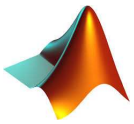
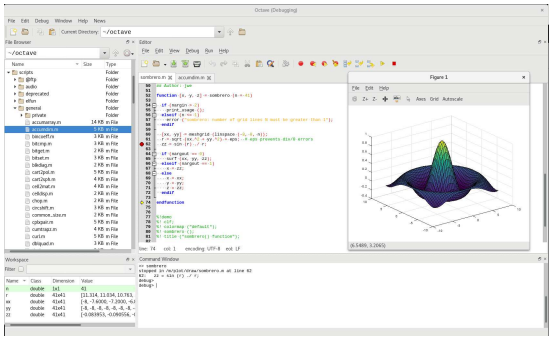


Travaux Pratiques MATLAB : Généralités (Corrigé partiel)

CONTENU :

- **Environnement MATLAB**
- **Opérations élémentaires**
- **Nombres complexes**
- **Les vecteurs**
- **Notion de scripts**
- **Instructions graphiques**
- **Instructions de base**



EXERCICE 1 - *Se familiariser avec l'environnement*

Réinitialisation de l'environnement

1. Dans le menu en haut, Faire : Fenêtre → Rétablir la disposition des fenêtres par défaut
   Quelles sont les fenêtres présentes ? Navigateur, Espace de travail, Fenêtre de commande, Documentation Editeur, Editeur de variables

   Dans la fenêtre de commande :

2. **vider le contenu** de la fenêtre de commandes : clc
3. **Créer 2 variables A et B avec pour valeurs 2 et 5** A = 2 ; B = 5 ;
4. **supprimer toutes les variables** de l'espace de travail clear all et **vérifier la suppression** whos
5. **supprimer l'historique** (avec la commande *history -c*)
6. tapez la commande *pwd*. Identifier alors le chemin de votre dossier de travail : C :\.....\.....

Recherche d'informations sur la Transformée de Fourier

La tranformée de Fourier est identifiée sous MATLAB par la fonction *fft* (Fast Fourier Transform) : dans la fenêtre de commande, **en utilisant la commande adéquate**, recherchez des informations sur la transformée de Fourier : help fft

- Quelles sont les différentes syntaxes possibles pour utiliser la fonction *fft* :
  fft (X), fft (X, N), fft (X, N, DIM)

- Que représentent les variables X et N ? :
  X : signal et N : dimension du signal en nombre d'échantillons

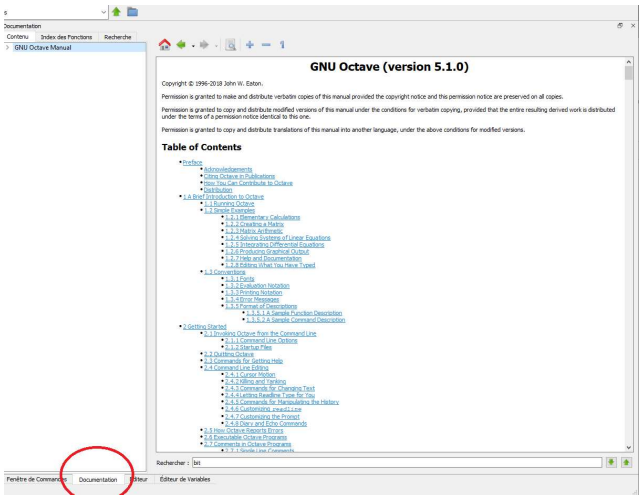
- Citez d'autres fonctions associées à la transformée de Fourier :
  ifft, fft2, fftn, fftw

- Faire une recherche de toutes les fonctions associées à la transformée de Fourier :
  lookfor fft

Utilisation de la documentation

On se propose de faire des opérations binaires avec les nombres $A = 12$ et $B = 9$ en exploitant la documentation interne de MATLAB :

- Conversion du décimal vers le binaire
- Opération de ET LOGIQUE



1. Dans l'invite de commande, réaliser les opérations $A = 12$ et $B = 9$: vérifiez qu'elles sont bien crées avec la commande *whos*. Donnez leur tailles en octets et leur type.

   taille : 8 octets / type : double
2. Avec la commande *dec2bin*, réaliser la conversion en binaire de A et B : dec2bin(A) et dec2bin(B)
   A = 1100 B = 1001
3. Via l'onglet "Documentation" en bas de l'environnement MATLAB, **identifier la commande permettant de réaliser l'opération ET LOGIQUE**
   (Faire : Documentation → Recherche → mot-clé = bit → Clique sur *Bit Manipulation*) bitand(x,y)
4. Dans l'invite de commande, réaliser l'opération ET LOGIQUE : R= (A ET B) : R = bitand(A,B)

5. Convertissez le résultat en binaire et reliez le résultat à la question 2 : Rb = dec2bin(R) R = 1000 = résultat du ET bit à bit entre A et B

EXERCICE 2 - *Opérations élémentaires*

Soit $A = 342$ et $B = 5$

1. Réalisez les opérations suivantes :

$$
\begin{aligned}
R &= A + B \\
R &= A - B \\
R &= A./B \\
R &= A * B \\
R &= 3\hat{}2 \\
R &= \text{sqrt}(A) \\
R &= (A == B) \\
R &= (A \sim= B)
\end{aligned}
$$

2. Que fait l'opération suivante ?

$$R = \text{rem}(A, B) \quad \text{(vous pouvez utiliser l'aide : } \textit{help rem}\text{)}$$

Soit $A = 2.999$

3. Que font les opérations suivantes (vous pouvez utiliser l'aide : *help*)

$$\begin{aligned} R &= \text{floor}(A) \\ R &= \text{ceil}(A) \\ R &= \text{round}(A) \\ R &= \text{fix}(A) \end{aligned}$$

EXERCICE 3 - Nombres complexes

Soit $A = 3 + j * 4$

1. Que font les opérations suivantes (vous pouvez utiliser l'aide : *help*)

$$\begin{aligned} R &= \text{real}(A) \\ R &= \text{imag}(A) \\ R &= \text{abs}(A) \\ R &= \text{arg}(A) \\ R &= \text{atan2}(\text{imag}(A), \text{real}(A)) \\ R &= \text{conj}(A) \\ R &= A. * \text{conj}(A) \\ R &= \text{abs}(A).\hat{}2 \end{aligned}$$

EXERCICE 4 - *Les vecteurs*

On définit les vecteurs ligne de la manière suivante $L_1 = [1, 2, 3]$ et $L_2 = [3, 4, 5]$ et les vecteurs colonne de la manière suivante $C_1 = [6; 7; 8]$ et $C_2 = [9; 10; 11]$

1. Que font les opérations suivantes (vous pouvez utiliser l'aide : *help*)

$$\begin{aligned} N &= \text{size}(L_1) \quad \text{et} \quad M = \text{size}(C_1) \\ V &= L_1.' \quad \text{et} \quad W = C_1.' \\ L &= L_1 \ . + L_2 \\ C &= C_1 \ . * C_2 \\ R &= L_1 \ ./ L_2 \\ V &= [0 : 0.5 : 10] \\ V &= \text{linspace}(0, 10, 100) \\ V &= \text{ones}(1, 4) \\ V &= \text{zeros}(1, 4) \end{aligned}$$

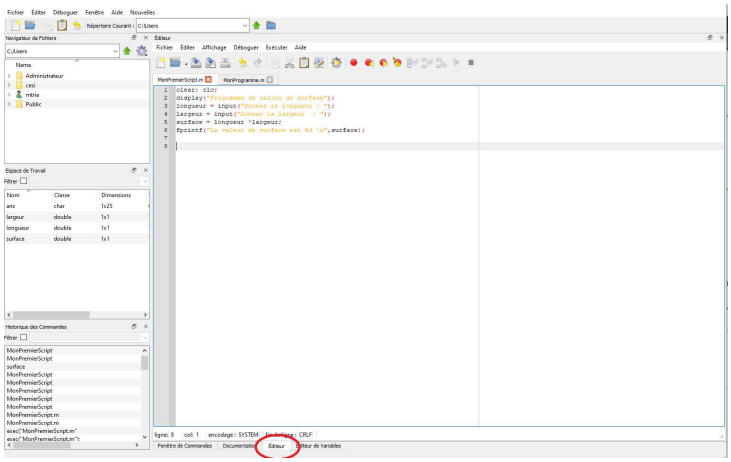
Soit $L_3 = [1 - 2 * i, 2 - 3 * i, 4 - 5 * i]$.

2. Que fait l'opération suivante

$$V = L_3{}' \quad \text{Transposition et forme conjuguée}$$

EXERCICE 5 - *Notions de scripts*

On se propose d'écrire un script qui calcule la surface d'une pièce.



Avant cela, nous allons créer notre *espace de travail* (toutes les étapes ci-dessous sont à faire dans la fenêtre de commande sauf indication contraire) :

1. Placez-vous dans un répertoire de votre choix (avec les commandes cd, cd ..)

2. Dans ce répertoire, créer votre **répertoire de travail** : *mkdir* MonRepertoire

3. Placez-vous dans ce répertoire : *cd* MonRepertoire

4. **Créez un script** avec l'extension .m : *edit* MonProgramme.m (effacer le contenu du fichier + sauvegardez le fichier : Ctrl S)–> vous avez basculé sur l'éditeur

5. **Écrire votre programme** de calcul de surface en débutant par :
   clear ; clc ;
   display("Programme de calcul de surface") ;
   ....

6. Revenir sur la fenêtre de commande et lancez votr programme : >> MonProgramme

A ce stade, tant que vous restez dans votre répertoire de travail, votre fichier est reconnu par MATLAB et vous pouvez donc l'exécuter.

Si vous quittez votre répertoire de travail, vous ne pourrez plus exécuter votre programme : essayez d'exécuter votre programme en dehors de votre répertoire de travail.

Une manière d'avoir son programme reconnu depuis n'importe quel endroit sur l'arborescence, est d'ajouter son chemin absolu dans la base des chemins reconnus par MATLAB :

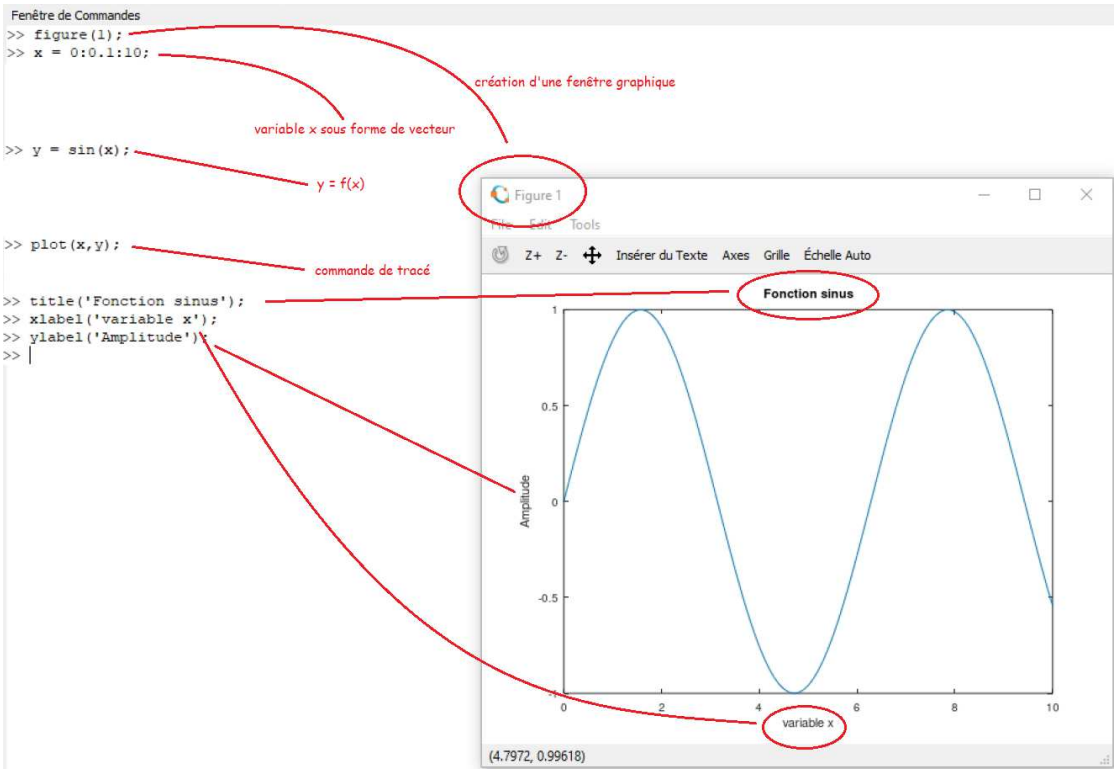
>> addpath('C :\Users\... \ MonRepertoire\') ;

Remarque : vous pouvez faire un copier coller de votre chemin avec la commande *pwd*.

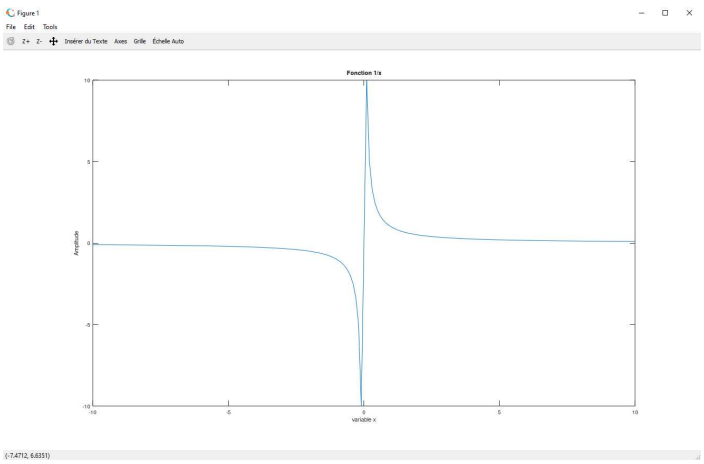
Maintenant, vous pouvez lancer votre programme depuis n'importe quel endroit dans l'arborescence.

EXERCICE 6 - *Instructions graphiques*

Ci-dessous, la liste des commandes pour afficher la courbe d'une fonction $y = f(x)$



On se propose d'afficher la fonction inverse $y = 1/x$ dans l'intervalle $x \in [-10 \quad -0.1] \cup [0.1 \quad 10]$.



Constuire l'intervalle $x \in [-10 \quad -0.1] \cup [0.1 \quad 10]$:

1. Construire un premier vecteur x_1 allant de -10 à -0.1 par pas de 0.1
2. Construire un second vecteur x_2 allant de 0.1 à 10 par pas de 0.1
3. Faire l'union des 2 intervalles : $x = \text{cat}(2, x_1, x_2)$

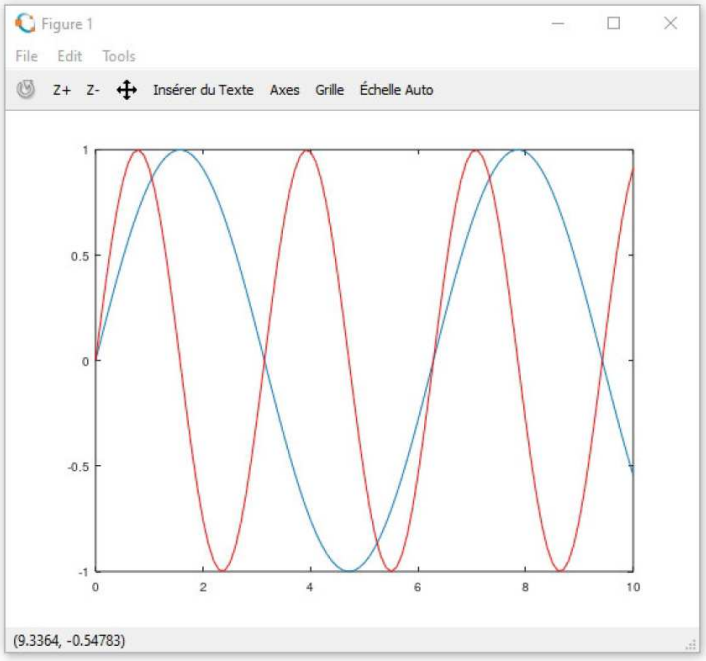
Créer alors et afficher la fonction $y = 1/x$

Ci-dessous, la liste des commandes pour superposer une nouvelle courbe $y2 = g(x)$

```
>> x = 0:0.1:10;
>> y = sin(x);
>> figure(1);
>> plot(x,y)
>> hold on
>> y2 = sin(2*x);
>> plot(x,y2,'r');
>>
```

superposition d'une courbe à une autre : hold on

tracé en rouge : 'r'



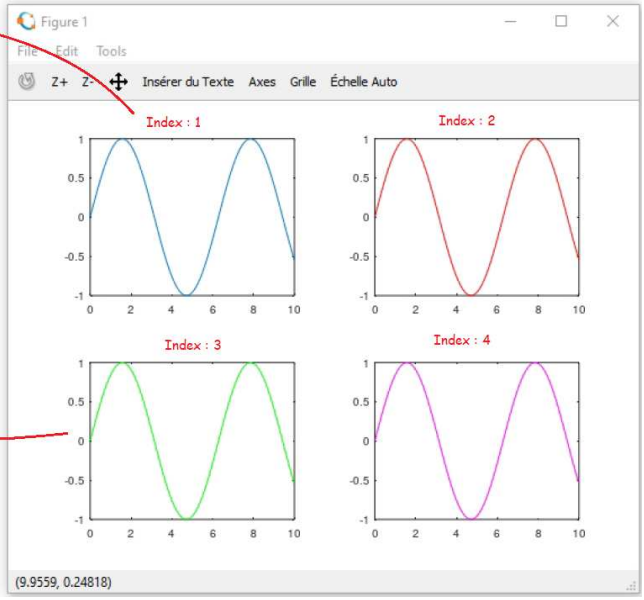
Ci-dessous, la liste des commandes pour afficher plusieurs cadrans

```
Fenêtre de Commandes
>> figure(1)
>> subplot(2,2,1);
>> plot(x,y);
>> subplot(2,2,2);
>> plot(x,y,'r');
>> subplot(2,2,3);
>> plot(x,y,'g');
>> subplot(2,2,4);
>> plot(x,y,'m');
>>
```

Index

nombres de colonnes

nombres de lignes



EXERCICE 7 - *Synthèse des instructions de base*

Réalisez les scripts suivants :

1. Calculer la surface d'un disque de rayon R entré par l'utilisateur (on utilisera la constante *pi* proposé par Matlab). On affichera le résultat à l'écran.
   R = input('Donnez votre rayon') ;
   S = pi*R^2 ;
   disp(['La surface est : ', S]) ;
2. Nous avons déja vu la commande *rem(A,B)* qui permet d'obtenir le reste d'une division A/B. Proposer un script alternatif qui obtient le reste d'une division sans utiliser la commande *rem* (en utilisant simplement l'opération de division, de soustraction et *floor* : on pourra prendre A = 342 et B = 5). Comparer ensuite le résultat obtenu avec le résultat obtenu avec la commande *rem(A,B)*.

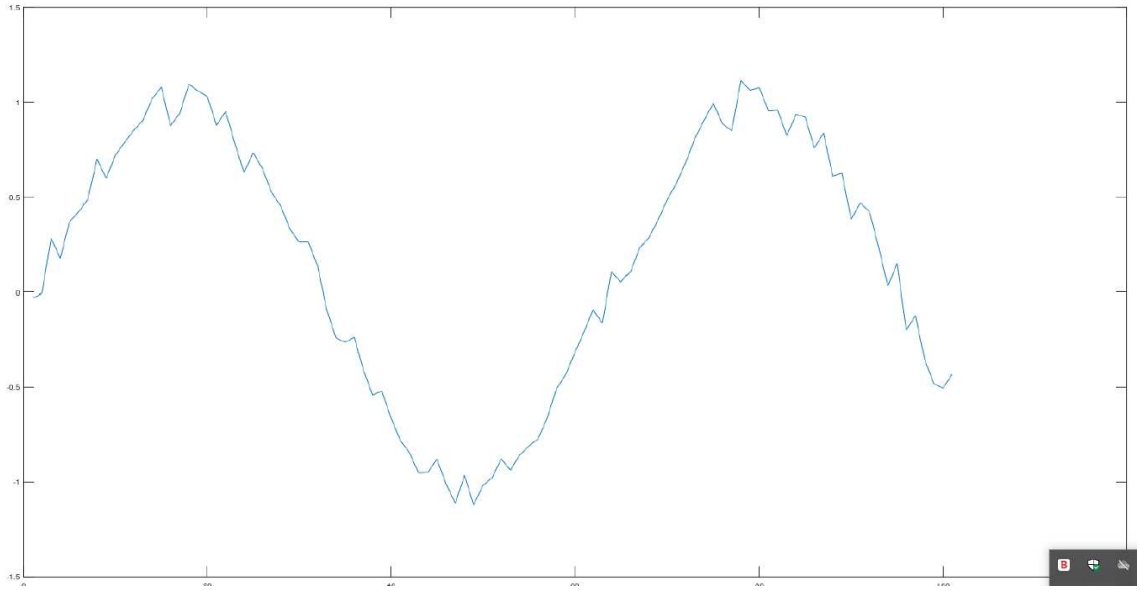
   A = 342 ; B = 5 ;
   quotient = floor(A/B) ;
   R = A - quotient*B ;
3. La commande rand(1) renvoit un nombre aléatoire compris entre 0 et 1. Nous allons utiliser cette commande pour génèner un bruit aléatoire compris entre -0.125 et 0.125 :

   - Au préalable construisez un bruit centré autour de 0, entre -0.5 et 0.5
     b = rand(1)-0.5 ; $b \in [-0.5, 0.5]$
     b = b/4 ; $b \in [-0.125, 0.125]$

   - Pour créer un vecteur constitué de nombres aléatoires, on utilise la commande *rand(1,N)* avec N le nombre déchantillons du vecteur. Faire le test avec N = 4.
     N = 4 ;
     b = rand(1,N)-0.5 ; $b \in [-0.5, 0.5]$
     b = b/4 ; $b \in [-0.125, 0.125]$

   - En combinant les 2 points précédents, on obtient un signal bruit. Ajoutez-le à un signal sinusoidale (voir exercice 6) et afficher le résultat : le résultat doit ressembler au signal ci-dessous :

     - Au préalable, construire la sinuoide y (voir exercice 6)
     - Determiner la taille du vecteur signal : N = length(y)
     - Construire le vecteur bruit et l'ajouter au signal puis afficher le signal bruité


1.5
1
0.5
0
-0.5
-1
-1.5