

# Initiation Matlab pour les stats

Le but n'est pas de faire de vous des pros  
de Matlab,  
mais de savoir l'utiliser rapidement



# Généralités



Lancer Matlab

Matlab = Matrix Laboratory avec ses Toolboxes  
**Besoin de la toolbox STATS**

The screenshot shows the MATLAB R2022b interface. The Command Window contains the following code and output:

```
>> % c'est la « command window »
>> 1 + 1
ans =
     2
>> x = ans + 3
x =
     5
>> y = 2*x^2 - 2;
>> Y = cos(2*pi*0.5) % différent de y
% Matlab sensible aux minuscules
% et Majuscules
```

The Workspace window shows the following variables:

Name	Value
ans	2
x	5
y	48

# Généralités

Type de variables : pas de déclaration nécessaire.... 😊 ou ☹️ ?

Il existe cinq grands types de variables sous Matlab : les entiers, les réels, les complexes, les chaînes de caractères et le type logique. Définissons une variable de chaque type :

```
>> a = 1.3; b = 3+i; c = 'bonjour';  
>> d1 = true(1==1); d2 = logical(1);  
>> e = int8(2);
```

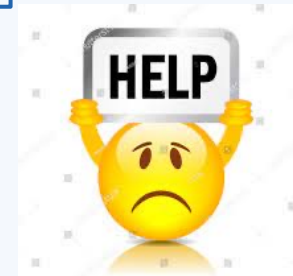
```
>> who ou whos % testez les deux commandes
```

## Définir un vecteur :

```
>> v = [1 2 3.3 4 5 6]; % vecteur ligne  
>> v % tapez enter pour le voir  
>> w = [1 ; 2 ; 3] % vecteur colonne  
>> length(w) % tapez enter  
>> size(w) % tapez enter
```

## Concaténer 2 vecteurs :

```
>> vw = [v w'] % un seul vecteur ligne  
>> vw_encolonne = [v' ; w] % ou alors vw' 😊
```



```
>> help ones  
>> help zeros
```

# Généralités

## Opérations en plus de + - / \*

Voici une liste (non exhaustive) des fonctions incorporées dans Matlab :

<code>exp(x)</code>	: exponentielle de x
<code>log(x)</code>	: logarithme népérien de x
<code>log10(x)</code>	: logarithme en base 10 de x
<code>x^n</code>	: x à la puissance n
<code>sqrt(x)</code>	: racine carrée de x
<code>abs(x)</code>	: valeur absolue de x
<code>sign(x)</code>	: 1 si $x > 0$ et 0 si $x \leq 0$
<code>sin(x)</code>	: sinus de x
<code>cos(x)</code>	: cosinus de x
<code>tan(x)</code>	: tangente de x
<code>asin(x)</code>	: sinus inverse de x (arcsin de x)
<code>sinh(x)</code>	: sinus hyperbolique de x
<code>asinh(x)</code>	: sinus hyperbolique inverse de x

On pourra également utiliser les fonctions d'arrondis :

<code>round(x)</code>	: entier le plus proche de x
<code>floor(x)</code>	: arrondi par défaut de x
<code>ceil(x)</code>	: arrondi par excès de x

Enfin lorsque l'on travaille avec des complexes on pourra utiliser :

<code>conj(z)</code>	: conjugué de z
<code>abs(z)</code>	: module de z
<code>angle(z)</code>	: argument de z
<code>real(z)</code>	: partie réelle de z
<code>imag(z)</code>	: partie imaginaire de z

# Généralités

## La force de Matlab : les opérations sur les vecteurs (et les matrices)

```
% créer un vecteur qui va de 0 à 1 sur 1024 points
>> vect1 = linspace(0,1,1024); % remarque : plusieurs façons de faire
>> vect2 = (1 : 3 : 1 + 3 * (1024-1)); % un autre vecteur de taille 1024
>> somme = vect1+vect2; % par ex
>> prodTermATerm = vect1 .* vect2; % le « . » avant l'opérateur = terme à terme
>> prodScal = vect1 * vect2' % ça fait une valeur = produit scalaire
>> M = vect1' * vect2; % ça fait une matrice 😊
```

`sum(x)` : somme des éléments du vecteur x

`prod(x)` : produit des éléments du vecteur x

`max(x)` : plus grand élément du vecteur x

`min(x)` : plus petit élément du vecteur x

`mean(x)` : moyenne des éléments du vecteur x

`sort(x)` : ordonne les éléments du vecteur x par ordre croissant

`fliplr(x)` : renverse l'ordre des éléments du vecteur x

# Généralités

## La force de Matlab : les opérations sur les vecteurs (et les matrices)

```
>> A = [ 8 1 6; 3 5 7; 4 9 2] % une matrice particulière
```

```
>> A = magic(3) % pour s'amuser
```

A =

```
8  1  6
3  5  7
4  9  2
```

```
>> A(3,2) % attention, indices commencent à 1 !!!
```

```
>> A'
```

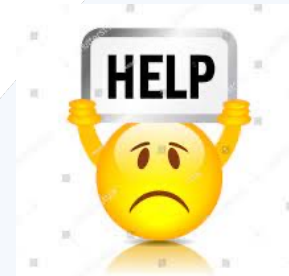
```
>> diag(A)
```

```
>> sum(diag(A))
```

```
>> sum(A,1)
```

```
>> sum(A,2)
```

```
>> inv(A)
```



```
>> help min
```

Les opérations classiques résumées ci-dessous peuvent être appliquées sur les variables scalaires, vectorielles ou matricielles.

### Opérations matricielles

+	addition
-	soustraction
*	multiplication
/	division
^	puissance

### Opérations élément par élément

.*	multiplication élément par élément
./	division élément par élément
.^	puissance élément par élément

# Généralités

On crée un « script » (programme) :

New script

Save as Initiation.m

On démarre un programme : en commentaire les infos

```
% Initiation matlab – février 2024
```

```
% auteur : Prénom NOM
```

```
% exemple : créer un vecteur comme une fonction type cosinus et la représenter
```

```
N = input(' Nombre de points du signal : '); % pour poser une question
```

```
t = linspace(0,2*pi, N); % vecteur en abscisse
```

```
b=-0.4; a = 0.4/2/pi; % autres paramètres
```

```
for k = 1 : N
```

```
    x(k) = cos(t(k));
```

```
    y(k) = a*t(k) + b; % autre fonction
```

```
end
```

```
% mieux en une ligne !!!
```

```
x = cos(t); y = a*t + b; % on peut souvent éviter for/end
```

```
for indice = valeur initiale:pas:valeur finale
```

```
    ...suite d'instructions...
```

```
end
```

# Généralités

## Visualisation des courbes

% de nombreuses possibilités et options !!!

% a minima

figure;

**plot(t,x);** hold on; plot(t,y);

**xlabel('phase en radians');**

**ylabel('amplitude');**

**title('Evolution de deux fonctions');**

**legend('cosinus','linéaire');**

v=axis; v(2) = pi; axis(v); % utile pour bien dimensionner la figure

% autre possibilité utile

**figure('Name','Visualisation en sous-figures');** % pour s'y retrouver dans l'affichage

**subplot(2,3,1);** plot(t,x); title('sous-figure 1');

subplot(2,3,2); plot(t,y); title('sous-figure 2');

subplot(2,3,3); **plot(t,x,'r');** title('sous-figure 3');

subplot(2,3,4); plot(t,y,'g\*'); title('sous-figure 4');

subplot(2,3,5); **plot(t,x,'s--');** title('sous-figure 5');

subplot(2,3,6); plot(t,y); title('sous-figure 6'); **set(gca, 'fontsize', 15)**

Voir ensuite toutes les options en faisant :

>> help plot

Et cliquer sur [Documentation for plot](#)



# Généralités

```
if expression à tester
    ...suite d'instructions à exécuter si l'expression est vraie...
else
    ...suite d'instructions à exécuter si l'expression est fausse...
end
```

```
Test = zeros(size(x));
```

```
for k=1:N
```

```
    if x(k) > y(k)
        test(k) = 1;
    else
        test(k) = -1;
    end
```

```
end
```

```
% figure (1); % pour y revenir
```

```
figure; % une nouvelle
```

```
plot(t,x,'r',t,y,'b',t,test,'c');
```

```
% plus fort !!!
```

```
test = (x>y); % variable logique (mais qu'on peut visualiser comme les autres)
```

```
Test = 2*test -1; % devient réelle 😊
```

```
% ou alors
```

```
[test indices] = find(x>y); Test = length(test);
```

```
Plot(t(indices), test, '*'); % ne pas tracer la ligne entre les moindres
```

```
Disp(['Le résultat est ' num2str(Test/N*100) '%']); % pour montrer « disp »
```

## Opérateurs relationnels et logiques de Matlab

== égalité

~= inégalité

< strictement plus petit que

> strictement plus grand que

<= plus petit ou égal

>= plus grand ou égal

| ou logique

& et logique

# Généralités

## Matlab et les fichiers de données

Fichiers au format Matlab : **.mat** (option **'-ascii'**)

save et load

Mais Matlab sait lire et écrire beaucoup de formats de fichiers

### Categories

#### Text Files

Delimited and formatted text files

#### Spreadsheets

Microsoft® Excel® spreadsheets

#### Images

imread, imwrite, ...

JPEG, TIFF, PNG, and other formats

#### Scientific Data

netCDF, HDF, FITS, and CDF formats

#### Audio and Video

audioread, audiowrite, ...

Read and write video and audio files; Record and play audio

#### Structured Data and XML Documents

Work with structured data and Extensible Markup Language documents

#### JSON Format

JavaScript® Object Notation format

Et aussi dans sa forme la plus « brute »  
fopen, fread, fclose,...

# Généralités

## Matlab et les fonctions (sous-programmes)

### Exemple

```
%%% ceci est le main
clear all; close all;

nech = 1000; amp = 1; f0 = 0.21;
x = genersignal(nech, amp, f0); % c'est la fonction
plot(x);
```

Et dans un fichier séparé appelé **genersignal.m**

```
function signal = genersignal(n, a, f0)

signal = a*cos(2*pi*f0*(0:n-1));
signal = signal.^2; % pour corser
signal = signal + a^3;
```

# Généralités

En conclusion, pour savoir comment faire quelque chose sous matlab :

>> help mean

>> lookfor mean



matlab calcul moyenne



# Faire des stats avec Matlab



```
>> ver  
>> help stats
```

## Simulation de variables aléatoires : algorithme de Box-Müller

Soient  $U_1$  et  $U_2$  deux v.a.r. indépendantes de loi uniforme  $\mathcal{U}([0, 1])$ .

Posons

$$\begin{cases} X_1 = \sqrt{-2 \ln(U_1)} \cos(2\pi U_2), \\ X_2 = \sqrt{-2 \ln(U_1)} \sin(2\pi U_2). \end{cases}$$

Alors  $X_1$  et  $X_2$  sont deux v.a.r. indépendantes de loi normale centrée réduite  $\mathcal{N}(0, 1)$ .

C'est à  
**VOUS!**



Simuler des v.a.  $\mathcal{N}(0,1)$  à l'aide de l'algorithme ci-dessus.  
Pour vérifier, utiliser :

histogram : on peut faire hold on 😊 et superposer avec la densité de proba d'une  $\mathcal{N}(0,1)$ , avec l'option « Normalization » d'histogram

Ou mieux,

histfit

(faire help histogram et histfit pour les paramètres)

Bien sûr, générateur  $\mathcal{N}(m,s)$  existe sous Matlab : voir `randn`

# Faire des stats avec Matlab

C'est à VOUS!



<https://voiceage.com/Audio-Samples-AMR-WB.html> % télécharger le fichier Sports original

[https://voiceage.com/wbsamples/in\\_mono/Sports.wav](https://voiceage.com/wbsamples/in_mono/Sports.wav)

```
[ x fs] = audioread("Sports.wav"); % x signal, fs fréquence d'échantillonnage
```

```
sound(x,fs); % pour écouter ☺
```

```
% faire l'histogramme : régler le nombre de classes pour montrer l'effet
```

% remarque 1 : Règle empirique nombre de classes (bins) = round(sqrt(Nech))

% fitter avec la loi qui paraît la bonne (faire help histfit pour voir les lois possibles)

% remarque 2 pour aider : si X suit une loi double exponentielle (pas prévu dans histfit), abs(X) suit une loi exponentielle (prévu dans histfit ☺)

% remarque 3 : les traiteurs de parole prennent pour hypothèse que la parole suit une loi de type exponentielle

## Estimation de paramètres de variables aléatoires :

	valeur théorique	valeur empirique
Moyenne	$\mu$	$m_e = \frac{1}{n} \sum_{i=1}^n x_i$
Variance	$\sigma^2$	$\sigma_e^2 = \frac{1}{n} \sum_{i=1}^n (x_i - m_e)^2$

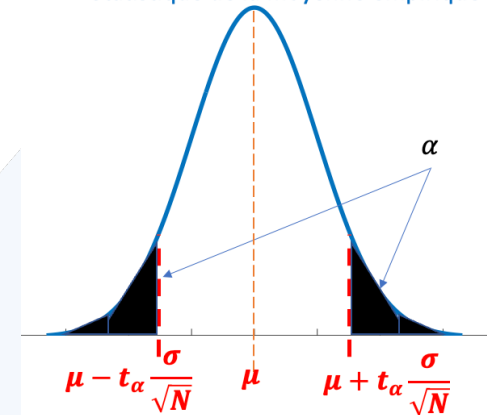
# Faire des stats avec Matlab

## Intervalles de confiance

À tout  $n$ -échantillon  $(x_1, \dots, x_n)$  de moyenne empirique  $m_e$  on associe l'intervalle de confiance au niveau de confiance  $1 - \alpha$  de la moyenne  $\mu$  suivant :

$$I_n^\alpha := \left[ m_e - t_\alpha \frac{\sigma}{\sqrt{n}}, m_e + t_\alpha \frac{\sigma}{\sqrt{n}} \right],$$

Statistique de la moyenne empirique



C'est à VOUS!



Comparaison du pourcentage d'intervalles de confiance contenant effectivement  $\mu$  avec le seuil de confiance.]

1. Simuler  $m$  échantillons de tailles  $n$  d'une variable aléatoire suivant une loi de votre choix.
2. Pour un niveau de confiance  $1 - \alpha$  fixé, déterminer  $m$  intervalles de confiance de  $\mu$  à partir des  $m$  échantillons.
3. Calculer le pourcentage d'intervalles contenant réellement  $\mu$ .