe

$$5 * \cos((25 * \pi) / 180) = 4.5315 \quad ==> \text{partie réelle}$$

$$5 * j * \sin((25 * \pi) / 180) = 2.1131 \quad ==> \text{partie imaginaire}$$

$$5 * (\cos((25 * \pi) / 180) + j * \sin(25 / (180 * \pi))) = 4.5315 + 2.1131i$$

- b. $5ej\alpha$ avec $\alpha = 25^\circ$ ==> forme exponentielle (Euler) sous matlab

$$5 * (\exp(j * ((25 * \pi) / 180))) = 4.5315 + 2.1131i$$

- c. Donc ==> $X cjs\alpha = X eja$

La **forme exponentiel** ($ej\alpha$) permet de retrouver la **forme algébrique** ($a+jb$) tout comme la **forme trigonométrique** ($cjs\alpha$) d'un **nombre complexe**.

Il s'agit là des 3 formes que peut avoir un nombre complexe.

- d. $7cjs167^\circ$

$$7 * \cos((167 * \pi) / 180) = -6.8206 \quad ==> \text{partie réelle}$$

$$7 * j * \sin((167 * \pi) / 180) = 1.5747 \quad ==> \text{partie imaginaire}$$

$$7 * (\cos((167 * \pi) / 180) + j * \sin((167 * \pi) / 180)) = -6.8206 + 1.5747i$$

OU en forme expo

$$7 * (\exp(j * ((167 * \pi) / 180))) = -6.8206 + 1.5747i$$

8. >>t=1 :10 définira un vecteur composé des nombres 1 à 10
>>s=1 :0.1 :10 définira un vecteur composé des nombres de 1 à 10 mais avec un pas de 0.1

9. >>who
>>whos
Interpréter les résultats obtenus par ces deux intructions
Constater que t est est une matrice de 1 ligne et 10 colonnes (1 x 10) et s est une matrice de 1 ligne et 91 colonnes (1 x 91)

Whos donne donc une description d'une variable

10. Vous pouvez visualiser les différentes composantes de t comme suit :

- (a) `t(1,5)` donnera le 5ème élément du vecteur t càd 5, `s(1,25)` donnera le 25ème élément de s càd 3.4
- (b) en cliquant sur l'élément correspondant dans la zone "workspace".
- (c) essayer de taper `>>t(0,1)`
Sous Matlab, les indices d'un vecteur commencent par 1 contrairement au langage C, par exemple, où ces indices commencent par 0
Ce point sera une source d'erreur fréquente.

(a)

Je mets "`t=1:0.1:10`" donc un **vecteur** (*suite de chiffres sur une ligne*) de **nombre de 1 à 10** par **pas de 0.1** donc `t(1,25)` affiche la **25^{ème} valeur** de la **première ligne** (il n'y en a qu'une ici) c'est-à-dire **3.4**

(b) Il est possible de visualiser la matrice de l'élément en cliquant dessus dans **workspace**.

(c) **essayer de taper `>>t(0,1)`**

On essaye donc de visualiser la valeur de coordonnée ligne 0 et valeur 1 mais il n'y a pas de ligne 0 en matlab.

Effectivement, sous Matlab, les indices d'un vecteur commencent par 1 contrairement au langage C, par exemple, où ces indices commencent par 0.

11. La flèche vers le haut permet de récupérer une instruction précédente

12. Entrer `length(t)` et `size(t)`. Interpréter les résultats au moyen de l'aide de ces deux fonctions
Vérifier que `length(t)` est équivalent à `max(size(t))`

```
>> length(t)
```

```
ans =
```

```
10
```

```
>> size(t)
```

```
ans =
```

```
1 10
```

```
>> max(size(t))
```

```
ans =
```

```
10
```


Size renvoie donc les 2 éléments du vecteur (le nombre de lignes (une pour un vecteur) et le nombre de colonnes (donc d'éléments pour un vecteur))

Length lui renvoie la longueur donc le nombre d'éléments du vecteur qui correspond ici à 10. Maintenant si on avait fait l'erreur de définir notre vecteur ainsi : « t=0 :10 », « length » nous aurait renvoyé 11 ainsi que « max(size(t)) ».

Parcontre lorsqu'il s'agit d'une matrice, Length renvoi le nombre d'éléments des colonnes ou des lignes en fonction de là où il y en a le plus. Donc une matrice 3x4 length renvoie 4, une matrice 4x3 length renvoie 4.

13. Soit $x=[1 \ 3 \ 7 \ 9 \ 20 \ 40]$.
Que valent :
(a) $x(2)$
(b) $x(2 :4)$
(c) $x(2 :2 :6)$

```
>> x = [1 3 7 9 20 40]
```

```
x =
```

```
1    3    7    9   20   40
```

(a) $x(2)$

```
>> x(2)
```

```
ans =
```

```
3
```

(b) $x(2 :4)$

```
>> x(2 :4)
```

```
ans =
```

```
3    7    9
```

(c) $x(2 :2 :6)$

```
>> x(2 :2 :6)
```

```
ans =
```

```
3    9   40
```

14. Si les nombres sont affichés avec 5 décimales, le format sélectionné est "short"
get(0,'format') donne le format utilisé
Vous pouvez afficher les nombres avec 15 décimales en utilisant l'instruction :
>>format long
Visualiser le résultat par : >>a par ex.
>>format hex affichera les nombres au format hexadécimal
>>format rat affichera les nombres au format d'une fraction de petits nombres

TEST :

[illegible]

AVEC PI :

```
>> format short  
>> pi
```

```
ans =
```

```
3.1416
```

```
>> format long  
>> pi
```

```
ans =
```

```
3.141592653589793
```

```
>>
```

15. vecteurs, matrices et cellules :
Essayer les instructions suivantes :

VECTEURS

```
>> a=[1 2 3]
```

```
a =
```

```
1 2 3
```

```
b=[1,2,3]
```

```
b =
```

```
1 2 3
```

```
c=[1;2;3]
```

```
c =
```

```
1
```

```
2
```

```
3
```

Vecteur = une ligne ou une colonne

CONCATENATION

en colonnes:

```
d=[a,b]
```

d =

1	2	3	1	2	3
---	---	---	---	---	---

en lignes:

```
>> c=[1;2;3]
```

c =

1
2
3

```
>> e=[4;5;6]
```

e =

4
5
6

```
>> f=[c;e]
```

f =

1
2
3
4
5
6

ELEMENTS D'UN VECTEUR

```
>> d(1,3)
```

ans =

3

MATRICES

```
>> m=[1,2,3;4,5,6;7,8,9]
```

m =

1	2	3
4	5	6
7	8	9

Matrice = lignes et colonnes

```

ELEMENTS D'UNE MATRICE
>> m(2,3)
ans =
     6
> e=m(2,:)
e =
     4     5     6
>> m(:,3)
ans =
     3
     6
     9

```

```

CELLULES (cell array)
> A(1,1)={ [1 2;0 5;7 2] }
A =
    [3x2 double]
>> A(1,2)={'bonjour'}
A =
    [3x2 double]    'bonjour'
>> A(2,1)={3+i}
A =
    [3x2 double]    'bonjour'
    [3.0000+ 1.0000i]    []
>> A(2,2)={-1:0.2:1}
A =
    [3x2 double]    'bonjour'
    [3.0000+ 1.0000i]    [1x11 double]

```

Dans $A(x,y) \Rightarrow A$ est une matrice et x,y sont les coordonnées d'une cellule (donc $A(x,y)$ = cellule de matrice A) qui elle-même peut contenir des éléments (matrice, vecteurs).

```

ELEMENTS D'UNE CELLULE
>> A{1,2}
ans =
    bonjour

```

Exercices :

Que valent :

[1,2 : 3,4 : 5,6] ?

[1,2 4,4 : 5,6] ?

[1,2 : 3; 4, 5, 6] ?

[1,2 : 3,4 : 5,6] ➔ un vecteur

```
>> [1,2 : 3,4 : 5,6]
```

```
ans =
```

1	2	3	4	5	6
---	---	---	---	---	---

[1,2 4,4 : 5,6] ➔ toujours un simple vecteur

```
>> [1,2 4,4 : 5,6]
```

```
ans =
```

1	2	4	4	5	6
---	---	---	---	---	---

[1,2 : 3; 4, 5, 6] ➔ une matrice de 2 lignes et 3 colonnes

```
>> [1,2 : 3; 4, 5, 6]
```

```
ans =
```

1	2	3
4	5	6

16. Petite synthèse

(a) `[]` correspond à un vecteur.

Exemple, `[1,2]` ou `[1 2]` correspondent au vecteur ligne 1 2
virgule (,) ou espace (space) sont des séparateurs de colonnes.

(b) Point-virgule (;) est un séparateur de lignes.

Par exemple, `[1;2]` correspond au vecteur colonne
1
2

(c) `{ }` correspond à une cellule.

par exemple, `{1, 'pomme'}` ou `{1 'pomme'}` correspondent à la cellule :

`[1] 'pomme'`
virgule (,) ou espace (space) sont des séparateurs de colonnes.

(d) Point-virgule (;) est un séparateur de lignes.

Par exemple : `{1; 'pomme'}` correspond au vecteur colonne
`[1]`
`'pomme'`

(e) `[1,2;3,4]` donnera :

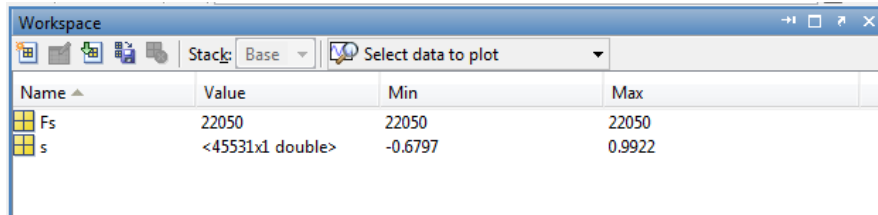
1 2
3 4

(f) `{1, 'bonjour'; 'au revoir', 2}` donnera :

`[1] 'bonjour'`
`'au revoir' [2]`

17. Nous allons vérifier qu'un son est un vecteur.

(a) Ouvrir le fichier chirac1.wav au moyen de l'instruction "wavread"
`>>[s,Fs]=wavread('chirac1.wav')`



The screenshot shows the MATLAB Workspace window. It has a toolbar with icons for workspace, command window, and other tools. Below the toolbar is a dropdown menu for 'Stack' set to 'Base' and another dropdown for 'Select data to plot'. A table lists the variables in the workspace:

Name	Value	Min	Max
Fs	22050	22050	22050
s	<45531x1 double>	-0.6797	0.9922

S = signal

Fs = Fréquence d'échantillonnage

(b) Ecouter ce son au moyen de l'instruction "sound"
`>>sound(s,Fs)`
Quelle est la valeur de Fs ?
Quelle est la valeur maximale et la valeur minimale de s ?

Ces valeurs sont déjà données à la lecture du fichier (voir ci-dessus).

(c) L'instruction `>>sound(s)` ne donne pas un résultat correct. Expiquer pourquoi.

Si on ne donne pas la valeur de la fréquence (d'échantillonnage) le son ne sera pas lu à la bonne fréquence, ici il est lu au ralenti.

(d) Ecrire un vecteur qui prend les 100 premiers éléments de ce son et écouter le résultat.
Pourquoi n'entendez-vous rien ?

```
V = s(1 : 100);  
sound(V,Fs);
```

Le signal est un vecteur allant de 1 à 45531, 100 représente donc une valeur minime que l'on n'a pas le temps d'entendre.

Si maintenant j'écris un vecteur prenant les 5000 premiers éléments, on peut écouter 1/5 du son.

```
>> V = s(1 : 5000);  
>> sound(V,Fs);
```


(e) Ecrire un vecteur qui prend les éléments 10000 à 20000 de ce son en divisant l'amplitude par 4 : écouter le résultat.

```
>> V = s(10000 :20000);  
>> sound(V/4,Fs);
```

Comme on s'en doute...le son est moins fort !

(f) Ecrire un vecteur qui est la concaténation du vecteur d'indices 1 à 10000, du vecteur d'indices 10001 à 20000 et du vecteur d'indice 20001 jusqu'à la fin du fichier wav.
Ecouter le résultat obtenu.
note : La concaténation des vecteurs a et b est obtenue par [a,b]



On obtient une erreur, en effet il n'est pas possible de concaténer des vecteurs qui n'ont pas le même nombre d'éléments :

```
>> [s,Fs]=wavread('chirac1.wav');  
>> a = s(1:10000);  
>> b = s(10001:20000);  
>> c = s(20001:end);  
>> V = [a,b,c];  
??? Error using ==> horzcat  
CAT arguments dimensions are not consistent.
```

Du coup il faut faire :



```
>> [s,Fs]=wavread('chirac1.wav');  
>> a = s(1:10000);  
>> b = s(10001:20000);  
>> c = s(20001:30000);  
>> V = [a,b,c];  
>> sound(V,Fs);  
??? Error using ==> playsnd  
Data must have one or two columns.
```

```
Error in ==> sound at 58  
playsnd(y,fs,bits);
```

 V	<10000x3 double>	-0.6797	0.9922
 s	<45531x1 double>	-0.6797	0.9922

Autre erreur ! On concatène les 3 parties du signal dans des colonnes différentes ! Du coup :

```
>> V = [a; b; c];
>> sound(V,Fs);
```

 V	<30000x1 double>	-0.6797	0.9922
 s	<45531x1 double>	-0.6797	0.9922

On obtient une grande partie du signal de base en concaténant 3 de ses parties en une colonne comme il l'était à la base. On entend donc bien la majeure partie du son.

(g) Existe-t-il une méthode plus simple pour obtenir le résultat souhaité? Si oui appliquez-la.

```
>> [s Fs]=wavread('chirac1.wav');
>> tailleS = length(s)
```

```
tailleS =

    45531
```

```
>> tailleV = round(tailleS/3)
```

```
tailleV =

    15177
```

```
>> a = s(1:15177);
```

```
>> 15176+15178
```

```
ans =

    30354
```

```
>> b = s(15178:30354);
```

```
>> 15176+30355
```

```
ans =

    45531
```

```
>> c = s(30355:45531);
```

```
>> V = [a;b;c];
```

```
>> sound(V,Fs)
```

Ici j'ai concaténé 3 vecteurs du signal et j'obtiens tout le signal de base ! J'ai juste divisé la taille du signal par 3 et chaque'un des 3 vecteurs pourra donc avoir la même taille et donc être concaténé et garder l'entièreté du signal. **Le vert sont juste des calculs intermédiaire pour trouver les valeurs des vecteurs.**

18 Comment obtenir de l'aide sous Matlab?

(a) Par l'onglet Help du menu

(b) Par l'instruction "help" de Matlab suivie de la fonction dont vous désirez de l'aide

(c) Si vous taper `>>an` suivi de deux "tab", le nom de toutes les fonctions existantes qui commencent par "an" s'afficheront.

(d) Matlab fournit également plusieurs fichiers au format pdf décrivant les notions et les instructions utilisées.

2.2 Programmation sous Matlab

1. Dans Matlab, il ne faut pas définir le type d'une variable avant de l'utiliser.

2. Attention. Matlab distingue les majuscules et les minuscules : les variables A et a sont différentes. Faire le test en définissant, par exemple, une variable a dont la valeur vaut 2 et une variable A dont la valeur vaut 5.

3. Matlab ne contient pas de structure de pointeur

4. Matlab est un outil de programmation complet : boucles for, while, if...

- (a) for nn=1 :10 instructions; end;
- (b) while condition instructions; end;
- (c) if condition1 instructions; elseif condition2 instructions; else instructions; end;
- (d) Pour les conditions, on utilisera :
< <= > >= == ~= les deux dernières expressions sont utilisées pour "est égal à" ou "n'est pas égal à".
Les opérateurs logiques
& | et ~
sont utilisés pour ET, OU et NON.

2.3 Editer et exécuter un programme .m

1. Editer votre programme :
dans le menu file sélectionner new puis sélectionner M-file
2. Sauver votre programme
3. Exécuter votre programme
dans le menu debug sélectionner run
4. Les erreurs de compilation sont affichées dans la fenêtre de commande de Matlab.

5. Vous pouvez exécuter un programme .m pas à pas en procédant de la manière suivante :

- (a) Insérer un "breakpoint" par un clic gauche à côté du numéro de la ligne correspondante.
- (b) Cliquer sur run dans le menu debug. Le programme s'exécutera jusque cette ligne.
- (c) Cliquer sur step dans le menu debug ou appuyer sur F10

6. Utilisation d'un programme existant.

- (a) charger le programme `plot_stem.m`
- (b) exécuter ce programme
- (c) placer un breakpoint à la ligne 4 puis réexécuter ce programme pas à pas.

7. Ecrire, au moyen d'une boucle un programme qui affiche le carré des nombres de 10 à 20.

Exécuter ce programme pas à pas.

```
clear all;  
close all;  
clc;  
  
for i = 10 : 1 : 20  
    k=i*i;  
end;
```

8. Ecrire un programme qui donne le sinus d'un angle de 0° à 360° avec un pas de 10°.

```
clear all;  
close all;  
clc;  
  
for a = 0 : 10 : 360;% vecteur commence à 0°(pas la règle du v. à 1)  
    y = sin((a*pi)/180); %transformer les degrés en radians  
end;
```

9. Les deux derniers exercices sont des cas typiques où les boucles sont à éviter à cause de leur lenteur sous matlab.
Il sera de loin préférable d'utiliser les outils vectoriels comme l'opération `.`[^]
Ceci sera vu plus loin.

10. Lancer un exécutable à partir de Matlab.
Pour ce faire, on utilisera la commande :
`>> !nom_de_l'exécutable &`
Exercice :
– écrire un programme en C qui affiche “Bonjour tout le monde”. Construire son exécutable.
– exécuter ce programme à partir d'un script Matlab.

Perso, j'ai un peu la flemme de le faire...

2.4 Calcul matriciel et complexe par Matlab

1. Créer la matrice $M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
1 2 sera la première ligne que l'on peut retrouver par $M(1, :)$
3 4 sera la deuxième ligne que l'on peut retrouver par $M(2, :)$
 $M(:, 1)$ donnera la première colonne

```
>> M=[1 2 ;3 4]
```

```
M =
```

```
     1     2  
     3     4
```

```
>> M(1, :)
```

```
ans =
```

```
     1     2
```

```
>> M(2, :)
```

```
ans =
```

```
     3     4
```

```
>> M(:, 1)
```

```
ans =
```

```
     1  
     3
```

2. Créer la matrice N dont la première ligne vaut 2 1 et la deuxième ligne vaut 1 -1
on pourra utiliser N(1,:)= [2 1]

```
>> N(1, :)= [2 1]
```

```
N =
```

```
2    1
```

```
>> N(2, :)= [1 -1]
```

```
N =
```

```
2    1
1   -1
```

3. calculer M*N et vérifier votre résultat "à la main".

Théorie sur multiplication de 2 matrices:

$$\text{if } A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \text{ and } B = \begin{bmatrix} e & f \\ g & h \end{bmatrix} \text{ then } {}^{[1]}AB = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

MATLAB :

```
>> M*N
```

```
ans =
```

```
4    -1
10   -1
```

A LA MAIN :

$$\begin{array}{cc} 1a & 2b \\ 3c & 4d \end{array} * \begin{array}{cc} 2e & 1f \\ 1g & -1h \end{array} = \begin{array}{cc} ae & bg \\ ce & dg \end{array} \begin{array}{cc} af & bh \\ cf & dh \end{array}$$

$$\begin{array}{cc} 1*2 + 2*1 & 1*1 + 2*(-1) \\ 3*2 + 4*1 & 3*1 + 4*(-1) \end{array}$$

4. Calculer N*M et constater la différence avec le résultat précédent

```
>> N*M
```

```
ans =
```

```
    5    8
   -2   -2
```

$$\begin{pmatrix} 2a & 1b \\ 1c & -1d \end{pmatrix} * \begin{pmatrix} 1e & 2f \\ 3g & 4h \end{pmatrix} = \begin{pmatrix} ae & bg \\ ce & dg \end{pmatrix} = \begin{pmatrix} 2*1 + 1*3 & 2*2 + 1*4 \\ 1*1 + (-1)*3 & 1*2 + (-1)*4 \end{pmatrix}$$

5. Créer la matrice O constituée de la colonne -2 1. Calculer O*M et vérifier votre calcul "à la main"

```
>> O = [-2;1]
```

```
O =
```

```
   -2
    1
```

```
>> O*M
```

```
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Le nombre d'éléments des lignes de A doit être le même que le nombre d'éléments des colonnes de B. Si ce n'est pas le cas, le produit A*B n'est pas défini (ou est indéfini), comme nous pouvons le voir dans l'exemple suivant

$$\begin{pmatrix} -3 & 3 & 2 \\ 1 & -4 & 3 \\ -3 & 0 & -5 \end{pmatrix} \cdot \begin{pmatrix} -2 & 3 \\ 1 & -8 \end{pmatrix} \text{ n'est pas défini!}$$

CCL : Si les matrices n'ont pas soit le même nombre de colonnes ou de lignes, il est impossible de les multiplier.

6. Créer une matrice M 4x4 avec l'instruction ones
remplacer la deuxième colonne par des 2 sachant que M(:,n) donne la n^{ème} colonne
remplacer 3^{ème} ligne du résultat par des 3 et afficher le résultat.

```
>> help ones
ONES    Ones array.
        ONES(N) is an N-by-N matrix of ones.
        ...
```

```
>> N = ones(4)
```

```
N =
```

```

     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

Remplacer la deuxième colonne par des 2 sachant que M(:,n) donne la n^{ème}

```
>> N(:,2)=2
```

```
N =
```

```

     1     2     1     1
     1     2     1     1
     1     2     1     1
     1     2     1     1
```

Remplacer la 3^{ème} ligne par des 3

```
>> N(3,:)=3
```

```
N =
```

```

     1     2     1     1
     1     2     1     1
     3     3     3     3
     1     2     1     1
```

7. Créer une matrice 4x4 formée de 0 partout sauf sur la diagonale principale sur laquelle il y aura des 1. On appellera cette matrice I (cette matrice peut directement être créée par l'instruction `eye(m,n)`)

```
>> I = eye(4,4)
```

```
I =
```

```
1    0    0    0
0    1    0    0
0    0    1    0
0    0    0    1
```

```
>> help eye
```

```
EYE Identity matrix.
```

```
EYE(N) is the N-by-N identity matrix.
```

```
EYE(M,N) or EYE([M,N]) is an M-by-N matrix with 1's on  
the diagonal and zeros elsewhere.
```

```
...
```

8. Calculer $M \cdot I$ et $I \cdot M$ conclusion

```
>> I*N
```

```
ans =
```

```
1    2    1    1
1    2    1    1
3    3    3    3
1    2    1    1
```

```
>> N*I
```

```
ans =
```

```
1    2    1    1
1    2    1    1
3    3    3    3
1    2    1    1
```

La **matrice identité** est une [matrice](#) carrée avec des 1 sur la diagonale et des 0 partout ailleurs.

CCL : Même résultat lorsque l'on multiplie une matrice par une matrice identité (lorsque l'une des 2 est une matrice `EYE(N,M)`).

9. Calculer $F=M+I$ puis F^{-1} , $F \cdot F^{-1}$ conclusion ?

(pas s'étonner que j'mettes des N j'ai inversé les 2 matrices dans matlab)

```
>> F = N+I
```

```
F =
```

2	2	1	1
1	3	1	1
3	3	4	3
1	2	1	2

```
>> F^(-1)
```

```
ans =
```

0.9091	-0.4545	-0.0909	-0.0909
-0.0909	0.5455	-0.0909	-0.0909
-0.5455	0.2727	0.4545	-0.5455
-0.0909	-0.4545	-0.0909	0.9091

```
>> F*F^(-1)
```

```
ans =
```

1.0000	-0.0000	-0.0000	0.0000
0.0000	1.0000	0	0.0000
0.0000	0	1.0000	0
0.0000	-0.0000	0	1.0000

Conclusion : on peut retrouver « I » en multipliant F par son inverse ($1/F$).

10. Calculer M^{-1} : toute matrice ne possède pas une inverse (car ici, 3 colonnes sont identiques)

```
>> N
```

```
N =
```

```
1 2 1 1
1 2 1 1
3 3 3 3
1 2 1 1
```

```
>> N^(-1)
```

```
Warning: Matrix is singular to working precision.
```

```
ans =
```

```
Inf Inf Inf Inf
Inf Inf Inf Inf
Inf Inf Inf Inf
Inf Inf Inf Inf
```

Dès que minimum 2 colonnes sont identiques la matrice n'est pas inversible (déterminant = 0).

Exemple de calcul d'inverse d'une matrice 2 x 2

On veut calculer la matrice inverse de

$$A = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}$$

Première étape : on calcule le déterminant de la matrice A :

$$\det A = \begin{vmatrix} 2 & 3 \\ 4 & 5 \end{vmatrix} = 10 - 12 = -2$$

Le déterminant de cette matrice n'est pas nul donc la matrice A est inversible

Deuxième étape :

$$A^{-1} = \frac{1}{-2} \begin{pmatrix} 5 & -3 \\ -4 & 2 \end{pmatrix} = \frac{-1}{2} \begin{pmatrix} 5 & -3 \\ -4 & 2 \end{pmatrix} = \begin{pmatrix} -\frac{5}{2} & \frac{3}{2} \\ 2 & -1 \end{pmatrix}$$

11. Créer une matrice C de 1 ligne et de 4 colonnes [0 1 2 3]. Calculer M*C et C*M pourquoi obtient-on une erreur avec M*C ?

Rappel N est une matrice 4x4.

```
>> C = [0 1 2 3]

C =

     0     1     2     3

>> N*C
??? Error using ==> mtimes
Inner matrix dimensions must agree.

>> C*N

ans =

    10    14    10    10
```

Le nombre d'éléments des lignes de A doit être le même que le nombre d'éléments des colonnes de B. Si ce n'est pas le cas, le produit A·B n'est pas défini (ou est indéfini), comme nous pouvons le voir dans l'exemple suivant

$$\begin{pmatrix} -3 & 3 & 2 \\ 1 & -4 & 3 \\ -3 & 0 & -5 \end{pmatrix} \cdot \begin{pmatrix} -2 & 3 \\ 1 & -8 \end{pmatrix} \text{ n'est pas défini!}$$

⇒ En effet dans **C*N**, C possède **4 éléments sur une ligne** et N possède **4 éléments sur une colonne**. Par contre dans **N*C**, N possède **4 éléments sur une ligne** mais C ne possède que **1 élément par colonne**.

CCL : Si les matrices ont soit le même nombre de colonnes, soit le même nombre de lignes, il est possible de les multiplier dans un sens, si elles ont exactement la même taille, il est possible de les multiplier dans les 2 sens. Si elles n'ont ni le même nombre de colonnes, ni le même nombre de lignes, il est impossible de les multiplier.

12. Transposée : calculer C' et $M \cdot C'$. Pourquoi n'obtient-on pas d'erreur en calculant $M' \cdot C'$?

attention : soit $D = \begin{bmatrix} 0 & 1 & 2+j & 3-j \end{bmatrix}$ D' donnera

0

1

2-j

3+j

['] donne la matrice transposée puis prend le complexe conjugué

```
>> C
```

```
C =
```

```
      0      1      2      3
```

```
>> C'
```

```
ans =
```

```
      0
```

```
      1
```

```
      2
```

```
      3
```

```
>> N*C'
```

```
ans =
```

```
      7
```

```
      7
```

```
     18
```

```
      7
```

```
>> N'*C'
```

```
ans =
```

```
     10
```

```
     14
```

```
     10
```

```
     10
```

Le nombre d'éléments des lignes de A doit être le même que le nombre d'éléments des colonnes de B. Si ce n'est pas le cas, le produit $A \cdot B$ n'est pas défini (ou est indéfini), comme nous pouvons le voir dans l'exemple suivant

$$\begin{pmatrix} -3 & 3 & 2 \\ 1 & -4 & 3 \\ -3 & 0 & -5 \end{pmatrix} \cdot \begin{pmatrix} -2 & 3 \\ 1 & -8 \end{pmatrix} \text{ n'est pas défini!}$$

⇒ En effet dans $N \cdot C'$, N possède **4 éléments sur une ligne** et C' possède **4 éléments sur une colonne**.

⇒ En effet dans $N' \cdot C'$, N' possède **4 éléments sur une ligne** et C' possède **4 éléments sur une colonne**.

CCL : Si les matrices ont soit le même nombre de colonnes, soit le même nombre de lignes, il est possible de les multiplier dans un sens, si elles ont exactement la même taille, il est possible de les multiplier dans les 2 sens. Si elles n'ont ni le même nombre de colonnes, ni le même nombre de lignes, il est impossible de les multiplier.

attention : soit $D = \begin{bmatrix} 0 & 1 & 2+j & 3-j \end{bmatrix}$ D' donnera

0
1
2-j
3+j

```
>> D = [ 0 1 2+j 3-j]
```

```
D =
```

0	1.0000	2.0000 + 1.0000i	3.0000 - 1.0000i
---	--------	------------------	------------------

```
>> D'
```

```
ans =
```

0
1.0000
2.0000 - 1.0000i
3.0000 + 1.0000i

13. $\gg D.'$ donnera la transposée seule

```
>> D.'
```

```
ans =
```

0
1.0000
2.0000 + 1.0000i
3.0000 - 1.0000i

Donc « D' » ne donne que la transposée de la matrice mais ne donne pas le complexe conjugué de « $2+i$ » ni « $3-i$ ».

14. Calculer $D \cdot D'$. Quelle différence avec $D' \cdot D$ et $D \cdot D'$?

```
>> D*D'
```

```
ans =
```

```
16
```

```
>> D*D.'
```

```
ans =
```

```
12.0000 - 2.0000i
```

```
>> D'*D
```

```
ans =
```

```

0          0          0          0
0          1.0000      2.0000 + 1.0000i  3.0000 - 1.0000i
0          2.0000 - 1.0000i  5.0000      5.0000 - 5.0000i
0          3.0000 + 1.0000i  5.0000 + 5.0000i  10.0000

```

Le nombre d'éléments des lignes de A doit être le même que le nombre d'éléments des colonnes de B. Si ce n'est pas le cas, le produit $A \cdot B$ n'est pas défini (ou est indéfini), comme nous pouvons le voir dans l'exemple suivant

$$\begin{pmatrix} -3 & 3 & 2 \\ 1 & -4 & 3 \\ -3 & 0 & -5 \end{pmatrix} \cdot \begin{pmatrix} -2 & 3 \\ 1 & -8 \end{pmatrix} \text{ n'est pas défini!}$$

- ⇒ En effet dans $D \cdot D'$, D possède **4 éléments sur une ligne** et D' possède **4 éléments sur une colonne**.
- ⇒ En effet dans $D' \cdot D$, D' possède **1 élément sur une ligne** et D possède **1 élément sur une colonne**.

???????? Dans $D \cdot D'$ on ne garde que la forme complexe, je ne sais pas pourquoi ????????????

15. Calculer M^2 et $M*M$ comparer

```
>> N*N
```

```
ans =
```

7	11	7	7
7	11	7	7
18	27	18	18
7	11	7	7

```
>> N^2
```

```
ans =
```

7	11	7	7
7	11	7	7
18	27	18	18
7	11	7	7

16. Calculer $M.^2$ et comparer au résultat précédent. Cette distinction est très importante

```
>> N.^2
```

```
ans =
```

1	4	1	1
1	4	1	1
9	9	9	9
1	4	1	1

On peut remarquer que chaque élément est élevé au carré.

Rappel :

```
>> N
```

```
N =
```

1	2	1	1
1	2	1	1
3	3	3	3
1	2	1	1

17. $x \cdot y$ donne une multiplication terme à terme. Si l'un des deux est scalaire, il multiplie tous les termes de l'autre sinon x et y doivent avoir la même dimension.

En tant que notion commune aux mathématiques et à la physique :

Un **scalaire** est une quantité qui ne possède pas de direction, à l'opposé des **vecteurs**, qui ont une direction.

La **multiplication** d'un **vecteur** par un **scalaire**, permet d'**agrandir**, ou de **contracter**, c'est-à-dire de **changer d'échelle**.

Un **scalaire** est souvent un nombre réel ou complexe.

18. Calculer $M \cdot I$ et $C \cdot M$ expliquer votre réponse

RAPPEL :

```
>> I
```

```
I =
```

```
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

Donc :

```
>> N.*I
```

```
ans =
```

```
1 0 0 0
0 2 0 0
0 0 3 0
0 0 0 1
```

```
>> N
```

```
N =
```

```
1 2 1 1
1 2 1 1
3 3 3 3
1 2 1 1
```

Ici aucun des 2 éléments de la multiplication terme à terme est un scalaire mais par contre ce sont des matrices de même taille/dimension.

```
>> C
```

```
C =
```

```
0 1 2 3
```

```
>> N
```

```
N =
```

```
1 2 1 1
1 2 1 1
3 3 3 3
1 2 1 1
```

```
>> C.*M
```

```
??? Error using ==> times
Matrix dimensions must agree.
```

Ici aucun des 2 éléments de la multiplication terme à terme est un scalaire et en plus de ça ce sont deux matrices de dimension différente (cfr 17.).

19. Interpréter length(variable) et size(variable) pour M,I,C et D

```
>> length(N)

ans =

     4

>> length(I)

ans =

     4

>> size(N)

ans =

     4     4

>> size(I)

ans =

     4     4

>> size(C)

ans =

     1     4
```

- Size renvoie donc le nombre de lignes et de colonnes.
- Length –lui- renvoi le nombre d'éléments des colonnes ou des lignes en fonction de là où il y en a le plus. Donc une matrice 3x4 length renvoie 4, une matrice 4x3 length renvoie 4.

20. Ecrire un programme qui affiche le carré des nombres de 10 à 20 sans utiliser de boucle.

```
>> s=10:20;
>> y=s.^2

y =

    100    121    144    169    196    225    256    289    324    361    400
```

Suffit d'utiliser les vecteurs et de calculer chacun de ses éléments au carré comme nous l'avons vu précédemment « .^2 » permet de le faire.

21. Nous allons vérifier qu'une image en niveaux de gris (8 bits) est une matrice de nombres entiers compris entre 0 et 255.

(a) Ouvrir l'image Lenna.bmp au moyen de l'instruction "imread"

```
>> img = imread('LENNA.BMP');
```

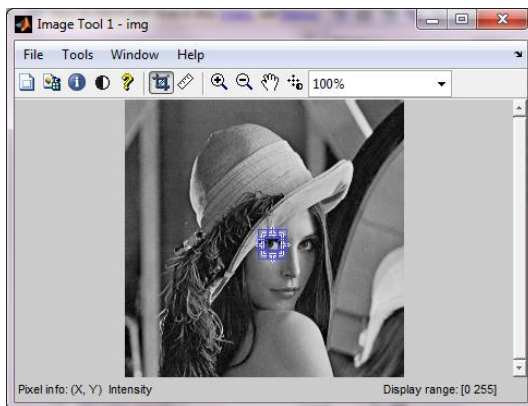
(b) Visualiser cette image au moyen de l'instruction "imshow"

```
>> imshow(img);
```

(c) Sélectionner un rectangle contenant uniquement un de ses yeux.

```
>> imtool(img);
```

(crop the image)



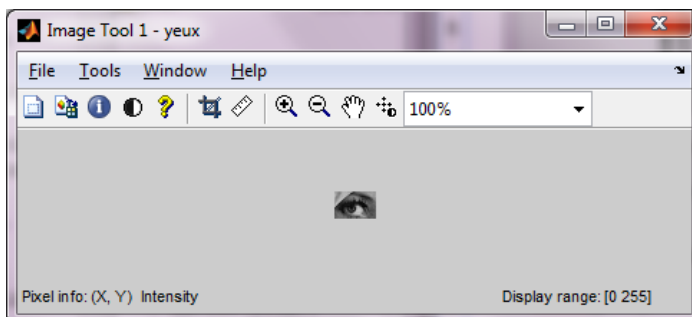
Oubien on s'amuse à prendre les coordonnées X et Y

==> (X les colonnes Y les lignes ==> inversé car sous forme d'axe orthonormé ou x=abscisse et y=ordonnée)

```
>> yeux = img(122:140,122:150);
```

Oubien on exporte la matrice de l'œil obtenue par le crop (option dans file).

```
>> imtool(yeux);
```



(d) Calculer la moyenne de la valeur de cette sélection

```
>> mean(oeil)

ans =

Columns 1 through 7

115.8421 110.1053 104.2105 98.3158 84.1579 78.2632 81.2105

Columns 8 through 14

82.1053 70.1579 72.5263 65.6316 73.2105 77.5263 89.6842

Columns 15 through 21

119.2105 125.7368 110.8947 110.4211 87.8947 77.6842 76.3158

>> mean(mean(oeil))

ans =

91.0050
```

La moyenne d'une matrice ne renvoie que la moyenne des éléments de chaque colonne, donc il faut faire une moyenne de cette moyenne pour obtenir la moyenne totale de la matrice.

(e) Ses yeux sont-ils plutôt clair ou foncés?

Vu que le niveau de gris le plus foncé est de 0 (noir) et le plus clair de 255 (blanc), 127.5 est le seuil pour savoir si l'image est claire ou foncée.

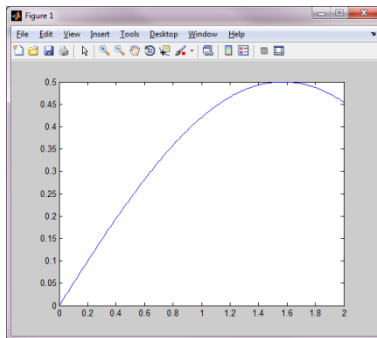
Ici elle est en dessous du seuil, l'image est donc foncée.

2.5 Dessiner avec Matlab

1. instruction plot : exemple

```
t=0 :0.01 :2 ;  
y=0.5*sin(t) ; plot(t,y)
```

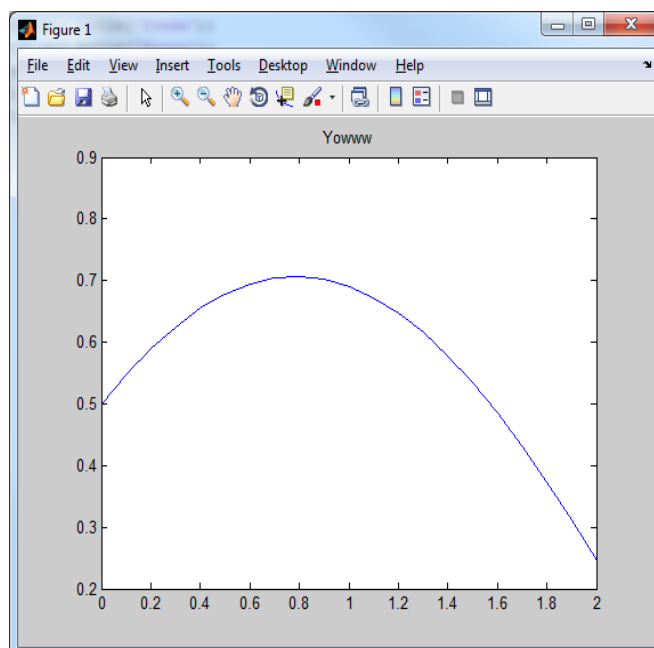
```
>> t=0:0.1:2;  
>> y=0.5*sin(t);  
>> plot(t,y);
```



2. remarques

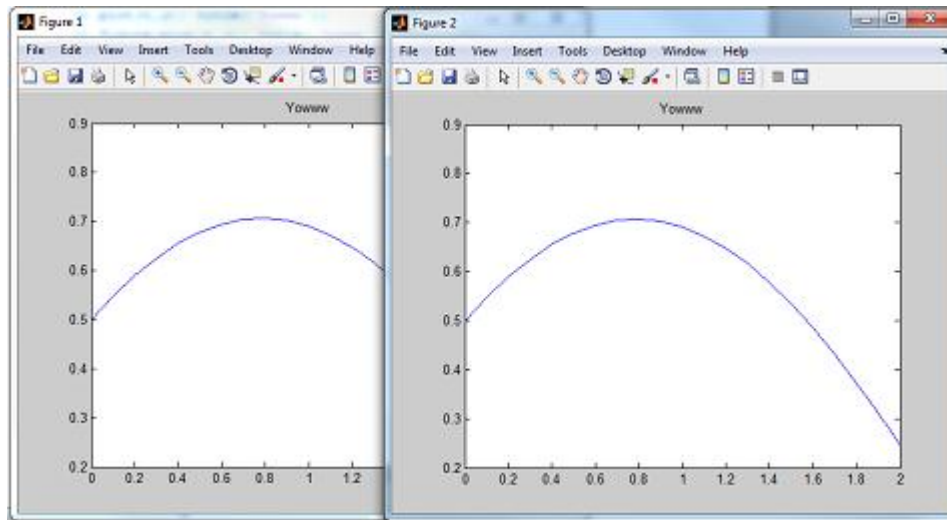
(a) après l'instruction `plot`, l'instruction
`>> title('titre du graphe')` permet de donner un titre au graphe

```
>> title('Yowww'); plot(t,y);
>> plot(t,y); title('Yowww');
```

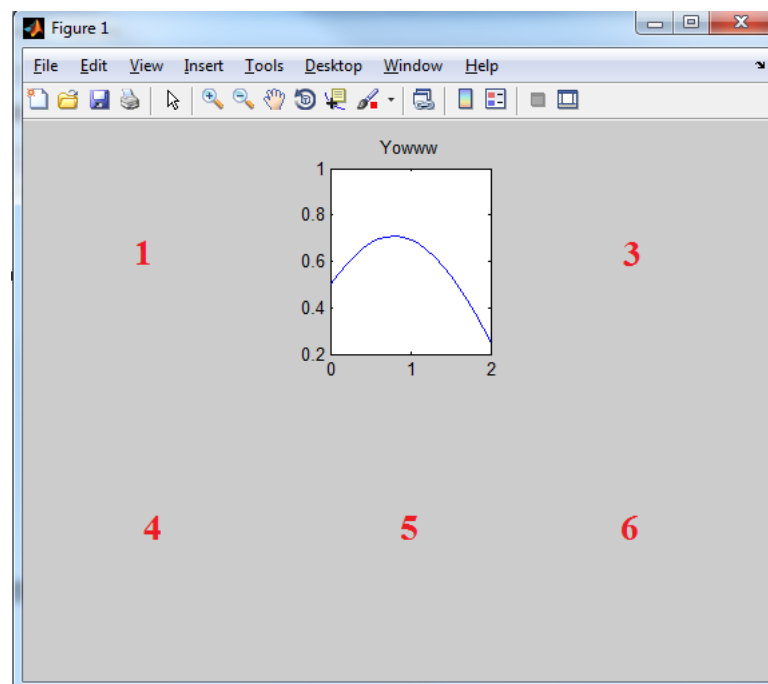


(b) l'instruction `>> figure` permet de dessiner le graphe sur une nouvelle figure

```
>> plot(t,y); title('Yowww');
>> figure; plot(t,y); title('Yowww');
```

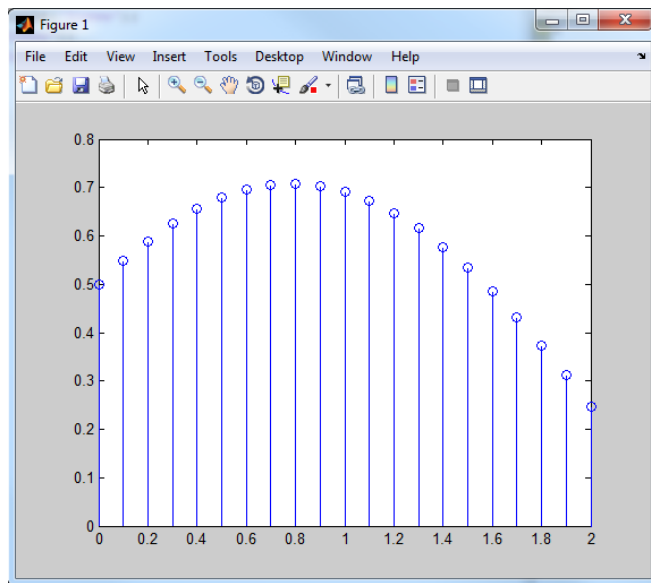



(c) l'instruction `>>subplot(a,b,c)` permet de dessiner plusieurs graphiques sur un même écran.
ex. `subplot(2,3,2)` dit que l'écran sera divisé en une grille 2x3 et que le graphique sera en deuxième position



(d) l'instruction `>>stem(x,y)` permet de dessiner uniquement les points (x,y) du graphe sans relier ces points

```
>> stem(t,y);
```



(e) l'instruction `>>close all` permet d'effacer toutes les fenêtres graphiques que vous avez créé.

(f) l'instruction `>>close all` permet d'effacer toutes les variables que vous avez créé.

Vous avez intérêt à commencer vos programmes par les instructions `clear all;close all`

3. Exercices :

- (a) Dessiner sur 4 graphiques, l'un en dessous de l'autre, la fonction $\sin(t)$
- au moyen de l'instruction `plot` avec t variant de 0 à 4 avec 21 points
 - au moyen de l'instruction `plot` avec t variant de 0 à 4 avec 9 points
 - au moyen de l'instruction `stem` avec t variant de 0 à 4 avec 21 puis 9 points
- Vérifier le nombre de points utilisés.

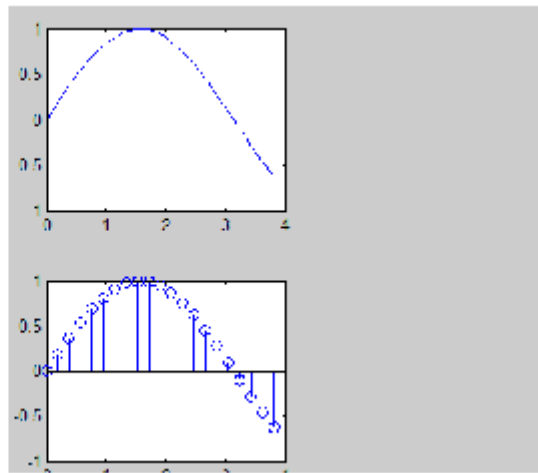
i. au moyen de l'instruction `plot` avec t variant de 0 à 4 avec 21 points

```
>> 4/21
```

```
ans =
```

```
0.1905
```

```
>> t=0:0.1905:4;  
>> y=sin(t);  
>> subplot(2,2,1);plot(t,y);  
>> subplot(2,2,3);stem(t,y);
```



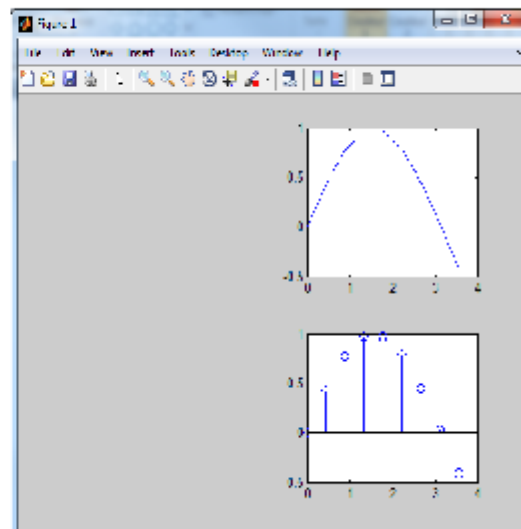
ii. au moyen de l'instruction `plot` avec t variant de 0 à 4 avec 9 points

```
>> 4/9
```

```
ans =
```

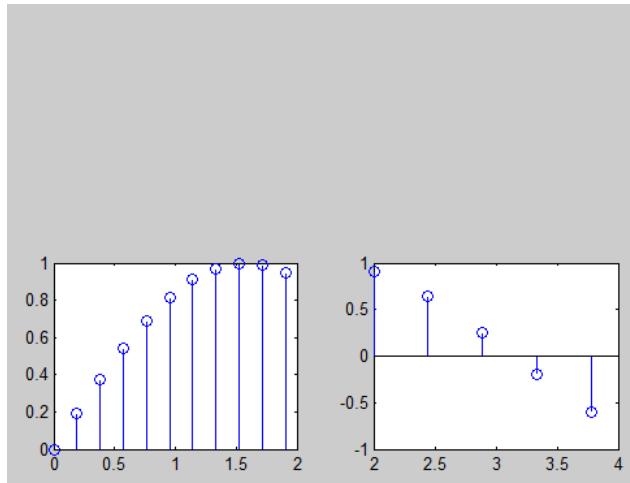
```
0.4444
```

```
>> t=0:0.445:4;  
>> y=sin(t);  
>> subplot(2,2,2);plot(t,y);  
>> subplot(2,2,4);stem(t,y);
```



iii. au moyen de l'instruction stem avec t variant de 0 à 4 avec 21 puis 9 points

```
clc;close all;clear all;  
  
t=0:0.1905:2;  
y=sin(t);  
subplot(2,2,3);stem(t,y);  
t=2:0.445:4;  
y=sin(t);  
subplot(2,2,4);stem(t,y);
```



Vérifier le nombre de points utilisés : 16 points

(b) Réaliser l'exercice précédent en exécutant un fichier d'extension .m
Il est de loin préférable d'écrire un programme plutôt qu'exécuter les instructions les unes à la suite des autres.
Ceci est plus facile et engendre moins d'erreurs.

Pas oublier

```
clear all ;  
close all ;
```

Ca me fait chier je le fais pas...

(c) Entrer une matrice A 10x10 composée de nombres aléatoires compris entre 0 et 1. Dessiner sur 3 graphiques disposés en colonnes :

- i. les éléments de la première ligne
- ii. les éléments de la première colonne
- iii. les éléments de la diagonale principale

Exécuter ce programme à partir d'un fichier matrice1.m

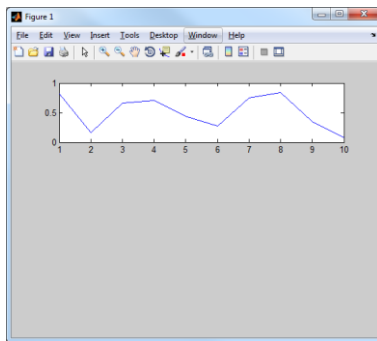
```
A = rand(10);
```

Rand(X) renvoie une matrice XsurX de nombre aléatoire...

Dessiner sur 3 graphiques disposés en colonnes :

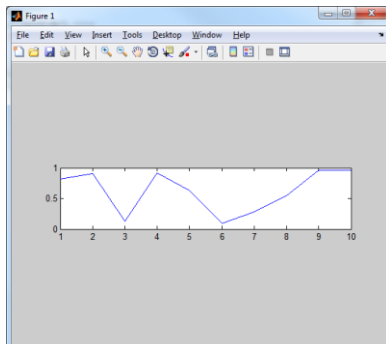
i. les éléments de la première ligne

```
subplot(3,1,1);  
plot(A(1,:));
```



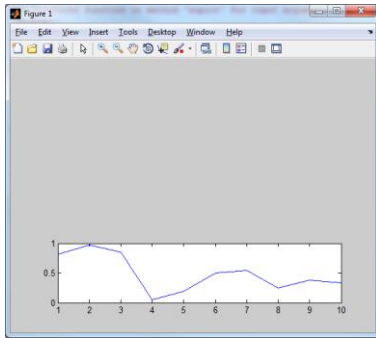
ii. les éléments de la première colonne

```
subplot(3,1,2);  
plot(A(:,1));
```



iii. les éléments de la diagonale principale

```
for i= 1:10  
    y(i) = A(i,i);  
end;  
  
subplot(3,1,3);  
plot(y);
```



Exécuter ce programme à partir d'un fichier `matrice1.m`

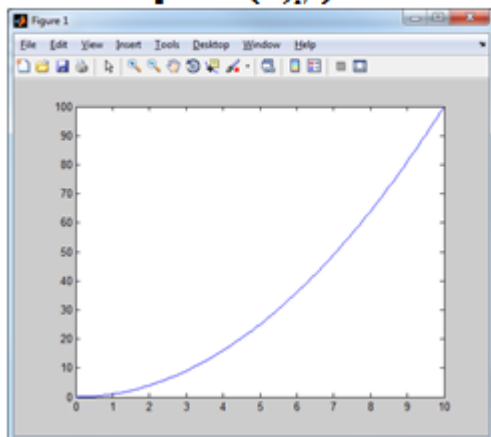
Ouais ouais c'est ça...

(d) Soit $t = 0 : 0.1 : 10$ et $y = t.^*t$
Quelle différence existe-il entre `plot(t,y)` et `plot(y)` ?

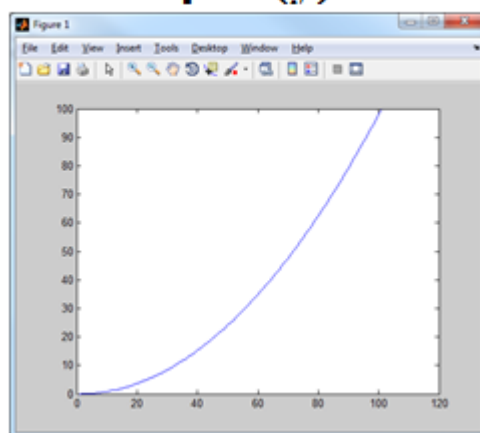
N.B. :

Par défaut *Matlab* considère que tout est une matrice. Donc l'opérateur d'exponentiation `^` est l'exponentiation matricielle (qui ne fonctionne que sur des matrices carrées). Pour préciser que l'on veut utiliser l'exponentiation sur les réels (et donc l'appliquer individuellement à chaque case du tableau) il faut utiliser la notation `.^`.

plot(t,y)



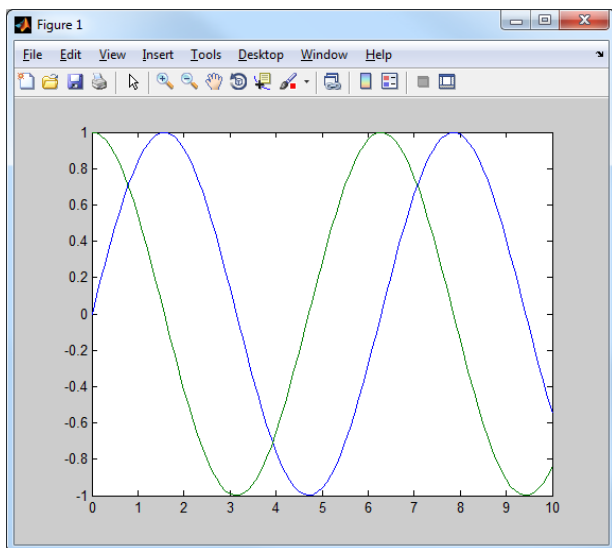
plot(y)



Dans `plot(y)` on ne lui précise pas la longueur du vecteur. 100 est donc le nombre de valeur car $[0.1*100 = 10]$.

(e) Exécuter l'instruction suivante :
`plot(t,sin(t),t,cos(t))`
Observer le résultat obtenu

`plot(t,sin(t),t,cos(t)) ;`



Sinus en bleu cosinus en vert... rien d'extra.

4. Optimisation d'un programme

Les calculs faisant intervenir des boucles (for, ...) sont très lents sous Matlab. Par contre Matlab est très rapide en ce qui concerne le calcul matriciel.

Comparons les deux programmes suivants qui permettent d'afficher le sinus des angles de 1 à 1000000 :

```
tic
    for i=1:1000000
        sol=sin(i)
    end
toc

tic
    i=1:1000000;
    sol=sin(i)
toc
```

Les instructions `tic` et `toc` permettent de connaître le temps d'exécution d'un programme.

Le second programme est équivalent au premier mais s'exécute plus rapidement.

Nous dirons que nous avons "vectorialisé" notre programme.

Putain d'exo de merde...à peine fini la première partie !