Outils Mathématiques pour l'Ingénieur
...
Traitement du Signal
...
TD d'initiation à Matlab

Benoît Marx

Centre de Recherche en Automatique de Nancy (CRAN)
Ecole Nationale Supérieure de Géologie (ENSG)

Introduction à MATLAB

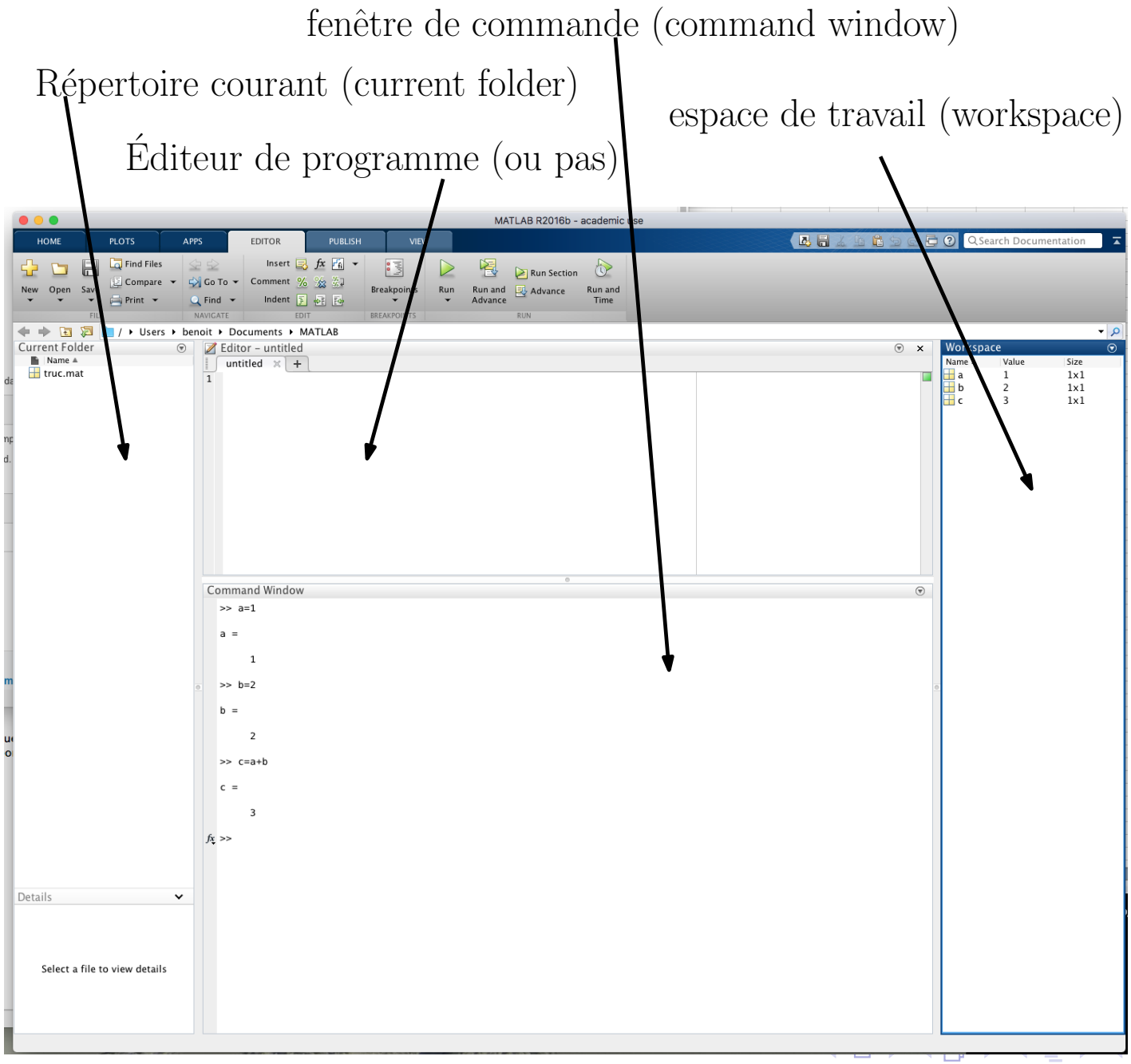
- **Où**
  - UNIVERSITÉ DE LORRAINE | ENT > VIRT'UL > MATLAB R2020a Sciences
  - autre possibilité : une version "Matlab Student" sur votre machine, en se créant un compte avec votre e-mail de l'UL

- **Quoi ?**
  - Logiciel de calcul **numérique** (et non symbolique)

- **Pourquoi ?**
  - langage de programmation simple (dérivé du C)
  - toolboxes variées (TdS, éléments finis, aéronautique, finance, etc.)
  - répandu dans les universités / industries
  - bien interfacé et maintenu

- **Pourquoi pas ?**
  - cher
  - alternatives libres : Octave et Scilab

Fenêtre d'accueil de MATLAB



Dans la fenêtre de commande (command window)

- Commandes lignes à lignes utilisant des fonctions prédéfinies, comme une calculatrice

```
>> a=1
        a = 1
>> b=2
        b = 2
>> c=a+b
        c = 3
>> cos(a)
        ans = 0.5403
```

- Les variables définies (ici `a`,`b`,`c` et `ans`) apparaissent dans l'espace de travail (workspace)

Calcul numérique / Calcul formel

- Définir une variable est simplement l'affectation d'une valeur numérique dans une variable

```
>> a=1
        a = 1
>> c=2*a
        c = 2
>> a=2
        a = 2
>> c
        c = 2
```

Calcul numérique / Calcul formel

- Définir une variable est simplement l'affectation d'une valeur numérique dans une variable

```
>> a=1
        a = 1
>> c=2*a
        c = 2
>> a=2
        a = 2
>> c
        c = 2
```

- Exemples

```
>> f=cos(t)
```
affecte la valeur `cos(t)` à `f` (si `t` est défini)

```
>> a=a+2
```
augmente la valeur de `a` de 2, c'est une affectation et non une égalité

Appel à une fonction et aide

- Syntaxe d'appel : $\underbrace{[s1, s2, s3, ...]}_{arguments\ de\ sortie} = nomfonction\ \underbrace{(e1, e2, e3, ...)}_{arguments\ d'entrée}$

  >> `cos(3)` ou `max(1,5)`
  affiche un résultat (numérique ou graphique)
  >> `c3=cos(3)`

  ```
  c3 = -0.9900
  ```

  affecte le résultat dans la variable c3

Appel à une fonction et aide

- Syntaxe d'appel : $\underbrace{[s1, s2, s3, ...]}_{\textit{arguments de sortie}} = \textit{nomfonction} \underbrace{(e1, e2, e3, ...)}_{\textit{arguments d'entrée}}$

  `>>` `cos(3)` ou `max(1,5)`
  affiche un résultat (numérique ou graphique)
  `>>` `c3=cos(3)`

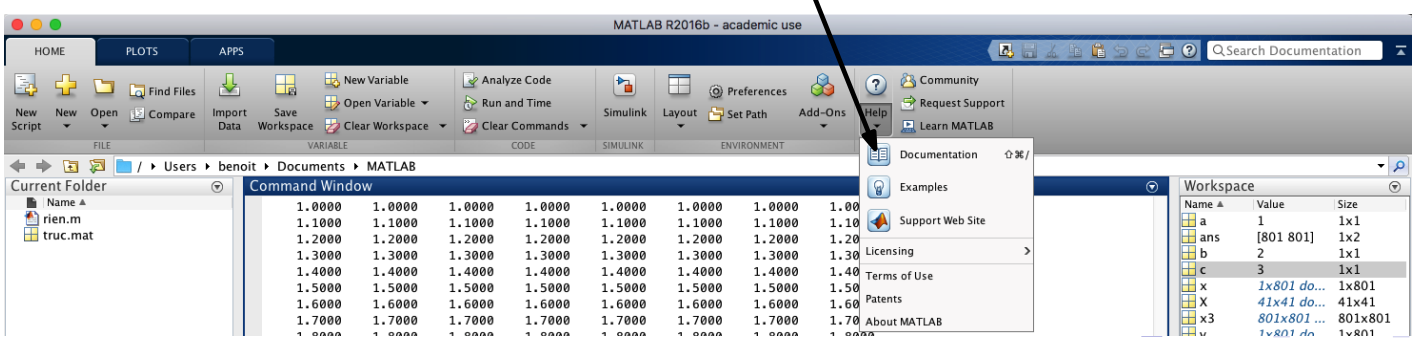
  ```
  c3 = -0.9900
  ```

  affecte le résultat dans la variable `c3`

- Consulter l'aide d'une fonction pour connaitre sa syntaxe

  ```
  >> help nomfonction
  >> doc nomfonction
  ```

  ou ouvrir la fenêtre d'aide de MATLAB là



Enregistrement de données

- Les données peuvent être sauvegardées sous forme de fichiers `nom.mat` avec la fonction `save`

  `>> save('truc.mat')`
  sauvegarde tout l'espace de travail dans le fichier `truc.mat`

  `>> save('truc2.mat','a','b')`
  sauvegarde les variables `a` et `b` dans le fichier `truc2.mat`

Enregistrement de données

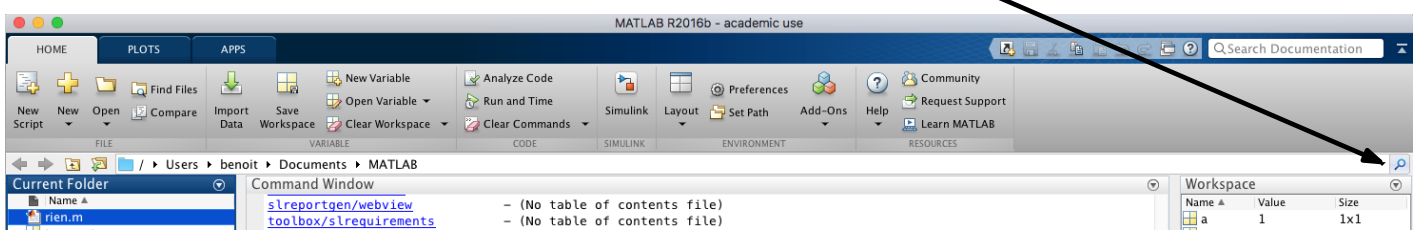
- Les données peuvent être sauvegardées sous forme de fichiers `nom.mat` avec la fonction `save`

  `>> save('truc.mat')`
  sauvegarde tout l'espace de travail dans le fichier `truc.mat`
  `>> save('truc2.mat','a','b')`
  sauvegarde les variables `a` et `b` dans le fichier `truc2.mat`

- Les fichiers créés sont enregistrés dans le `répertoire courant (current folder)`. Pour choisir ce répertoire courant :

  `>> cd('U:\ ... ')`

  ou choisir le répertoire courant en cliquant là



Enregistrement de données via Virt'UL (so 2020!)

- Avec Virt'UL, les données/fichiers sont enregistrées sous :

  `UL-ESPACE PERSONNEL PEDA (O:) > ...`

- Pour copier/coller des fichiers depuis/vers cet espace :
  - UNIVERSITÉ DE LORRAINE | ENT > VIRT'UL > Explorateur Windows Bureautique
  - UNIVERSITÉ DE LORRAINE | ENT > Espace de stockage

Chargement de données en mémoire

- Les données peuvent être chargées en mémoire à partir d'un fichiers `nom.mat` avec la fonction `load`

  `>> load('truc.mat')`
  charge toutes les variables contenues dans le fichier `truc.mat`

  `>> load('truc2.mat','a')`
  charge la variable a contenue dans le fichier `truc2.mat`

Chargement de données en mémoire

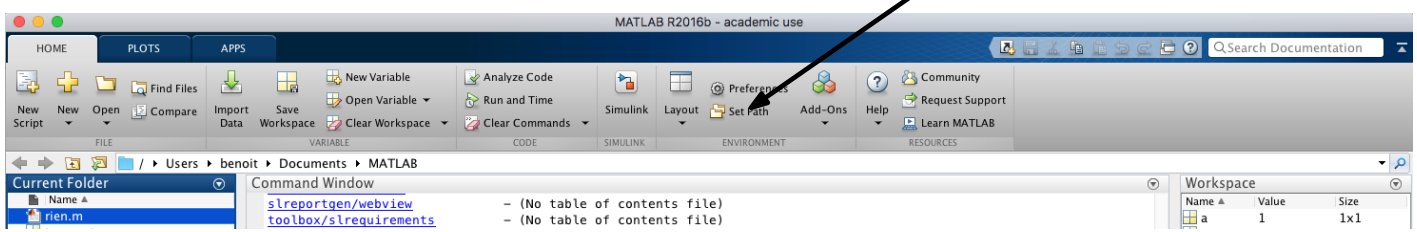
- Les données peuvent être chargées en mémoire à partir d'un fichiers `nom.mat` avec la fonction `load`

  `>> load('truc.mat')`
  charge toutes les variables contenues dans le fichier `truc.mat`
  `>> load('truc2.mat','a')`
  charge la variable a contenue dans le fichier `truc2.mat`

- Les fichiers créés sont enregistrés dans le `répertoire courant (current folder)`. Pour choisir ce répertoire courant :

  `>> addpath('U:\ ... ')`

  ou inclure le répertoire dans le `path` en cliquant là



- Créer des données (variables *a*, *b*, ... par exemple).
- Enregistrer ces données (commande `save`) dans un fichier de données (extension `.mat`).
- Effacer le workspace (commande `clear`).
- Charger les données en mémoire (commande `load`).
- Répéter l'opération en changeant le répertoire courant (commande `cd` ou sur l'interface) et le chemin d'accès (commande `addpath` ou sur l'interface).

Pour créer une matrice $M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

- Définir tous ses coefficients

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

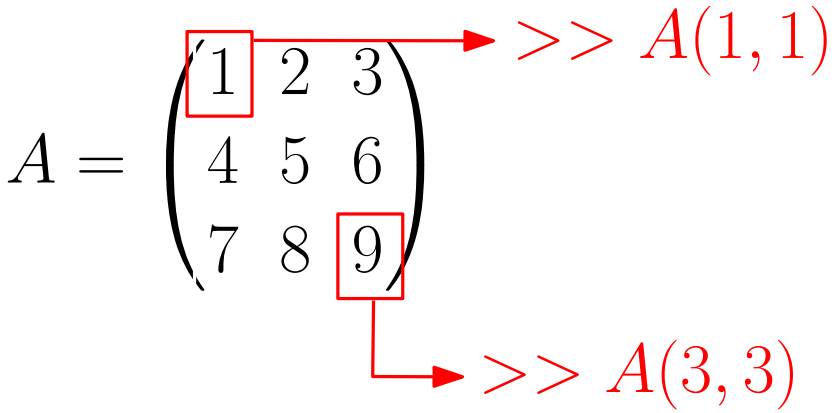
- Définir ses lignes et les concaténer verticalement

```
>> L1=[1 2 3]
>> L2=[4 5 6]
>> L3=[7 8 9]
>> M=[L1; L2; L3]
```

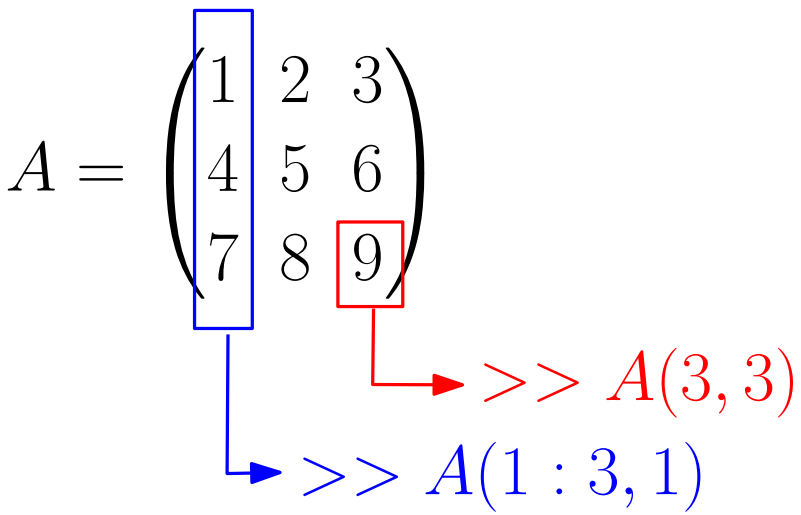
- Définir ses colonnes et les concaténer horizontalement

```
>> C1=[1; 4; 7]
>> C2=[2; 5; 8]
>> C3=[3; 6; 9]
>> M=[C1 C2 C3]
```

- La syntaxe est `A(n° ligne,n° colonne)`
- La numérotation commence à 1
- Exemples

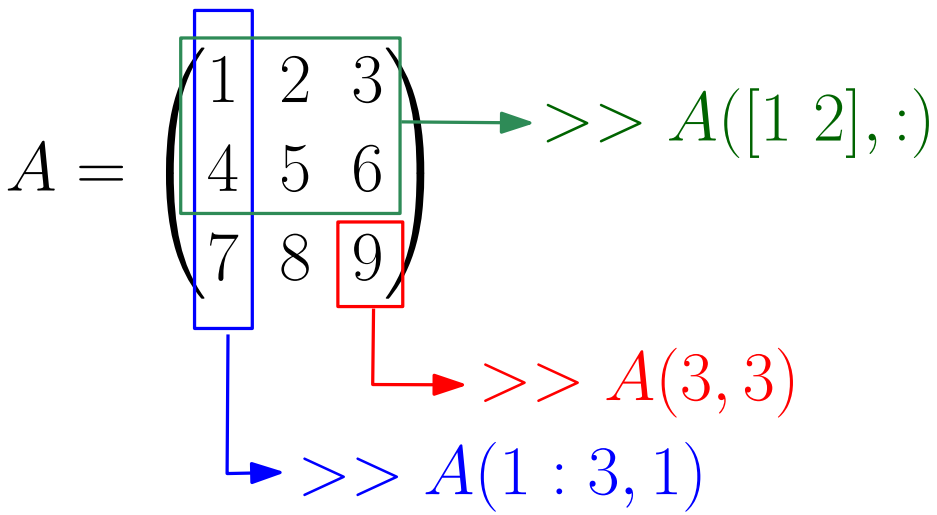


- La syntaxe est `A(n° ligne,n° colonne)`
- La numérotation commence à 1
- Exemples



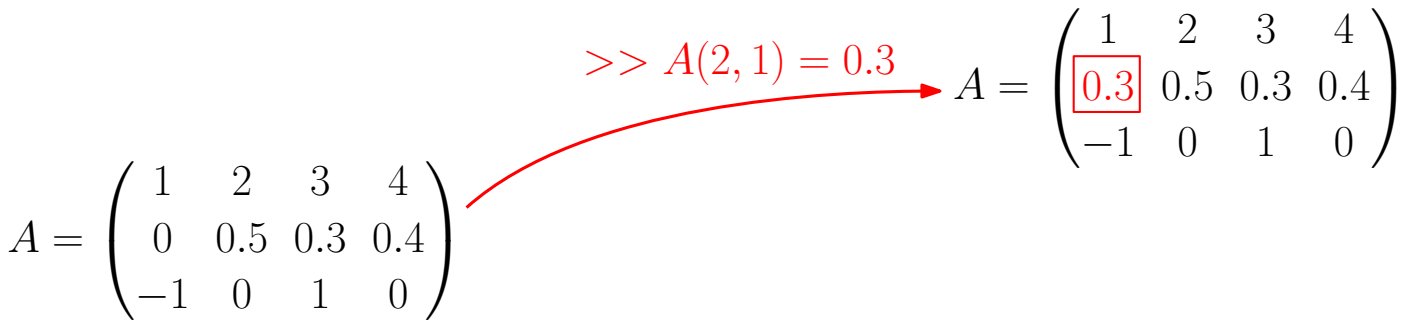
`>> A(1:3,1)` : de la 1ère à la 3ème lignes

- La syntaxe est `A(n° ligne,n° colonne)`
- La numérotation commence à 1
- Exemples

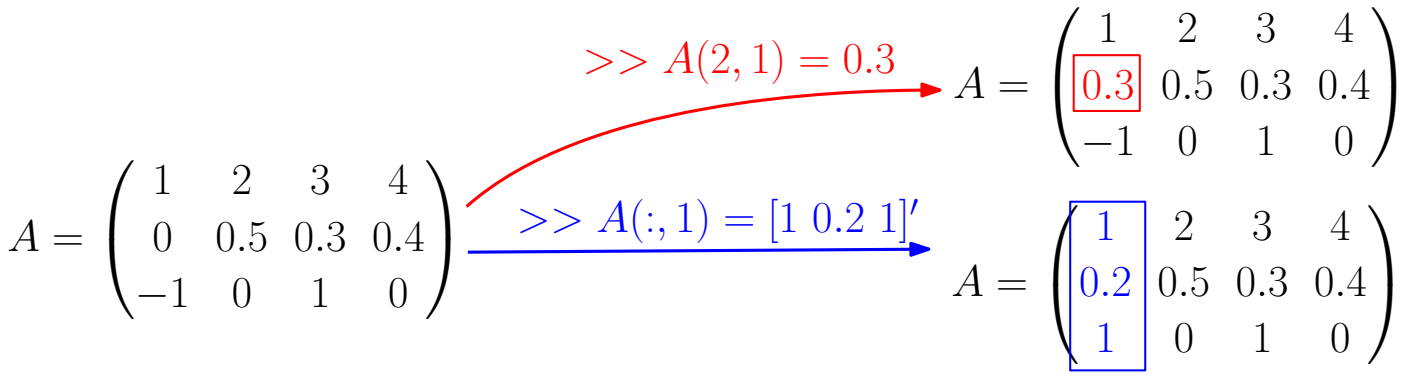


>> `A(1:3,1)` : de la 1ère à la 3ème lignes
>> `A([2 3],1)` : la 2ème et la 3ème lignes
>> `A(:,1)` : toutes les lignes

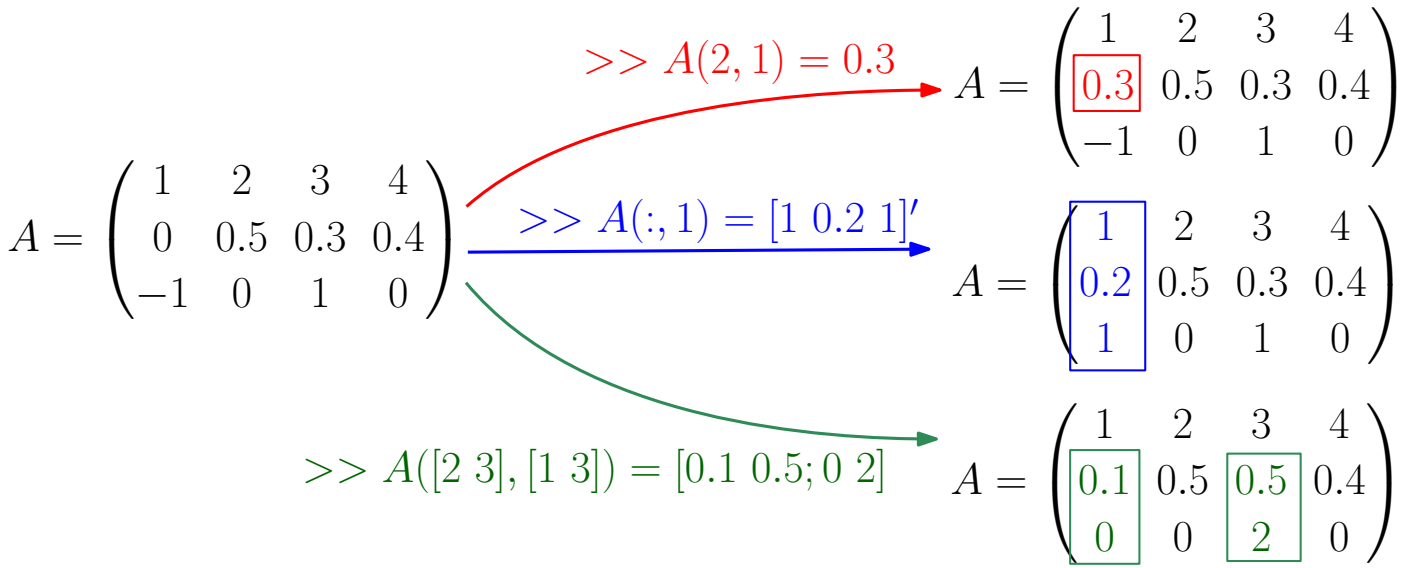
- Syntaxe identique en lecture et en écriture :
  (sous réserve que les dimensions soient compatibles)



- Syntaxe identique en lecture et en écriture :
  (sous réserve que les dimensions soient compatibles)



- Syntaxe identique en lecture et en écriture :
  (sous réserve que les dimensions soient compatibles)



- Vecteur de valeurs linéairement espacées (base de temps) :
  >> `t=[0:0.01:20]`
  crée le vecteur $t = [0\ 0.01\ 0.02\ \ldots\ 19.98\ 19.99\ 20]$

  >> `t=[0:20]`
  crée le vecteur $t = [0\ 1\ 2\ \ldots\ 19\ 20]$

  >> `t=[0:-1:-10]`
  crée le vecteur $t = [0\ -1\ -2\ \ldots\ -9\ -10]$

- Vecteur de valeurs linéairement espacées (base de temps) :
  >> `t=[0:0.01:20]`
  crée le vecteur $t = [0\ 0.01\ 0.02\ \ldots\ 19.98\ 19.99\ 20]$

  >> `t=[0:20]`
  crée le vecteur $t = [0\ 1\ 2\ \ldots\ 19\ 20]$

  >> `t=[0:-1:-10]`
  crée le vecteur $t = [0\ -1\ -2\ \ldots\ -9\ -10]$

- Vecteur de valeurs logarithmiquement espacées (base de ω dans un diagramme de Bode) :
  >> `logspace(a,b,n)`
  crée un vecteur de n valeurs logarithmiquement espacées de 10^a à 10^b.

>> `eye(n)`
crée la matrice I_n

>> `zeros(n,m)`
crée la matrice $\begin{pmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{pmatrix}_{n \times m}$

>> `ones(n,m)`
crée la matrice $\begin{pmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{pmatrix}_{n \times m}$

>> `rand(n,m)`
crée une matrice de dimension $n \times m$ dont chaque composante est le tirage d'une loi uniforme sur [0 1]

>> `randn(n,m)`
crée une matrice de dimension $n \times m$ dont chaque composante est le tirage d'une loi normale $\mathcal{N}(0,1)$

- **Transposition** ($M \to M^T$)
  `>> M'`
- **Addition, soustraction, multiplication**
  `>> A+B`
  `>> A-B`
  `>> A*B`
- **Puissance, inversion, exponentielle**
  `>> A^2`
  `>> inv(A)` ou $\texttt{A}^{-1}$
  `>> exp(M)`
- **Opération terme à terme**
  `>> C=A.*B`
  Crée la matrice C telle que : $c_{i,j} = a_{i,j} b_{i,j}$

- **Obtenir les dimensions d'une matrice / un vecteur**

```
>> A=[1 2 3;4 5 6];
>> d=size(A)
        d = [2 3]
>> nl=size(A,1)
        nl = 2
>> nc=size(A,2)
        nc = 3
>> n=length([1 2 3 4 5])
        n = 5
```

- **Valeurs propres / vecteurs propres**

```
>> eig(M)
```

voir l'aide (`help eig`) pour avoir les vecteurs propres...

- **Rang d'une matrice**

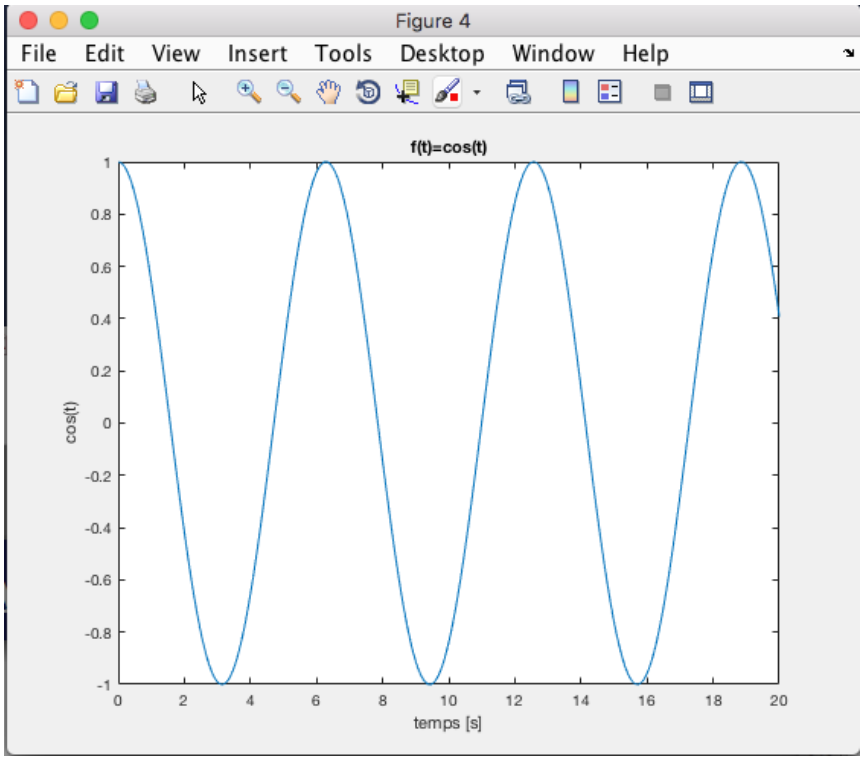
```
>> rank(M)
```

- **Diagonalisation d'une matrice**
  Définir la matrice
  $$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$
  et utiliser la fonction `eig` pour la diagonaliser.

- **Résolution d'un système linéaire**
  Résoudre le système linéaire suivant :
  $$\begin{cases} 1 = x + y + z \\ 2 = x - 2y + 3z \\ 3 = 2x + y - z \end{cases}$$

```
>> t=[0:0.01:20];
>> x=sin(t);
>> figure
```
ouvre une fenêtre graphique

```
>> plot(t,x)
```
tracé des points de coordonnées $(t(k), x(k))$



Graphiques > En coordonnées cartésiennes

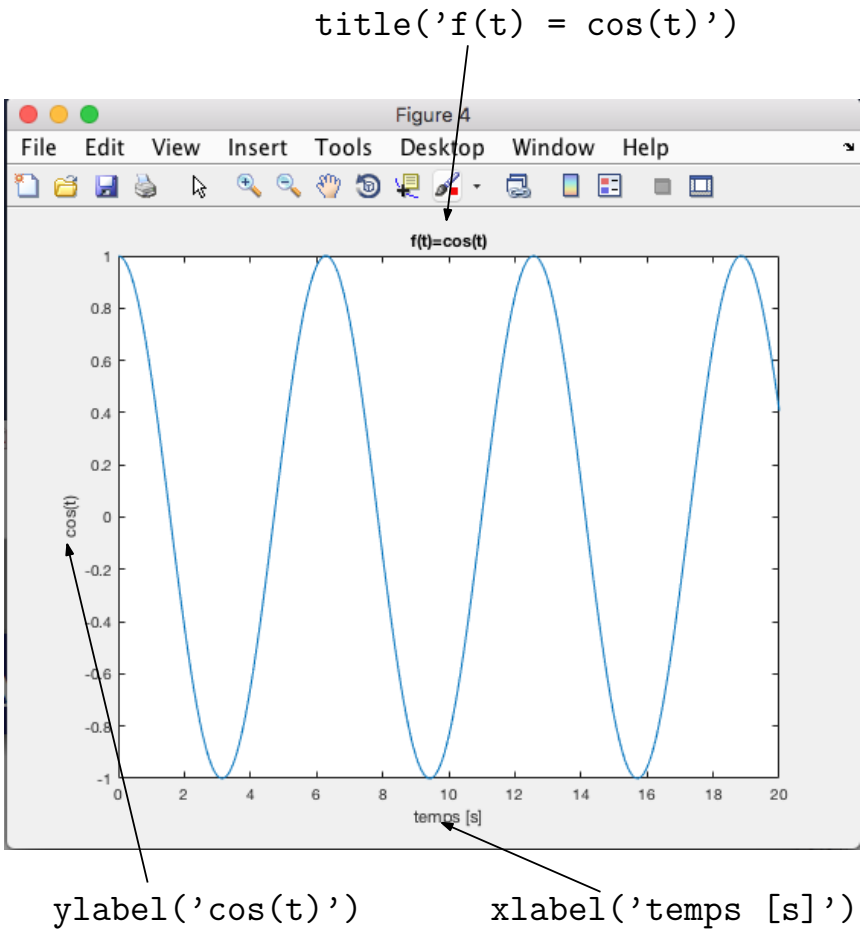
```
>> t=[0:0.01:20];
>> x=sin(t);
>> figure
```
ouvre une fenêtre graphique

```
>> plot(t,x)
```
tracé des points de coordonnées $(t(k), x(k))$

```
>> title('f(t)=cos(t)')
```
ajoute un titre à la figure

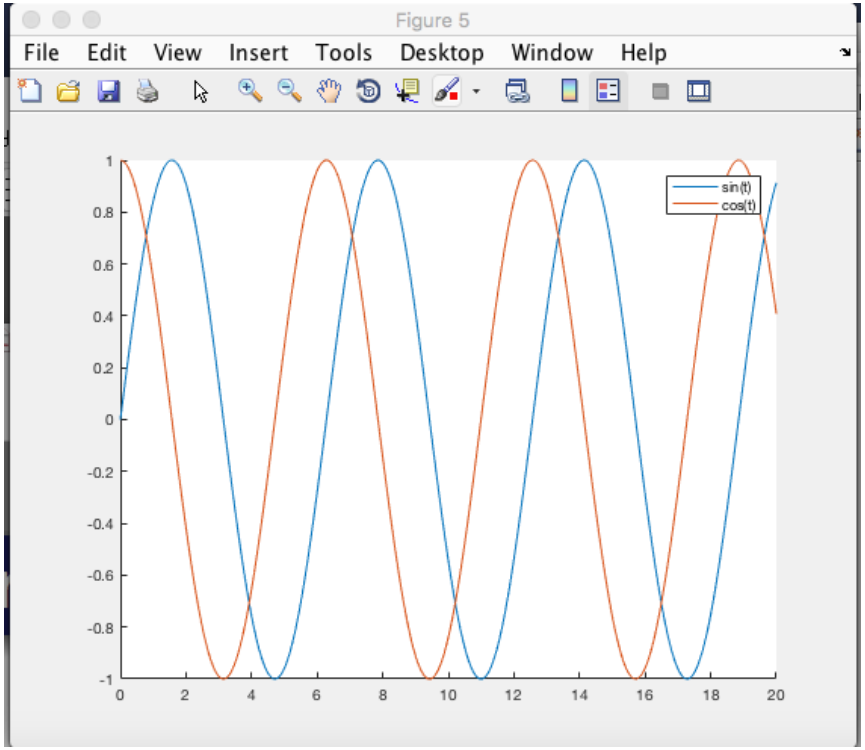
```
>> xlabel('temps [s]')
```
légende l'axe des abscisses

```
>> ylabel('cos(t)')
```
légende l'axe des ordonnées



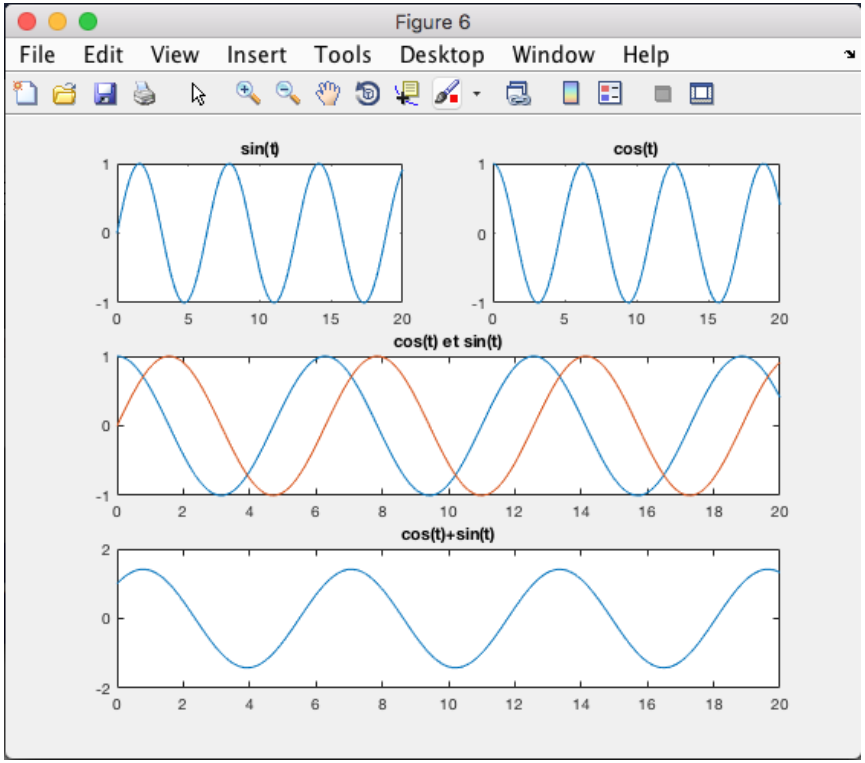
- **Superposition dans le même système d'axes**

```
>> t=[0:0.01:20];
>> x=sin(t);
>> y=cos(t);
>> figure
>> hold on
>> plot(t,x)
>> plot(t,y)
>> legend('sin(t)','cos(t)')
```

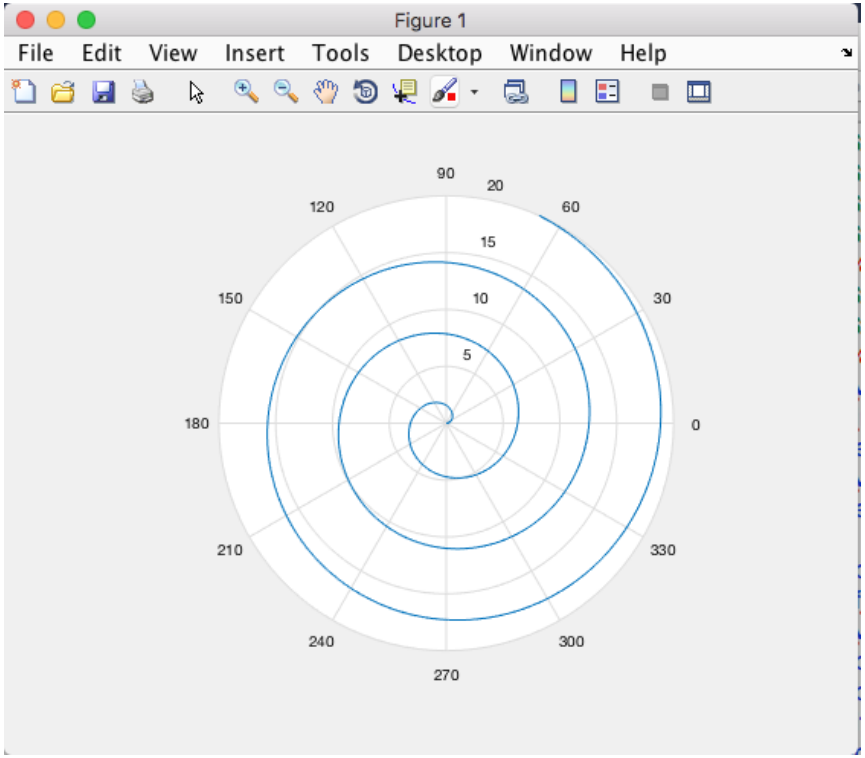


- **Plusieurs systèmes d'axes**

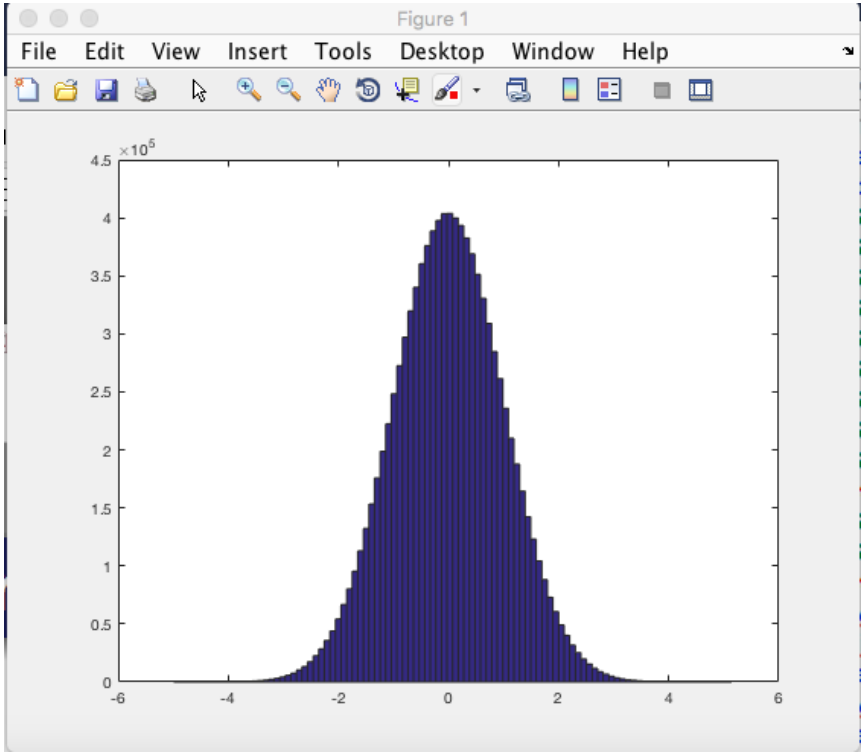
```
>> figure
>> subplot(3,2,1)
>> plot(t,x)
>> title('sin(t)')
>> subplot(3,2,2)
>> plot(t,y)
>> title('cos(t)')
>> subplot(3,2,3:4)
>> plot(t,y,t,x)
>> subplot(3,2,5:6)
>> plot(t,x+y)
```



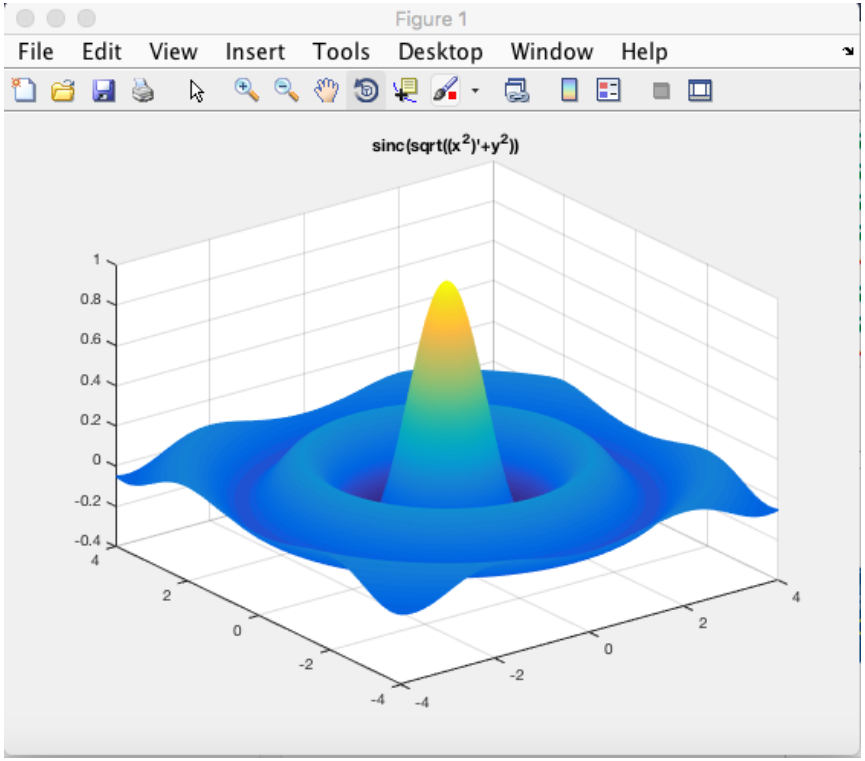
- **Echelle logarithmique**
  `semilogx` : axe des abscisses
  `semilogy` : axe des ordonnée
  `loglog` : les deux axes
- **Coordonnées polaires**
  `polar(theta,rho)`



- **Echelle logarithmique**
  `semilogx` : axe des abscisses
  `semilogy` : axe des ordonnée
  `loglog` : les deux axes
- **Coordonnées polaires**
  `polar(theta,rho)`
- **Histogramme**
  `>> hist(x,nc);`



- **Echelle logarithmique**
  `semilogx` : axe des abscisses
  `semilogy` : axe des ordonnée
  `loglog` : les deux axes
- **Coordonnées polaires**
  `polar(theta,rho)`
- **Histogramme**
  `>> hist(x,nc);`
- **Tracé en 3D**
  `>> mesh(x,y,Z)`
  $x : (1 \times n)$, $y : (1 \times m)$, $Z : (m \times n)$



- **Diagramme de Bode**
  Tracer le diagramme de Bode de $H(p) = \frac{10}{5p+1}$ dans une figure.
  Rappels :
  $$\begin{cases} G_{db}(\omega) = 20log\left(\left|\frac{K}{\sqrt{1+T^2\omega^2}}\right|\right) \\ \varphi(\omega) = -Atan(T\omega) \end{cases}$$
  Indication : utiliser `subplot, semilogx, log10, sqrt, ...`

- **Fonction de répartition de la loi normale**
  Tracer l'histogramme de la fonction de répartition de la loi normale centrée réduite $\mathcal{N}(0,1)$ et caler sa courbe théorique.
  Indication : utiliser `randn, hist, exp, ...`

- **Impact d'une goute sur une surface**
  On modélise l'impact d'une goute par $z(x,y) = sinc(\sqrt{x^2+y^2})$.
  Indication : utiliser `mesh, sinc, sqrt, ...`

Script

– ensemble de lignes de commande
– tapées dans l'éditeur (>> `edit`)
– enregistré dans un fichier `.m`
– workspace partagé entre éditeur et command window

Exemple

essai.m

```
s=a+b;
p=a*b;
q=a/b;
```

– appelé en tapant le nom du fichier
>> `essai`

\+ débuggage

Script

– ensemble de lignes de commande
– tapées dans l'éditeur (>> `edit`)
– enregistré dans un fichier `.m`
– workspace partagé entre éditeur et command window

Exemple

essai.m

```
s=a+b;
p=a*b;
q=a/b;
```

– appelé en tapant le nom du fichier
>> `essai`

\+ débuggage

Fonction

idem sauf :
– syntaxe de la première ligne
– 2 workspaces distincts
– nom de fichier = nom de fonction

Exemple

essaif.m

```
function [s,p]=essaif(a,b)
s=a+b;
p=a*b;
q=a/b;
```

– syntaxe d'appel à une fonction
>> `[som,prod]=essaif(2,3);`

\+ pas d'écrasement

- **Exemple : programmation de la fonction signe**

$$signe(x) = \begin{cases} 1 & si\ x > 0 \\ 0 & si\ x = 0 \\ -1 & si\ x < 0 \end{cases}$$

```
if x>0
    signe=1;
end
if x==0
    signe=0;
end
if x<0
    signe=-1;
end
```

- **Exemple : programmation de la fonction signe**

$$signe(x) = \begin{cases} 1 & si\ x > 0 \\ 0 & si\ x = 0 \\ -1 & si\ x < 0 \end{cases}$$

```
if x>0
    signe=1;
end
if x==0
    signe=0;
end
if x<0
    signe=-1;
end
```

```
if x>0
    signe=1;
else
    if x==0
        signe=0;
    else
        signe=-1;
    end
end
```

- **Exemple : programmation de la fonction signe**

$$signe(x) = \begin{cases} 1 & si\ x > 0 \\ 0 & si\ x = 0 \\ -1 & si\ x < 0 \end{cases}$$

```
if x>0
    signe=1;
end
if x==0
    signe=0;
end
if x<0
    signe=-1;
end
```

```
if x>0
    signe=1;
else
    if x==0
        signe=0;
    else
        signe=-1;
    end
end
```

```
if x>0
    signe=1;
elseif x==0
    signe=0;
else
    signe=-1;
end
```

- **Exemple : programmation de la somme** $1/n^2$

$$S = \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

- avec une boucle FOR

```
S=0
for n=1:1e6
    S=S+1/n^2;
end
```

- **Exemple : programmation de la somme** $1/n^2$

$$S = \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

- avec une boucle FOR (à gauche) ou une boucle WHILE (à droite)

```
S=0
for n=1:1e6
    S=S+1/n^2;
end
```

```
S=0
n=1
while n<1e6
    S=S+1/n^2;
    n=n+1;
end
```

- **Calcul du produit de Kronecker**
  Ecrire un script ou une fonction qui calcule le produit de Kronecker de deux matrices $C = A \otimes B$ défini par :

$$A_{(n\times m)} \otimes B_{(l\times c)} = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1m}B \\ a_{21}B & a_{22}B & \dots & a_{2m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}B & a_{n2}B & \dots & a_{nm}B \end{bmatrix}_{nl\times mc}$$

  quelles que soient les dimensions de A et B

Fonctions liées au TdS

- **Définir une fonction de transfert** $H(p) = \frac{2p+1}{3p^2+2p+1}$
  ```
  >> H=tf([2 1],[3 2 1])
  ```
- **Calculer / tracer ses diagrammes de Bode**
  ```
  >> bode(H)
  ```
- **Calculer / tracer sa réponse impulsionnelle** $(x(t) = \delta(t))$
  ```
  >> impulse(H)
  ```
- **Calculer / tracer sa réponse indicielle** $(x(t) = \Gamma(t))$
  ```
  >> step(H)
  ```
- **Calculer / tracer sa réponse à** $x(t)$
  ```
  >> t=[0:0.01:10];
  >> x=sin(t);
  >> lsim(H,x,t)
  ```

Reprendre l'exercice 11 du TD 1

- Définir $H(p) = \frac{1}{p+1}$
- Tracer son diagramme de Bode
- Calculer et tracer la réponse de $H(p)$ à $x_1(t) = sin(10t)$
- Calculer et tracer la réponse de $H(p)$ à $x_2(t) = sin(0.1t)$
- Calculer et tracer la réponse de $H(p)$ à $x_3(t) = 2x_1(t) + 4x_2(t)$

Influence du coefficient d'amortissement

- Ecrire un script
  - qui demande de rentrer les valeurs de K, z et ω_0 (fonction `input`)
  - qui définit $H(p)$ la fonction de transfert d'ordre 2 correspondante
  - qui ouvre une figure et y trace le diagramme de Bode (à gauche) et la réponse indicielle (à droite) de $H(p)$
- Vérifier la présence de résonance et/ou d'oscillations selon la valeur de z